



Dogram Code

Accede a la Biblioteca Online +300 libros en PDF Gratis!

<https://dogramcode.com/biblioteca>

Únete al Canal en Telegram

[https://t.me/bibliotecagratis\\_dogramcode](https://t.me/bibliotecagratis_dogramcode)

Únete al Grupo en Facebook

<https://www.facebook.com/groups/librosyrecursosdeprogramacion>

# Diseño y construcción de **PÁGINAS WEB**

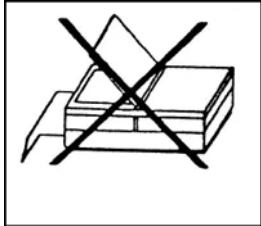


Pablo E. Fernández Casado



# Diseño y construcción de páginas web

Pablo E. Fernández Casado



La ley prohíbe  
fotocopiar este libro  
Diseño y construcción de páginas web  
© Pablo E. Fernández Casado  
© De la edición: Ra-Ma 2020

MARCAS COMERCIALES. Las designaciones utilizadas por las empresas para distinguir sus productos (hardware, software, sistemas operativos, etc.) suelen ser marcas registradas. RA-MA ha intentado a lo largo de este libro distinguir las marcas comerciales de los términos descriptivos, siguiendo el estilo que utiliza el fabricante, sin intención de infringir la marca y solo en beneficio del propietario de la misma. Los datos de los ejemplos y pantallas son ficticios a no ser que se especifique lo contrario.

RA-MA es marca comercial registrada.

Se ha puesto el máximo empeño en ofrecer al lector una información completa y precisa. Sin embargo, RA-MA Editorial no asume ninguna responsabilidad derivada de su uso ni tampoco de cualquier violación de patentes ni otros derechos de terceras partes que pudieran ocurrir. Esta publicación tiene por objeto proporcionar unos conocimientos precisos y acreditados sobre el tema tratado. Su venta no supone para el editor ninguna forma de asistencia legal, administrativa o de ningún otro tipo. En caso de precisarse asesoría legal u otra forma de ayuda experta, deben buscarse los servicios de un profesional competente.

Reservados todos los derechos de publicación en cualquier idioma.

Según lo dispuesto en el Código Penal vigente, ninguna parte de este libro puede ser reproducida, grabada en sistema de almacenamiento o transmitida en forma alguna ni por cualquier procedimiento, ya sea electrónico, mecánico, reprográfico, magnético o cualquier otro sin autorización previa y por escrito de RA-MA; su contenido está protegido por la ley vigente, que establece penas de prisión y/o multas a quienes, intencionadamente, reprodujeren o plagiaren, en todo o en parte, una obra literaria, artística o científica.

Editado por:

RA-MA Editorial

Calle Jarama, 3A, Polígono Industrial Igarsa  
28860 PARACUELLOS DE JARAMA, Madrid

Teléfono: 91 658 42 80

Fax: 91 662 81 39

Correo electrónico: [editorial@ra-ma.com](mailto:editorial@ra-ma.com)

Internet: [www.ra-ma.es](http://www.ra-ma.es) y [www.ra-ma.com](http://www.ra-ma.com)

ISBN: 978-84-9964-996-2

Depósito legal: M-24259-2020

Maquetación: Antonio García Tomé

Diseño de portada: Antonio García Tomé

Filmación e impresión: Safekat

Impreso en España en octubre de 2020

## ÍNDICE

[GLOSARIO DE TÉRMINOS 17](#)

## **1. CONCEPTOS PREVIOS 19**

1. [HERRAMIENTAS Y SOFTWARE A UTILIZAR EN EL CURSO 19](#)
2. [HISTORIA DE LA WEB 23](#)
3. [CÓMO FUNCIONA LA WEB 25](#)
4. [DISEÑO GRÁFICO Y EL PAPEL DEL DISEÑADOR 27](#)
5. [ARQUITECTURA DE LA INFORMACIÓN Y EL PAPEL DEL ARQUITECTO 27](#)
6. [ELEMENTOS DE BÁSICOS DE UNA PÁGINA WEB 29](#)
  1. [Configuración y estructura básica de una página web 29](#)
7. [LOS CONTENIDOS 31](#)
  1. [Tipos de contenido 31](#)
  2. [El árbol de contenidos 33](#)
8. [LA INTERACCIÓN 38](#)
  1. [Mecanismos lingüísticos 38](#)
  2. [Mecanismos contextuales 38](#)
  3. [Mecanismos por manipulación directa 41](#)
  4. [Mecanismos de interacción social 42](#)
9. [LAS BASES DE DATOS 43](#)
  1. [Bases de datos jerárquicas 43](#)
  2. [Base de datos de red 44](#)
  3. [Bases de datos transaccionales 45](#)
  4. [Bases de datos relacionales 46](#)
  5. [Bases de datos orientadas a objetos 48](#)
  6. [Bases de datos documentales 50](#)
  7. [Bases de datos multidimensionales 51](#)
  8. [Bases de datos deductivas 53](#)
10. [PRINCIPALES TIPOLOGÍAS DE WEB 54](#)
  1. [Según su diseño o estructura 54](#)
  2. [Según su funcionalidad u objetivo 54](#)
11. [LENGUAJES DE MARCADO 57](#)
  1. [Características de los lenguajes de marcado 58](#)
  2. [Clasificación de los lenguajes de marcado 59](#)

## **2. HTML5 63**

1. [ELEMENTOS BÁSICOS DE HTML5 64](#)
  1. [Definición del tipo de documento DTD \(!DOCTYPE\) 64](#)
  2. [Etiqueta html 66](#)
  3. [Etiqueta head 67](#)
  4. [Etiqueta body 67](#)
  5. [Comentarios 68](#)
2. [ATRIBUTOS COMUNES O GENÉRICOS 68](#)
  1. [Atributo accesskey 68](#)
  2. [Atributo autocapitalize 69](#)
  3. [Atributo autocomplete 69](#)
  4. [Atributo class 70](#)
  5. [Atributo contenteditable 70](#)
  6. [Atributo draggable 71](#)
  7. [Atributo contextmenu 71](#)

8. [Atributo dir](#) 71
9. [Atributo enterkeyhint](#) 71
10. [Atributo hidden](#) 72
11. [Atributo inputmode](#) 73
12. [Atributo id](#) 73
13. [Atributo is](#) 74
14. [Atributo itemid](#) 74
15. [Atributo itemprop](#) 75
16. [Atributo itemref](#) 75
17. [Atributo itemscope](#) 76
18. [Atributo itemtype](#) 76
19. [Atributo lang](#) 77
20. [Atributo nonce](#) 77
21. [Atributo slot](#) 78
22. [Atributo style](#) 78
23. [Atributo title](#) 78
24. [Atributo tabindex](#) 78
25. [Atributos personalizados](#) 79

### 3. [INFORMACIÓN DEL DOCUMENTO](#) 80

1. [Elemento base](#) 80
2. [Elemento link](#) 80
3. [Elemento meta](#) 82
4. [Elemento style](#) 82
5. [Elemento script](#) 83

### 4. [ESTRUCTURA DEL DOCUMENTO](#) 83

1. [Elemento article](#) 84
2. [Elemento aside](#) 84
3. [Elemento div](#) 85
4. [Elemento footer](#) 85
5. [Elemento header](#) 86
6. [Elementos h1..h6](#) 86
7. [Elemento hr](#) 87
8. [Elemento hgroup](#) 87
9. [Elemento main](#) 88
10. [Elemento nav](#) 88
11. [Elemento section](#) 89

### 5. [FORMATEADO DEL TEXTO](#) 90

1. [Elemento abbr](#) 90
2. [Elemento acronym](#) 90
3. [Elemento address](#) 90
4. [Elementos b y strong](#) 91
5. [Elemento blockquote](#) 91
6. [Elemento q](#) 91
7. [Elemento cite](#) 92
8. [Elemento code](#) 92
9. [Elementos ins y del](#) 92
10. [Elemento em](#) 93

- 11. [Elemento i](#) 93
  - 12. [Elemento pre](#) 93
  - 13. [Elemento sub](#) 94
  - 14. [Elemento sup](#) 94
6. [LISTAS](#) 94.
- 1. [Elemento UL](#) 94
  - 2. [Elemento OL](#) 95
  - 3. [Elemento DL](#) 96
7. [ENLACES](#) 97.
- 1. [Elemento A](#) 97
8. [PRÁCTICA 1: ESTRUCTURA BÁSICA DE UNA PÁGINA HTML](#) 5.98
- 1. [Resolución](#) 98
9. [PRÁCTICA 2: ESTRUCTURA DE “UNIVERSES”](#) 100
- 1. [Resolución](#) 101
3. [CSS3](#) 103
- 1. [SOPORTE A LOS NAVEGADORES](#) 103
  - 2. [CÓMO FUNCIONA CSS](#) 104
  - 3. [DEFINICIÓN DE SELECTOR Y CLASIFICACIÓN](#) 104
  - 4. [UNIDADES DE MEDIDA](#) 106
    - 1. [Unidades absolutas](#) 106
    - 2. [Unidades relativas](#) 106
  - 5. [CODIFICACIÓN DE COLORES](#) 108
    - 1. [Codificación RGB y RGBA](#) 108
    - 2. [Codificación Hexadecimal](#) 109
    - 3. [Codificación HSL y HSLA](#) 110
  - 6. [PROPIEDADES](#) 110
    - 1. [Texto, fuentes y tipos de letra](#) 110
    - 2. [Márgenes internos y externos](#) 122
    - 3. [Bordes](#) 124
    - 4. [Colores de fondo](#) 131
    - 5. [Listas](#) 132
    - 6. [Posicionamiento](#) 134
    - 7. [Comportamientos y tamaños](#) 140
    - 8. [Diseño de cajas flexibles](#) 146
    - 9. [Interfaz de usuario](#) 157
    - 10. [Animaciones y transiciones](#) 161
    - 11. [Reglas arroba](#) 163
  - 7. [FUNCIONES](#) 173
    - 1. [Funciones de pseudo-elementos](#) 173
    - 2. [Funciones de cálculo](#) 174
    - 3. [Funciones gráficas](#) 175
  - 8. [HACKS](#) 180
  - 9. [SELECTORES](#) 182
    - 1. [Simples y combinados](#) 183
    - 2. [Pseudo-clases](#) 186
    - 3. [Pseudo-elementos](#) 196

- 4. [Especificidad de los selectores 198](#)
- 10. [VARIABLES 199](#)
- 11. [PRÁCTICA 3: PRIMEROS ESTILOS DE “UNIVESES” 201](#)
  - 1. [Resolución 202](#)
- 4. IMÁGENES Y MULTIMEDIA 205**
  - 1. [TIPOS DE IMÁGENES 205](#)
  - 2. [ELEMENTOS DISPONIBLES EN HTML5 206](#)
    - 1. [Elemento audio 206](#)
    - 2. [Elementos figure y figcaption 207](#)
    - 3. [Elemento img 208](#)
    - 4. [Elementos map y area 209](#)
    - 5. [Elemento picture 210](#)
    - 6. [Elemento source 211](#)
    - 7. [Elemento track 212](#)
    - 8. [Elemento video 212](#)
  - 3. [PROPIEDADES DISPONIBLES EN CSS 213](#)
    - 1. [Propiedad background-attachment 213](#)
    - 2. [Propiedad background-clip 214](#)
    - 3. [Propiedad background-image 214](#)
    - 4. [Propiedad background-origin 216](#)
    - 5. [Propiedad background-position 216](#)
    - 6. [Propiedad background-repeat 217](#)
    - 7. [Propiedad background-size 218](#)
    - 8. [Propiedad clip-path 218](#)
    - 9. [Propiedad object-fit 220](#)
    - 10. [Propiedad object-position 221](#)
  - 4. [IMÁGENES RECEPTIVAS Y ADAPTATIVAS 222](#)
    - 1. [Adaptación de imágenes mediante propiedades CSS 224](#)
    - 2. [Adaptación de imágenes mediante consultas de medios 226](#)
    - 3. [Selección de imágenes mediante IMG y SRCSET 227](#)
    - 4. [Selección de imágenes mediante PICTURE y SOURCE 229](#)
  - 5. [VIDEOS RECEPTIVOS 230](#)
  - 6. [USABILIDAD Y ACCESIBILIDAD EN LOS ELEMENTOS MULTIMEDIA 231](#)
    - 1. [En imágenes 231](#)
    - 2. [En audio y vídeo 232](#)
  - 7. [PRÁCTICA 4: ESTRUCTURA, TEXTOS E IMÁGENES DE “UNIVERSES” 233](#)
    - 1. [Recursos para hacer la práctica 234](#)
    - 2. [Resolución 237](#)
  - 8. [PRÁCTICA 5: BANNER DE “UNIVERSES” A PANTALLA COMPLETA 238](#)
    - 1. [Resolución 239](#)
- 5. TABLAS 241**
  - 1. [ELEMENTOS DISPONIBLES EN HTML5 241](#)
    - 1. [Elemento caption 241](#)
    - 2. [Elemento table 242](#)
    - 3. [Elemento colgroup 243](#)
    - 4. [Elemento thead 244](#)

- 5. [Elemento tbody](#) 244
  - 6. [Elemento tfoot](#) 245
  - 7. [Elemento tr](#) 245
  - 8. [Elemento th](#) 245
  - 9. [Elemento td](#) 246
2. [ELEMENTOS DISPONIBLES EN CSS](#) 247
- 1. [Propiedad border-collapse](#) 247
  - 2. [Propiedad border-spacing](#) 248
  - 3. [Propiedad caption-side](#) 248
  - 4. [Propiedad empty-cells](#) 248
  - 5. [Propiedad table-layout](#) 249
3. [CREACIÓN DE TABLAS RESPONSIVE](#) 249
- 1. [Mediante barras de desplazamiento](#) 250
  - 2. [Mediante consultas de medios](#) 252
  - 3. [Mediante ocultación de columnas a petición de usuario](#) 255
4. [USABILIDAD Y ACCESIBILIDAD EN LAS TABLAS](#) 261
5. [PRÁCTICA 6: PÁGINA “DÓNDE ESTAMOS” DE “UNIVERES”](#) 261
- 1. [Recursos para hacer la práctica](#) 263
  - 2. [Resolución](#) 265
- 6. FORMULARIOS 269**
1. [TIPOS DE FORMULARIO](#) 270
- 1. [Formularios de contacto](#) 270
  - 2. [Formularios de suscripción](#) 272
  - 3. [Formularios de acceso](#) 272
  - 4. [Formularios de registro](#) 273
  - 5. [Formularios de entrada general](#) 275
2. [ELEMENTOS DISPONIBLES EN HTML5](#) 275
- 1. [Elemento form](#) 275
  - 2. [Elemento button](#) 277
  - 3. [Elemento datalist](#) 279
  - 4. [Elemento fieldset](#) 279
  - 5. [Elemento input](#) 280
  - 6. [Elemento label](#) 285
  - 7. [Elemento legend](#) 285
  - 8. [Elemento meter](#) 286
  - 9. [Elemento optgroup](#) 287
  - 10. [Elemento option](#) 287
  - 11. [Elemento output](#) 288
  - 12. [Elemento progress](#) 289
  - 13. [Elemento select](#) 290
  - 14. [Elemento textarea](#) 291
3. [ELEMENTOS DISPONIBLES EN CSS](#) 292
- 1. [Propiedad box-sizing](#) 292
  - 2. [Propiedad resize](#) 293
4. [VALIDACIÓN DE FORMULARIOS](#) 293
- 1. [La interfaz validitystate](#) 293

- 2. [Propiedades y métodos](#) [294](#)
- 3. [Eventos](#) [295](#)
- 5. [\*\*USABILIDAD Y ACCESIBILIDAD EN LOS FORMULARIOS\*\*](#) [297](#)
  - 1. [Autocompletado](#) [298](#)
  - 2. [Estructuración y optimización](#) [298](#)
  - 3. [Capacidades de los dispositivos](#) [298](#)
  - 4. [Combos o desplegables](#) [299](#)
  - 5. [Número de campos](#) [299](#)
  - 6. [Nombres de campo](#) [299](#)
  - 7. [Tipos de datos estándar](#) [299](#)
  - 8. [Botones y enlaces](#) [300](#)
  - 9. [Rangos de valores](#) [300](#)
- 6. [\*\*PRÁCTICA 7: PÁGINA DE ACCESO DE “UNIVERES”\*\*](#) [301](#)
  - 1. [Recursos para hacer la práctica](#) [302](#)
  - 2. [Resolución](#) [303](#)
- 7. [\*\*PRÁCTICA 8: EXPERIMENTO DE PÁGINA DE CONTACTO DE “UNIVERES”\*\*](#) [306](#)
  - 1. [Recursos para hacer la práctica](#) [307](#)
  - 2. [Resolución](#) [307](#)
- 7. [\*\*IFRAMES Y OBJETOS\*\*](#) [311](#)
  - 1. [ELEMENTOS DISPONIBLES EN HTML](#) [5.311](#)
    - 1. [Elemento iframe](#) [311](#)
    - 2. [Elemento object](#) [312](#)
    - 3. [Elemento embed](#) [313](#)
  - 2. [ELEMENTOS DISPONIBLES EN CSS](#) [314](#)
  - 3. [USABILIDAD Y ACCESIBILIDAD EN LOS MARCOS](#) [314](#)
  - 4. [IFRAMES Y OBJETOS MULTIMEDIA RESPONSIVE](#) [314](#)
- 8. [\*\*ANIMACIONES, TRANSICIONES Y EFECTOS\*\*](#) [317](#)
  - 1. [TRANSFORMACIONES](#) [317](#)
    - 1. [Función de escalado \(scale\)](#) [317](#)
    - 2. [Función de rotación \(rotate\)](#) [318](#)
    - 3. [Función de sesgado \(skew\)](#) [319](#)
    - 4. [Función de traslado \(translate\)](#) [320](#)
  - 2. [FILTROS O EFECTOS](#) [321](#)
    - 1. [Desenfoque \(blur\)](#) [321](#)
    - 2. [Brillo o iluminación \(brightness\)](#) [321](#)
    - 3. [Contraste \(contrast\)](#) [322](#)
    - 4. [Sombra paralela \(drop-shadow\)](#) [323](#)
    - 5. [Escala de grises \(grayscale\)](#) [324](#)
    - 6. [Rotación de color \(hue-rotate\)](#) [324](#)
    - 7. [Inversión \(invert\)](#) [326](#)
    - 8. [Opacidad \(opacity\)](#) [326](#)
    - 9. [Saturación \(saturate\)](#) [327](#)
    - 10. [Sepia \(sepia\)](#) [328](#)
  - 3. [TRANSICIONES](#) [328](#)
    - 1. [Propiedad transition-delay](#) [329](#)
    - 2. [Propiedad transition-duration](#) [329](#)

3. [Propiedad transition-property](#)<sub>330</sub>
  4. [Propiedad transition-timing-function](#)<sub>330</sub>
  5. [Animaciones](#)<sub>331</sub>
4. [PRÁCTICA 9: DISEÑO DE UN PLANETA EN 3D](#)<sub>336</sub>
    1. [Recursos para hacer la práctica](#)<sub>337</sub>
    2. [Resolución](#)<sub>338</sub>
  5. [PRÁCTICA 10: CREACIÓN DE UN LOADER](#)<sub>339</sub>.
    1. [Resolución](#)<sub>340</sub>

## **9. METADATOS**<sub>345</sub>

1. [ATRIBUTOS ASOCIADOS A LOS METADATOS](#)<sub>345</sub>
  1. [Atributo name](#)<sub>345</sub>
  2. [El atributo content](#)<sub>346</sub>
  3. [El atributo charset](#)<sub>346</sub>
  4. [El atributo http-equiv](#)<sub>346</sub>
  5. [El atributo scheme](#)<sub>347</sub>
2. [PRINCIPALES METADATOS HTML](#)<sub>347</sub>
  1. [Allow](#)<sub>348</sub>
  2. [Application-Name](#)<sub>348</sub>
  3. [Author](#)<sub>348</sub>
  4. [Cache-Control](#)<sub>348</sub>
  5. [Content-Disposition](#)<sub>349</sub>.
  6. [Content-Encoding](#)<sub>349</sub>.
  7. [Content-Language](#)<sub>350</sub>
  8. [Content-Script-Type](#)<sub>350</sub>
  9. [Copyright](#)<sub>350</sub>
  10. [Default-Style](#)<sub>350</sub>
  11. [Description](#)<sub>351</sub>
  12. [Date](#)<sub>351</sub>
  13. [Expires](#)<sub>352</sub>
  14. [Generator](#)<sub>352</sub>
  15. [Keywords](#)<sub>352</sub>
  16. [Last-modified](#)<sub>353</sub>
  17. [Location](#)<sub>353</sub>
  18. [Rating](#)<sub>353</sub>
  19. [Refresh](#)<sub>354</sub>
  20. [Robots, MSNbot y Googlebot](#)<sub>354</sub>
  21. [Viewport](#)<sub>355</sub>
  22. [X-UA-compatible](#)<sub>357</sub>
3. [DUBLIN CORE](#)<sub>358</sub>
  1. [Creator](#)<sub>358</sub>
  2. [Contributor](#)<sub>359</sub>.
  3. [Coverage](#)<sub>359</sub>.
  4. [Date](#)<sub>360</sub>
  5. [Description](#)<sub>361</sub>
  6. [Format](#)<sub>361</sub>
  7. [Identifier](#)<sub>362</sub>
  8. [Language](#)<sub>362</sub>

- 9. [Publisher](#) 363
  - 10. [Relation](#) 363
  - 11. [Rights](#) 364
  - 12. [Source](#) 364
  - 13. [Subject](#) 365
  - 14. [Type](#) 365
  - 15. [Title](#) 366
4. [OPEN GRAPH](#) 366
- 1. [Description](#) 366
  - 2. [Image](#) 367
  - 3. [Locale](#) 367
  - 4. [Title](#) 368
  - 5. [Type](#) 368
  - 6. [URL](#) 368
  - 7. [Video](#) 369
5. [TWITTER CARDS](#) 370
- 1. [twitter:card](#) 370
  - 2. [twitter:site](#) 371
  - 3. [twitter:site:id](#) 371
  - 4. [twitter:creator](#) 371
  - 5. [twitter:creator:id](#) 371
  - 6. [twitter:description](#) 371
  - 7. [twitter:title](#) 372
  - 8. [twitter:image](#) 372
  - 9. [twitter:image:alt](#) 372
  - 10. [twitter:player](#) 372
  - 11. [twitter:app](#) 372
6. [GOOGLE](#) 373
- 1. [nositelinkssearchbox](#) 373
  - 2. [notranslate](#) 373
  - 3. [nopagereadaloud](#) 374
  - 4. [google-site-verification](#) 374
7. [PRÁCTICA 11: ASIGNACIÓN DE METADATOS](#) 374
- 1. [Resolución](#) 374
- 10. [SVG](#) 377**
- 1. [INTRODUCCIÓN](#) 377
  - 2. [PRIMITIVAS BÁSICAS DE SVG](#) 378
    - 1. [Primitiva svg](#) 378
    - 2. [Primitiva defs](#) 381
    - 3. [Primitiva rect](#) 382
    - 4. [Primitiva circle](#) 383
    - 5. [Primitiva ellipse](#) 383
    - 6. [Primitiva line](#) 384
    - 7. [Primitiva polyline](#) 384
    - 8. [Primitiva polygon](#) 385
    - 9. [Primitiva a](#) 385
    - 10. [Primitiva path](#) 386

- 11. [Primitiva image](#) 391
- 3. [TEXTOS](#) 392
  - 1. [Primitiva text](#) 392
  - 2. [Primitiva tspan](#) 393
  - 3. [Primitiva textPath](#) 393
- 4. [GRADIENTES](#) 394
  - 1. [Primitiva lineargradient](#) 394
  - 2. [Primitiva stop](#) 396
  - 3. [Primitiva radialgradient](#) 396
- 5. [PATRONES](#) 398
- 6. [ANIMACIONES](#) 401
  - 1. [Attributos comunes a todos los elementos de animación](#) 402
  - 2. [Primitiva animate](#) 403
  - 3. [Primitiva set](#) 406
  - 4. [Primitiva animatemotion](#) 407
  - 5. [Primitiva animatetransform](#) 407
- 7. [FILTROS](#) 408
  - 1. [Antes de nada](#) 408
  - 2. [Compensaciones: primitiva feOffset](#) 409
  - 3. [Inundar: primitiva feFlood](#) 410
  - 4. [Imágenes: primitiva feImage](#) 410
  - 5. [Azulejos: primitiva feTile](#) 412
  - 6. [Fusionar: primitiva feBlend](#) 413
  - 7. [Desenfoque gausiano: primitiva feGaussianBlur](#) 417
  - 8. [Filtro matrices de color: primitiva feColorMatrix](#) 418
  - 9. [Componentes de transferencia: primitiva feComponentTransfer](#) 421
  - 10. [Composiciones: primitiva feComposite](#) 424
  - 11. [Matrices de convolución: primitiva feConvolveMatrix](#) 426
  - 12. [Iluminaciones: primitivas feEsellarLighting y feDiffuseLighting](#) 429
  - 13. [Mapas de desplazamiento: primitiva feDisplacementMap](#) 435
  - 14. [Morfologías: primitiva feMorphology](#) 436
  - 15. [Combinaciones: primitiva feMerge](#) 438
  - 16. [Turbulencias: primitiva feTurbulence](#) 440

## **11. USABILIDAD WEB 443**

- 1. [DEFINICIÓN DE USABILIDAD WEB](#) 443
  - 1. [Usabilidad objetiva o inherente](#) 445
  - 2. [Usabilidad subjetiva o aparente](#) 445
  - 3. [Cómo se puede asegurar la usabilidad](#) 445
- 2. [IMPORTANCIA DEL DISEÑO WEB CENTRADO EN EL USUARIO](#) 445
- 3. [DIFERENCIA ENTRE USABILIDAD Y ACCESIBILIDAD WEB](#) 446
- 4. [VENTAJAS E INCONVENIENTES DE CREAR SITIOS WEB USABLES](#) 447
- 5. [ANÁLISIS DE REQUERIMIENTOS DE USUARIO](#) 448
  - 1. [Diseño conceptual](#) 448
  - 2. [Diseño participativo o cooperativo \(DPoC\)](#) 449
  - 3. [Design Thinking](#) 449
  - 4. [Esqueumorfismo](#) 450

- 6. [CREACIÓN DE PROTOTIPOS ORIENTADOS AL USUARIO](#) 451
  - 1. [El proceso de prototipado](#) 451
  - 2. [Tipos de prototipos](#) 452
  - 3. [El proceso de desarrollo de un prototipo](#) 456
- 7. [PAUTAS PARA LA CREACIÓN DE SITIOS WEB USABLES](#) 461
  - 1. [Visibilidad del estado del sistema](#) 461
  - 2. [Relación entre el sistema y los usuarios](#) 461
  - 3. [Control y libertad para el usuario](#) 461
  - 4. [Consistencia y estándares](#) 462
  - 5. [Prevención de errores](#) 462
  - 6. [Reconocer antes que recordar](#) 462
  - 7. [Flexibilidad y eficiencia de uso](#) 463
  - 8. [Diseño estético y minimalista](#) 463
  - 9. [Reconocimiento, diagnóstico y recuperación de errores](#) 463
  - 10. [Ayuda y documentación](#) 464
- 8. [PRÁCTICA 12: VERIFICANDO Y MEJORANDO LA USABILIDAD WEB DE UNIVERSES](#) 464
  - 1. [Resolución](#) 464.
- 12. [ACCESIBILIDAD WEB](#) 477
  - 1. [DEFINICIÓN DE ACCESIBILIDAD WEB](#) 477
    - 1. [Tipos de discapacidad](#) 478
    - 2. [Ventajas e inconvenientes de la accesibilidad web](#) 479
  - 2. [TECNOLOGÍAS DÓNDE LA ACCESIBILIDAD ES APLICABLE](#) 480
    - 1. [HTML y XHTML](#) 480
    - 2. [CSS](#) 480
    - 3. [JavaScript](#) 481
    - 4. [Flash](#) 481
    - 5. [PDF](#) 482
    - 6. [XML/XSL](#) 482
    - 7. [Reproducción multimedia](#) 482
    - 8. [Otras tecnologías](#) 483
  - 3. [NORMATIVA Y ESTÁNDARES SOBRE ACCESIBILIDAD WEB](#) 484
    - 1. [Norma EN 301 549:2018](#) 484.
    - 2. [Norma UNE 139803:2012](#) 484
    - 3. [Estándar ISO/IEC 40500:2012](#) 485
    - 4. [Comparativa de estándares sobre accesibilidad web](#) 485
  - 4. [EL ESTÁNDAR SMIL](#) 486
    - 1. [Módulos de SMIL](#) 486
    - 2. [Ejemplo de contenido SMIL](#) 487
  - 5. [LA INICIATIVA WAI ARIA](#) 488
    - 1. [Partes de la WAI ARIA](#) 488
    - 2. [Soporte en navegadores y productos de apoyo](#) 491
    - 3. [Principales atributos de la WAI-ARIA](#) 492
  - 6. [LA RECOMENDACIÓN WCAG](#) 501
    - 1. [Principios y directrices](#) 501
    - 2. [Criterios de Conformidad por nivel de adecuación](#) 503
    - 3. [Descripción de las pautas de accesibilidad web](#) 503

4. [Proceso de la conformidad en accesibilidad web 520](#)
  7. [HERRAMIENTAS PARA LA VALIDACIÓN DE LA ACCESIBILIDAD WEB 522](#)
    1. [Basadas en navegador 522](#)
    2. [Mediante aplicaciones de escritorio 524](#)
    3. [Mediante servicios web externos 526](#)
    4. [Evolución de la accesibilidad. Nuevas tendencias 528](#)
  8. [PRÁCTICA 13: VERIFICANDO Y MEJORANDO LA ACCESIBILIDAD WEB DE UNIVERSES 528](#)
    1. [Resolución 529](#)
- 13. DISEÑO DE EMAILS BASADOS EN HTML 535**
1. [USO DE JAVASCRIPT 535](#)
  2. [TIPO DE DOCUMENTO A UTILIZAR 536](#)
  3. [PECULIARIDADES DE LOS CLIENTES DE CORREO 536](#)
  4. [EL MODO OSCURO 537](#)
  5. [SECCIÓN DE CABECERA 538](#)
  6. [SECCIÓN DEL CUERPO 539](#)
  7. [LÍMITES Y LIMITACIONES DE CSS 541](#)
  8. [USO DE IMÁGENES Y VÍDEOS 542](#)
  9. [HERRAMIENTAS PARA LA CREACIÓN DE EMAILS 542](#)
  10. [PRÁCTICA 14: NEWSLETTER DE UNIVERSES 544](#)
    1. [Recursos para hacer la práctica 546](#)
    2. [Resolución 546](#)
- 14. DIRECTRICES PARA LA CREACIÓN DE GUÍAS DE ESTILOS 547**
1. [EJEMPLOS DE GUÍAS DE ESTILO 549](#)

## [REFERENCIAS 551](#)

# GLOSARIO DE TÉRMINOS

A continuación, se explican algunos conceptos previos que se deben tener antes de empezar con el curso.

Concepto	Descripción
<b>Accesibilidad Web</b>	La accesibilidad web es una de las partes que engloba la usabilidad web. Mientras que la usabilidad web, entre otras cosas, se centra en el rendimiento, la semántica y la universalidad, la accesibilidad web, se asegura de que todo tipo de usuario tenga o no una limitación, discapacidad o incapacidad, puedan usar las páginas y aplicaciones con una experiencia de usuario óptima.
<b>Agente de usuario</b>	Es una programa informático o software que funciona como interfaz de interacción web o cliente de red. Habitualmente, este tipo de software hace referencia a los navegadores web y herramientas de asistencia a la accesibilidad como puedan ser los magnificadores de pantalla, los lectores de pantalla, etcétera.
<b>Diseño Gráfico</b>	Podría definirse como una disciplina que consiste en presentar información visual y cuyo objetivo es que, los usuarios, capten mensajes específicos sobre un tema o materia determinada.
<b>IDE</b>	IDE es un acrónimo que significa Entorno de Desarrollo Integrado. Su función, es la de mejorar la productividad de los desarrolladores valiéndose de unas herramientas para la edición de código, de construcción automáticas que evitan, entre otras cosas, errores de sintaxis y de estructuración y, un depurador. También pueden contener un compilador y/o intérprete como es el caso de NetBeans o Eclipse.

<b>Lenguaje de marcado</b>	Un lenguaje de marcado es un mecanismo para codificar documentos con todo tipo de objetos, sean textuales o no textuales. Esta codificación, a menudo, permite el uso de etiquetas y atributos para proporcionar información adicional a la estructura del texto o su presentación.
<b>Metadato</b>	Un metadato puede definirse como información adicional acerca de los propios datos. Muchos autores lo definen como datos sobre los datos, que es una forma de decir el contexto de los datos o a qué se refieren esos datos. Normalmente, el identificador del metadato ya indica su objetivo y, su valor, indica la explicación o contextualización.
<b>Microdato</b>	Los microdatos son unas marcas adicionales que se emplean para anidar metadatos sobre una información concreta dentro de un documento basado en lenguaje de marcado.
<b>Polyfill</b>	Un polyfill es un componente o fragmento de código que habilita o posibilita una funcionalidad que el agente de usuario no es capaz de proporcionar, ya sea porque es antiguo, ya sea porque utiliza un estándar incompatible.
<b>Mobile First</b>	Mobile First es una disciplina que considera que las páginas se deben diseñar dando prioridad a los dispositivos móviles, es decir, primero se tiene en cuenta el diseño para un dispositivo móvil y, si el escenario lo permite o es diferente, se le aplican una serie de reglas para su adecuado funcionamiento y correcta visualización.
<b>Seguridad Web</b>	La seguridad web se refiere cómo se debe contemplar o proveer la seguridad de la información en las páginas y aplicaciones, sean del tipo que sean. Su objetivo es evitar ataques de usuarios malintencionados y de softwares diseñados con un propósito no lícito o ético. En otras palabras, es una disciplina que permite proteger contra el acceso, manipulación, destrucción e interrupción de información no autorizado.
<b>Semántica Web</b>	La semántica web es un conjunto de recomendaciones y estándares desarrolladas por la W3C (World Wide Web Consortium) que están pensadas para hacer que los datos se vuelvan más legibles.
<b>Usabilidad Web</b>	En términos generales, la usabilidad se refiere a la facilidad con la que las personas utilizan los programas y máquinas que deben de manejar. En el campo de la informática, la usabilidad se refiere a la facilidad con la que las personas manejan las páginas y aplicaciones, sean web o móviles. Por tanto, un diseño usable es aquel que está centrado en el usuario.

# 1

## CONCEPTOS PREVIOS

### HERRAMIENTAS Y SOFTWARE A UTILIZAR EN EL CURSO

A continuación, se expone un resumen más o menos detallado sobre las herramientas y servicios que resultan beneficiosas para el aprendizaje y creación de sitios web. En muchos casos, se utilizarán o podrán utilizar a lo largo de este libro.

#### VISUAL STUDIO CODE

Visual Studio Code es un IDE (Entorno de Desarrollo Integrado) desarrollado por Microsoft que permite editar los archivos en modo texto. Esto es necesario para modificar los códigos fuente de las páginas que vamos a crear.

Además, presenta multitud de extensiones entre las que se incluyen soporte para la depuración, control integrado de Git y el resaltado de sintaxis.

Es descargable desde:

<https://code.visualstudio.com/download>

## FUENTES VECTORIALES

Una fuente vectorial es una biblioteca de elementos que representa letras, símbolos y caracteres creados a partir de un conjunto de vectores que responden a unos valores numéricos. La principal ventaja que tiene el formato vectorial es que, los elementos, pueden ser aumentados o reducidos sin pérdida alguna de calidad.

Entre los repositorios de fuentes vectoriales más conocidos podemos encontrar GOOGLE FONTS y ADOBE FONTS. Ambas pueden ser una buena opción, sin embargo, nosotros utilizaremos las posibilidades de Google Fonts, primero porque es gratuita y, segundo, porque está más extendida a nivel mundial.

### URL de consulta y ayuda:

<https://fonts.google.com/>

Un detalle importante sobre Google Fonts es que contiene gran cantidad de fuentes vectoriales que nos ayudarán a crear diseños de gran calidad y con formatos legibles compatibles para todos los dispositivos, sean móviles o no.

Otro tipo de fuentes vectoriales que se suelen utilizar en casi todos los diseños son las fuentes vectoriales icónicas. A diferencia de las anteriores, las fuentes vectoriales icónicas, presentan imágenes a modo de ícono que pueden ser manipuladas como si de texto se tratase a través de CSS.

Las fuentes vectoriales icónicas más conocidas quizás sean FontAwesome y Material Icons. Ambas nos proporcionan la posibilidad de aportar un modo gráfico moderno sin repercutir mucho en el rendimiento y tamaño de las páginas.

### URLS de consulta y descarga:

<https://material.io/resources/icons/?style=baseline>

<https://fontawesome.com/>

## IMÁGENES E ICONOS

Aunque las imágenes con fines decorativos son un recurso que, afortunadamente, está cada vez más en desuso, siempre serán necesarias en otras circunstancias. Buen ejemplo de ello podría ser la implementación de un banner publicitario o la presentación de un gráfico específico.

Sin embargo, el proceso de creación de imágenes puede llevar mucho tiempo y esfuerzo, por lo que disponer de bancos de imágenes e iconos puede ser la mejor forma de aumentar la productividad y un buen recurso a la hora de diseñar nuestras páginas web. En este sentido, dos de los bancos más conocidos y utilizados en diseño web quizás sean PIXABAY y UNSPLASH.

En estos bancos de imágenes podemos encontrar infinidad de imágenes sin derechos de autor que pueden ser adheridas a nuestras páginas.

### URLS de consulta y descarga:

<https://pixabay.com/es/>

<https://unsplash.com/>

En lo referente a iconos, los repositorios más conocidos sonICONFINDER y ICON8. En estos repositorios podemos encontrar iconos gratuitos para uso comercial fácilmente implementables en nuestros diseños. Además, pueden ser descargados tanto en formato PNG, como en formato SVG.

## **URLS de consulta y descarga:**

<https://www.iconfinder.com/>  
<https://icons8.com/>

## **COMPRESORES Y MINIFICADORES**

Como veremos, algo imprescindible a la hora de diseñar de sitios web es la necesidad de servir los contenidos comprimidos y optimizados, tanto a nivel de imágenes, como a nivel de archivos.

Si lo que deseamos es optimizar la compresión de una imagen, lo habitual es recurrir a algún software específico como Photoshop, Lightroom o Gimp. No obstante, si no disponemos de ningún editor de imágenes, se puede recurrir a alguna herramienta online como es TINYPNG o IMAGEOPTIM. Ambas utilidades son una forma rápida de realizar compresiones con pérdida en archivos PNG y JPEG y que pueden disminuir el tamaño de las mismas de forma drástica.

### **URL de acceso y uso:**

<https://tinypng.com/>  
<https://imageoptim.com/online>

Ahora, si lo que deseamos es comprimir o reducir el tamaño de nuestros archivos de texto, la opción es utilizar un minificador. Este tipo de utilidades son muy interesantes y permiten reducir el tamaño de los archivos mediante la eliminación de caracteres innecesarios y, como consecuencia, mejoran el SEO.

Entre los más conocidos podemos encontrar MINIFIER.ORG, un minificador de CSS y JS construido en PHP y fácil de integrar. Entre otras cosas, elimina los espacios en blanco, saltos de línea y signos de puntuación innecesarios, borra los comentarios, combina múltiples archivos y optimiza algunos patrones de programación comunes.

### **URL de ayuda y uso:**

<https://www.minifier.org/>

No obstante, según fuentes oficiales de Google, los más recomendados son HTMLMINIFIER para minificar HTML, CSSNANO o CSSO para minimizar CSS y, UGLIFYJS o CLOSURE COMPILER para minificar JavaScript.

### **URLS de ayuda y uso:**

**# Minificadores HTML**  
<https://github.com/kangax/html-minifier>  
**# Minificadores CSS**  
<https://github.com/cssnano/cssnano>  
<https://github.com/css/csso>  
**# Minificadores JS**  
<https://github.com/mishoo/UglifyJS>  
<https://developers.google.com/closure/compiler?hl=es>

## **COMUNIDADES DE CÓDIGO**

Otro recurso que nos puede venir bien a la hora de crear nuevos desarrollos es la utilización de fragmentos de código ya probados. Esto, y mucho más, es lo que ofrece la web de CODEPEN, una comunidad en línea que permite probar fragmentos de código en JS, CSS y HTML, incluyendo varios motores de plantillas, lenguajes compilados de marcado de hojas de estilo y transcompiladores de JavaScript.

Además, funciona como un editor online de código abierto donde los desarrolladores exponen sus trabajos a modo de presentación y pueden ser compartidos a través de varios métodos.

No obstante, aunque son menos “conocidas” o están menos extendidas para este contexto, existen otras comunidades de códigos en línea como son JSFIDDLE, CODESANDBOX o JSBIN.

### **URL de acceso y uso:**

<https://codepen.io/>  
<https://jsfiddle.net/>  
<https://codesandbox.io/>  
<https://jsbin.com/>

## **MANIPULACIÓN DE COLORES**

Para la manipulación de colores lo que podemos usar es COLORZILLA, una extensión disponible para Chrome y Firefox y descargable e instalable desde sus respectivos portales. Incluye un selector de color con conversión a varios formatos, cuentagotas y un generador de degradados, entre otras herramientas.

### **URLS para descarga de extensiones:**

<https://chrome.google.com/webstore>  
<https://addons.mozilla.org/es/firefox/>

Cabe mencionar que, aunque COLORZILLA incluye un generador de degradados, si no lo tenemos instalado y necesitamos crear un degradado de manera rápida, se puede recurrir a la utilidad online de ULTIMATE CSS GRADIENT GENERATOR, también suministrada y mantenida por Colorzilla.

### **URL de ayuda y uso:**

<https://www.colorzilla.com/gradient-editor/>

## **OTROS RECURSOS DE DISEÑO**

Aunque en este curso no se utilizarán, siempre es bueno tener una referencia a ciertos recursos por si hiciesen falta.

Uno de estos recursos son las plantillas porque, además de ser una buena herramienta para obtener una primera aproximación de manera rápida, pueden ser una buena fuente de inspiración para nuestros diseños.

En este sentido, los repositorios de plantillas más extendidos quizás sean HTML5UP, TEMPLATED, STYLESHOUT, THEMEWAGON y FREEHTML5, los cuales ofrecen plantillas muy funcionales de código abierto, con y sin derechos de autor.

### **URLS de consulta y descarga:**

<https://html5up.net/>  
<https://templated.co/>  
<https://www.styleshout.com/free-templates/>  
<https://freehtml5.co/>  
[https://themewagon.com/theme\\_tag/free/](https://themewagon.com/theme_tag/free/)

Otro de estos recursos son los frameworks de CSS, que ofrecen un set de utilidades que hacen posible que los diseños se realicen mucho más ágilmente. En este sentido, los frameworks de CSS más extendidos quizás sean BOOTSTRAP, MATERIALIZE, PURECSS, BULMA y FOUNDATION.

### **URLS de consulta y descarga:**

<https://getbootstrap.com/>  
<https://materializecss.com/>  
<https://purecss.io/>  
<https://bulma.io/>  
<https://get.foundation/>

## HISTORIA DE LA WEB

En los años 60 se produjo una nueva forma de compartir información con otros usuarios. Se trataba de un servicio de comunicación que sólo permitía la inclusión de textos y eran manipulables a través de navegadores de sólo texto. Sin embargo, no fue hasta principios de los noventa cuando se creó HTML, lo que provocó que la web empezara a tener una aceptación suficiente y extenderse como la pólvora.

En aquel entonces, la web era un sistema unidireccional de publicación estático de sólo texto que no presentaba gráficos o imágenes, no ofrecía opciones de personalización, no permitía la actualización y, mucho menos, la posibilidad de realizar intercambio de datos, por lo que los usuarios no podían interactuar con el contenido y, únicamente, se limitaban a consultar o leer la información que el administrador de la página web hubiese subido a la red. A esto, se denominó la Web 1.0.



La web 2.0, término que fue bautizado por O'Reilly en el año 2004, supuso la segunda generación de sitios y páginas web. Este tipo de webs ya no eran estáticas, ni de sólo lectura y permitían, entre otras cosas, compartir e interactuar con la información de una manera sencilla. Como consecuencia de ello, se produjo un desarrollo de la inteligencia colectiva que fomentaba la colaboración y el intercambio de información a través de comunidades y redes sociales.

Es, por esta época, cuando se crean y extienden sistemas tan conocidos como son los blogs, chats, wikis o foros. Sistemas bidireccionales, los cuales, permitían manipular y gestionar la información de forma sencilla, además de permitir la adición de comentarios y opiniones o interactuar con otros usuarios que presentaban las mismas inquietudes, pero que no requerían tener el mismo nivel técnico o cultural.

Sin embargo, no fue hasta la web 3.0 dónde se produjo un salto cuantioso en lo referente a los sistemas en red. La web 3.0, la tercera generación de sitios y páginas web, ya no sólo era una forma de interactuar y compartir la información de manera sencilla, ahora, su objetivo darle significado y enriquecer la experiencia del usuario.

Es aquí, como alguno ya habrá pensado, cuando nace la Semántica web y las páginas web empiezan a estructurarse a través de un lenguaje natural que puede ser interpretado por el software definiendo qué parte tiene qué función. De esta forma, acceder a la información resulta más sencillo y rápido de procesar, porque hasta las máquinas son capaces de “entender” los contenidos y su objetivo.

Se dice que la web 3.0 también tiene bastante que ver con la inteligencia artificial puesto que las páginas y aplicaciones web ya poseen la capacidad de conectarse entre sí para ofrecer un mejor servicio a los intereses de cada usuario. No obstante, es en la web 4.0 dónde esta premisa está más presente puesto que es quién obtiene un comportamiento más inteligente, predictivo y simple que implica menos movimientos y más acciones con mejores resultados.

Con la web 4.0, nacen el aprendizaje profundo (Deep Learning) y el aprendizaje automático (Machine Learning), tecnologías que forman parte de una familia de métodos de aprendizaje automático basados en redes neuronales con aprendizaje de representación. En otras palabras, tecnologías basadas en sistemas capaces de aprender a realizar tareas tras analizar diferentes patrones y muestras mediante técnicas de aprendizaje que permiten descubrir de manera automática las características de una entidad a partir de datos sin procesar.

El ejemplo más conocido o extendido de todo esto quizás sea Watson de IBM, un software capaz de responder preguntas realizadas en lenguaje natural y de realizar tareas como Speech To Text, el cual permite hablarle a una máquina y convertir lo dicho en texto escrito.

Pero esto no es todo, la web 4.0 es la responsable de que los usuarios sean advertidos por sus dispositivos móviles antes de que ellos mismos se den cuenta. Por ejemplo, ¿quién no ha recibido notificaciones con la ruta más corta al trabajo, avisos por atascos en la carretera, alertas por fuertes tormentas o lluvias o mensajes de advertencia sobre tu elevado ritmo cardíaco?

## CÓMO FUNCIONA LA WEB

De forma básica, cuando un usuario se conecta a Internet con un dispositivo cualquiera, se le asigna un identificador único mediante los protocolos TCP/IP (Protocolo de Control de Transmisión / Protocolo de Internet). El protocolo TCP proporciona el medio para crear las conexiones y el protocolo IP proporciona el mejor “camino” para alcanzar su destino.

Este identificador único, más conocido como dirección IP, suele estar compuesto por cuatro códigos de 8 bits y vinculado a un nombre, también único, el cual utilizamos para acceder a un sitio web (véase, por ejemplo, <https://google.es>).

¿Qué es lo que sucede entre medias? Como hemos dicho, Internet se mueve a través de direcciones IP, por lo que, para conseguir la dirección IP asignada a ese nombre que hemos introducido, primero se debe acceder a un sistema intermedio que almacena dicha relación.

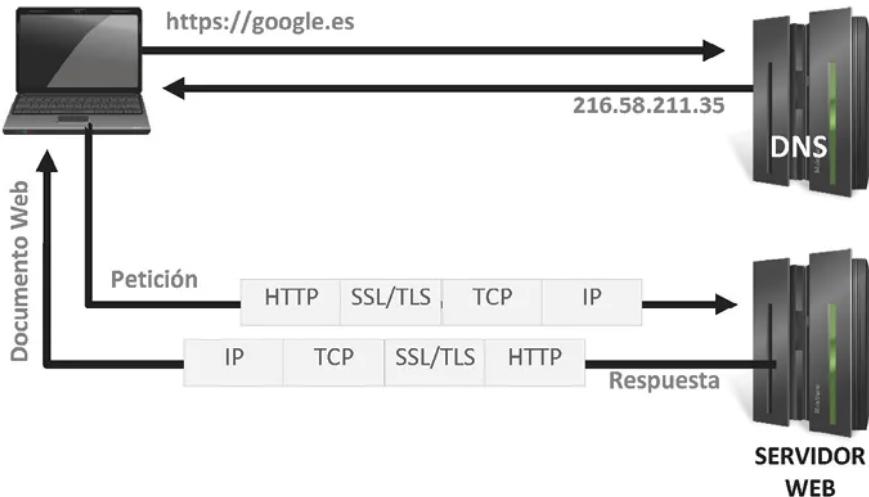
Ese sistema intermedio se conoce como DNS (Sistema de Nombres de Dominio) y, fundamentalmente, lo que hace es recopilar un catálogo de correspondencias de nombres e IPs y devolver un valor concreto como, por ejemplo, 216.58.211.35.

Una vez que se tiene el objetivo al que dirigirse, el navegador, también llamado Cliente en términos de comunicaciones, abre una instancia de comunicación con el Servidor mediante el protocolo HTTP (Protocolo de Transferencia de Hipertexto). Este protocolo es quien dicta las normas para que el Cliente se comunique con el Servidor Web asignado a la IP anteriormente adquirida y es, además, quien define la sintaxis y semántica que se debe utilizar en cada conexión.

No obstante, si accedemos a la consola del navegador (pulsando F12) y recuperamos la información de la pestaña NETWORK, al recargar la página veremos que la mayoría de estas conexiones entre el Cliente y el Servidor se realizan a través de HTTPS, o lo que es lo mismo, la versión segura del protocolo HTTP.

En este tipo de comunicación, el servidor establece un cifrado basado en la seguridad de textos mediante los protocolos criptográficos SSL/TLS, los cuales, permiten crear una capa codificada intermedia entre los protocolos HTTP y TCP/IP por el que envía el código HTML que el navegador muestra al usuario.

A continuación, se muestra un gráfico que representa todo el proceso:



Proceso de una petición web.

## DISEÑO GRÁFICO Y EL PAPEL DEL DISEÑADOR

El diseño gráfico podría definirse como una disciplina que consiste en presentar información visual y cuyo objetivo es que, los usuarios, capten mensajes específicos sobre un tema o materia determinada.

También podríamos definir el diseño gráfico como el arte de crear una composición de imágenes y texto de forma que se transmitan mensajes de forma efectiva.

Cuando hablamos de diseñadores gráficos, no estamos hablando de comerciales ni de artistas, no obstante, es un perfil que tiene algo de ambos. El objetivo de un diseñador gráfico no es vender nada, aunque pueda hacerlo. Ni tampoco es ser un experto comunicador, aunque puede y debe tener una cierta maestranza para poder abstraer los mensajes que se quieren divulgar y transmitirlos de forma eficiente.

Por tanto, lo primero que necesita un diseñador gráfico para empezar a trabajar es un documento o reunión informativa en la que se le proporcione todos los datos necesarios para poder afrontar de manera ordenada, estratégica y creativa el objetivo a cumplir.

Más tarde, cuando ya disponga de los requerimientos y necesidades, y tenga una idea más o menos clara de lo que se debe crear, el diseñador podrá ayudarse, y en general lo hará, de herramientas específicas como son PhotoShop, Illustrator, Gimp o FireWorks, y de lenguajes de marcado como HTML5, CSS3 o SVG, los cuales facilitan el trabajo en gran medida.

## ARQUITECTURA DE LA INFORMACIÓN Y EL PAPEL DEL ARQUITECTO

La Arquitectura de la Información (AI) es el arte, disciplina o metodología, según quien lo defina, encargada de estudiar, analizar, organizar, y estructurar la información.

Cuando se habla de Arquitectura de la Información, a lo que se hace referencia es a unos principios de diseño y construcción que ayudan a seleccionar y presentar la información en sistemas interactivos y pasivos.

En lo referente a lo que busca la Arquitectura de la Información podríamos decir que, su objetivo, es:

- Procesar y dosificar la ingente cantidad de información que se produce cada día en las redes.
- Desarrollar y verificar los procesos de producción de forma que los usuarios puedan acceder con los mínimos problemas posibles.
- Utilizar la información de manera clara y precisa.
- Organizar, estructurar, marcar y distribuir los sistemas de información para que la experiencia de los usuarios se vuelva más sencilla y menos frustrante.

Si pensamos en la Arquitectura de la Información como proceso, lo que encontramos es una especie de ecosistema que tiene un ciclo de vida completo y que va definiendo los objetivos para que la producción y desarrollo se realice de forma efectiva y eficiente.

De hecho, para que los contenidos puedan ser accedidos, utilizados y asimilados de manera eficiente y efectiva, la Arquitectura de la Información define una serie de técnicas que, dependiendo de cuáles sean y de cuándo se produzcan, ayudan al desarrollo y creación de sistemas de información.

Estas técnicas pueden definir:

- El objeto o propósito del producto.
- El público objetivo.
- Los estudios de audiencia y análisis de competencia.
- La planificación, gestión y desarrollo de contenidos.
- El diseño de la interacción, la Usabilidad y la Accesibilidad.
- El diseño de la navegación, organización y apariencia de los contenidos.
- El marcado de los datos para acceder a los contenidos y facilitar su búsqueda.
- Los procesos de reingeniería del sistema.

Por tanto, un arquitecto de la Información es una persona que lleva a cabo la ejecución y verificación del diseño del sistema o interfaz, además de estar en constante cooperación con los diseñadores gráficos y responsables de la parte de procesamiento y lógica de negocio para definirla.

Sin embargo, no me sentiría cómodo al cerrar esta parte, sin antes, realizar unos últimos comentarios. Cuando se habla de Experiencia de Usuario (UX) se puede estar haciendo referencia a la Arquitectura de la Información, porque es una parte o componente de misma. No obstante, mientras que la Arquitectura de la Información se centra en la información, la Experiencia de Usuario se centra en los usuarios y en cómo pueden actuar y pensar.

Piénsese que el concepto de UX es un conglomerado de técnicas y elementos que define o ayuda a definir la estructura y organización de la información, el diseño de interacción, la usabilidad y accesibilidad de los sistemas y que, para ello, se apoya en el diseño gráfico y procesos cognitivos y de percepción.

## **ELEMENTOS DE BÁSICOS DE UNA PÁGINA WEB**

Antes de entrar en materia, veamos qué es una página web. Desde un punto de vista conceptual, una página web se podría definir como un documento que contiene información electrónica que se presenta de forma ordenada, estructurada y/o jerarquizada.

Desde un punto de vista más formal, una página web podría definirse como un conjunto de contenidos textuales y no textuales (como puedan ser sonidos, imágenes, videos, scripts, etcétera) que se presentan de manera secuencial,

ordenada y estructurada, que son accesibles mediante una red informática externa o interna y que pueden ser presentados por un agente de usuario (como pueda ser un navegador, lector de pantalla, magnificador u otra herramienta de asistencia).

Dicho de un modo más sencillo. Si se observa una página web desde un navegador, lo que se percibe es un conjunto de entidades a las que se les dota de un significado determinado. Este conjunto de entidades tiene, además, una serie de características concretas como puedan ser la forma, el color o el tamaño que hacen que los elementos se perciban aisladamente o en conjunción con otros.

Si la entidad es una unidad léxica o palabra, su objetivo suele conllevar una interpretación cognitiva y un proceso de memorización, pero si la entidad es otra cosa como, por ejemplo, una imagen o un video, el objetivo suele ser el impacto sensorial, la comprensión y/o la satisfacción.

No obstante, una página o documento web no sólo es un conjunto de elementos relacionados entre sí, también puede contener personalizaciones, animaciones y transiciones, o llevar asociadas acciones y reacciones como el envío y/o consulta de información o el intercambio de datos.

Esto es posible gracias a que, una página web, no sólo se compone de un etiquetado en HTML, que es el nivel más bajo de marcado, sino que, por lo general, tiene partes definidas en otras tecnologías como puedan ser CSS, SVG o JavaScript.

La combinación de HTML con estas otras tecnologías es la máxima responsable de que las páginas web tengan la interacción que se produce entre el usuario y la máquina y, posiblemente, también es la máxima responsable de que el uso de Internet tenga éxito que tiene.

## **Configuración y estructura básica de una página web**

Todo documento web presenta siempre una misma estructura básica. Una cabecera que prepara el documento, un contenido central o principal que habitualmente se denomina cuerpo del documento y un pie de página que suele proporcionar algunos contenidos adicionales.

La cabecera de la página es la parte en donde se proporcionan todos los datos de configuración del documento y todos los datos que describen el contenido del documento que se va a presentar. Estos datos que describen el contenido del documento se denominan formalmente metadatos y, en pocas palabras, se definen como datos acerca de los datos.

Estos metadatos pueden ser de muy diferente índole, desde datos puramente informativos, como quién creó el documento o el título del mismo, hasta datos que configuran la presentación del contenido. Estos últimos, pueden preparar la codificación de caracteres a usar en el documento, establecer las palabras clave que permiten la indexación en los motores de búsqueda como Google o Yahoo!, permitir o no que la página sea rastreada por los robots de los motores de búsqueda, definir el periodo de validez de los datos en caché, etcétera, etcétera, etcétera.

El cuerpo de la página, por el contrario, no describe nada referente al propio documento, sino que proporciona todo el contenido perceptible, es decir, es la parte del documento donde se establecen los textos, imágenes, videos, gráficos estadísticos e, incluso, los sonidos o música de fondo.



Ejemplo de estructura básica de una página web.

Como se puede observar en la ilustración anterior, cada elemento de una página web suele estar vinculado a una sección determinada. Por ejemplo, el logo y menú principal suelen estar ubicados en la zona de la cabecera y, en el pie de página, se suele mostrar información sobre el copyright.

No obstante, las secciones, casi nunca suelen tener tan pocos elementos o ser tan triviales. De hecho, los pies de página, habitualmente, suelen llevar otros contenidos adicionales como puedan ser los enlaces de interés, los créditos, acceso a las redes sociales y, en algunos casos proporcionan un formulario para apuntarse a la Newsletter o para realizar un contacto.

## **LOS CONTENIDOS**

El contenido web podría definirse como un conjunto de datos expresados en lenguaje visual y que tienen en un formato específico. Así, un contenido web podría ser un contenido textual, que hace referencia a un contenido de sólo texto, o un contenido multimedia, que hace referencia a un contenido visual y/o auditivo.

A menudo, todos los contenidos de una web están ubicados en el cuerpo de la página, sin embargo, no todo lo que conforma el cuerpo de una página es contenido web. De hecho, un enlace no se considera contenido web debido, principalmente, a que representa un vínculo hacia otro contenido.

Evidentemente, esto tiene una excepción, porque, si un enlace forma parte de un listado temático si puede representar un contenido web, debido a que se considera un directorio o catálogo que da una visión global sobre una materia determinada.

Dicho esto, ahora viene la pregunta. ¿Qué tipos de contenidos puede haber?

### **Tipos de contenido**

#### **TEXTUAL**

El contenido textual es aquel en el que requiere de la lectura y escribe sobre una materia concreta, independientemente de la forma que se utilice para contarla. No obstante, dependiendo del tono, objetivo, formato o forma, los contenidos pueden ser interpretados de manera diferente.

Así, por ejemplo, si el tono es más riguroso y está escrito en segunda o tercera persona, podríamos pensar en blogs o artículos de investigación, pero si el tono es más visceral y está escrito en primera persona, podríamos estar hablando de artículos de opinión.

Si pensamos en el objetivo, la cosa también puede cambiar mucho. Por ejemplo, si el contenido explica un tema o materia de forma detallada y específica a través de un archivo PDF, o a través de una presentación interactiva que el

usuario puede ir probando, podríamos estar hablando de una guía o manual de ayuda.

Ahora bien, si el contenido es principalmente visual y está diseñado o construido a través de alguna herramienta como Prezi o InVision, podríamos estar hablando de una presentación para una ponencia o videoconferencia.

## AUDIOVISUAL

El contenido audiovisual es aquel que viene representado por una sucesión de imágenes y/o secuencias de audio susceptibles de ser emitidos y/o transmitidos.

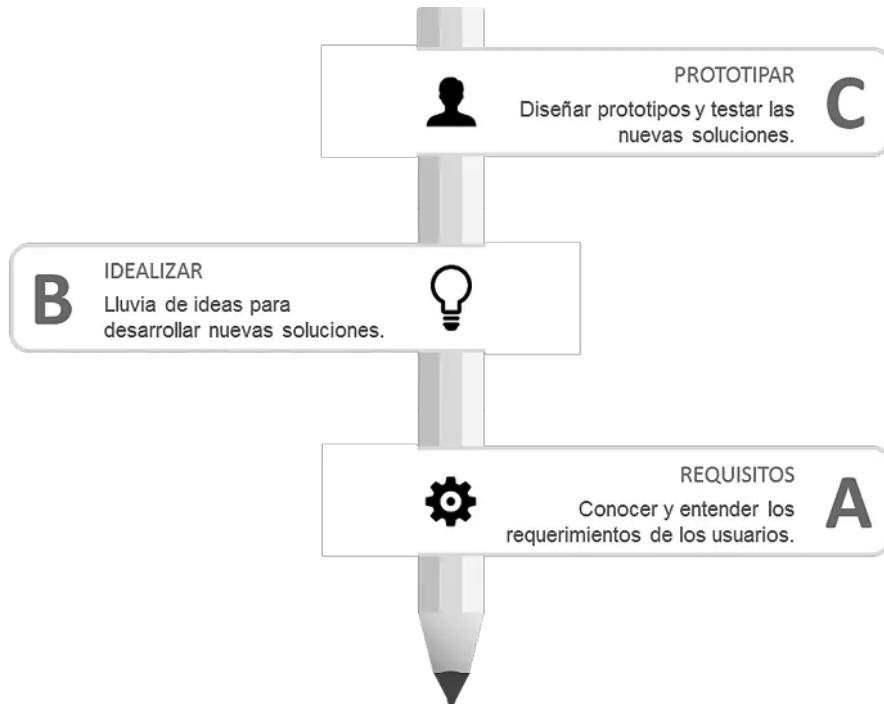
Este tipo de contenido es uno de los preferidos por los usuarios porque es el más fácil de entender, además de ser el más impactante. Además, no suele estar en la misma infraestructura que la página o documento, primero, porque requieren de unas características específicas diferentes al contenido textual y segundo, pueden provocar “cuellos de botella” en la transmisión y ocupar gran espacio en el servidor.

En general, este tipo de contenido viene codificado en formato MP4, OGG o AVI, aunque el más aceptado y extendido es el MP4. La opción de utilizar uno u otro, en general, suele determinarse en función de los navegadores a los que se les quiere dar soporte, como ya veremos.

## GRÁFICO

Una imagen es una representación visual de una figura u objeto real o imaginario. Las imágenes son otro de los contenidos más apreciados por los usuarios porque provocan sensación de moderno y atractivo, sin embargo, hay que utilizarlas con precaución porque pueden provocar problemas de rendimiento, memoria, accesibilidad y usabilidad.

Uno de los contenidos visuales más apreciados, además de las imágenes, son las infografías, las cuales permiten explicar, a través de imágenes, procesos complejos.



Ejemplo de infografía

En esta categoría, podríamos meter las newsletter, que son un tipo de contenido que se suele enviar por correo electrónico y suele estar formado por una estructura tipo tabla que intenta provocar la atención de los usuarios a

través de imágenes y con expresiones o frases cortas. El principal objetivo de estos contenidos es dar acceso a los contenidos o productos que ofrecen las empresas.

También, en esta categoría, podríamos introducir los memes, que son una mezcla de imágenes con expresiones o frases cortas, en las que figura un primer plano de un personaje real o de ficción, y una frase que podría decir el personaje, pero que no tiene por qué ser verídica, cierta o real.

## AUDITIVO

El contenido auditivo podríamos decir que viene representado por una sucesión de sonidos bajo un cierto orden legible.

En esta categoría, podríamos meter los efectos sonoros o la música de fondo, lo cuales suelen estar codificados en formato MP3, aunque no es el único posible.

También, en esta categoría, podríamos meter a los podcasts, que son un formato de audio que representa una emisión de radio o televisión que ha sido digitalizada para que los usuarios puedan descargarla y escucharla cuando deseen. Este tipo de contenido suele estar codificado en formato MP3 o MP4.

Al igual que sucede con el contenido audiovisual, la opción de utilizar uno u otro, en general, suele determinarse en función de los navegadores a los que se les quiere dar soporte, sin embargo, el más recurrente es el MP3.

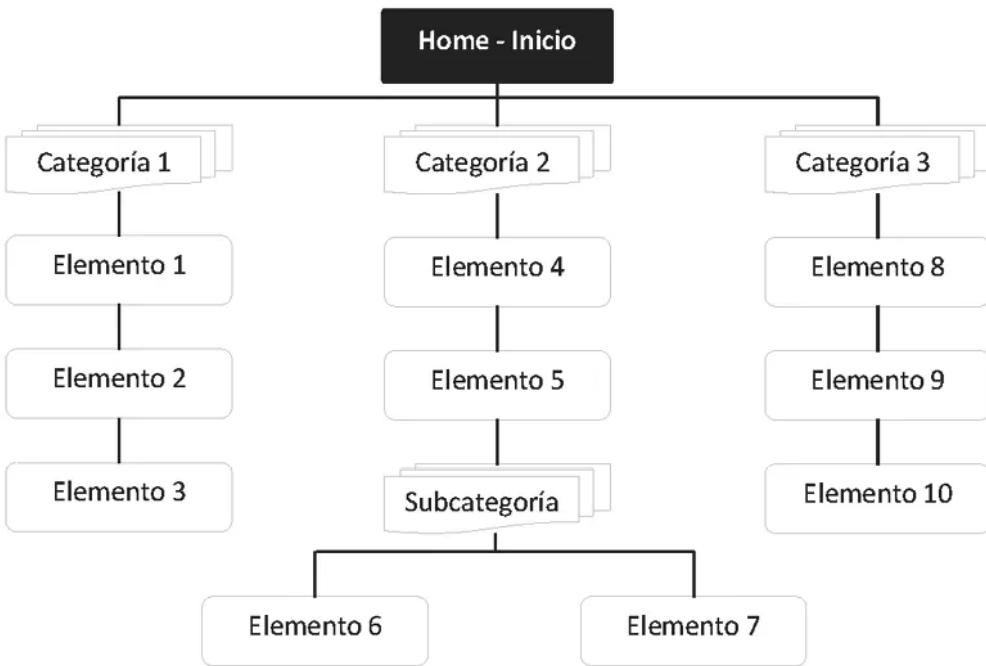
## E-BOOK

Este tipo de contenido suele considerarse especial porque hace referencia a grandes cantidades de información sobre temas específicos descargables y utilizables a través de dispositivos especiales, comúnmente denominados libros electrónicos.

Aunque, estos dispositivos, a menudo, suelen admitir la codificación en PDF, entre otros, las codificaciones más extendidas suelen ser en formato ePUB o Mobi.

## El árbol de contenidos

Un árbol de contenidos se puede definir como un esquema que representa, de forma visual, la composición o configuración integral de un documento, página o sistema de información. En otras palabras, un árbol de contenidos es una estructura jerarquizada que determina cómo se estructura la navegación dentro de un sistema o interfaz.



Ejemplo de árbol de contenidos sencillo.

Como se puede apreciar en la ilustración, a través de la representación visual se puede establecer una jerarquía para cada nodo o elemento que permite pronosticar cómo deberán aparecer los contenidos en el sistema o interfaz y cuantos pasos tendrán que hacerse para llegar al contenido deseado.

No obstante, un árbol de contenidos no sólo es útil para distribuir el contenido y definir las rutas de navegación, también sirve para ver si la estructura de contenidos es o no compleja y si, los usuarios, van a ser capaces o no de llegar a encontrar la información deseada de forma sencilla y natural.

## OPTIMIZACIÓN DE UN ÁRBOL DE CONTENIDOS

Si se analizasen unos cuantos sitios web, se podría comprobar que, en general, la mayoría de ellos suelen responder al nivel de complejidad propuesto en la figura 1.4, es decir, el nivel más profundo de contenidos está a dos o tres niveles por debajo de la página de inicio. Además, el número de categorías iniciales suele establecerse entre cinco y siete elementos, como máximo.

Esto no quiere decir que no se puedan integrar sistemas web con una mayor complejidad de niveles o categorización. Aunque la recomendación es esa, un árbol de contenidos puede requerir más elementos y más categorías. Cuando esto pasa, una de las formas para conseguir simplificarlo es tener claro cuáles son los objetivos del sitio web y, basándose en la competencia, crearlo de tal manera que una persona sin conocimientos adquiridos previos sea capaz de manejarlo.

Otra opción, si la anterior no es posible, es priorizar la información según unos criterios beneficiosos para el negocio bajo una distribución lógica e intuitiva. No obstante, también deberá estar diseñado para conseguir que los usuarios puedan percibirlo y comprenderlo de forma rápida y adecuada, con palabras claras y representativas.

Por ejemplo, imaginemos un árbol de contenidos de un comercio electrónico cualquiera en donde las zapatillas de andar por casa han sido clasificadas dentro de la sección de hogar. Aunque, en un momento dado, podría tener sentido que ese artículo pudiera ser clasificado así, lo más probable es que, cuando los usuarios vayan a buscarlo, lo hagan dentro de la sección de calzado y, de no estar ahí, lo más seguro es que se frustren y se vayan a otro comercio.

## NIVELES Y CAPAS DE NAVEGACIÓN

Los niveles y capas de navegación deben estar diseñados para proporcionar accesibilidad a los contenidos de los sitios web y aumentar el número de conversiones.

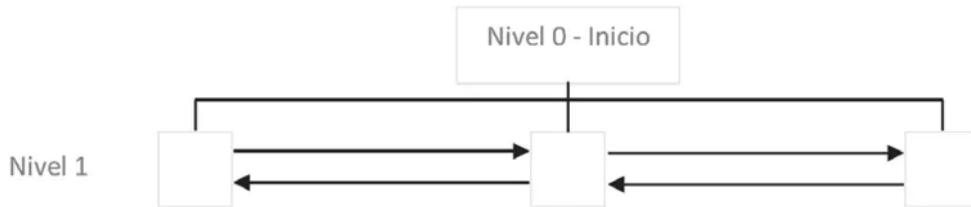
El concepto de conversión es una métrica que suele indicar el porcentaje de usuarios que realiza una acción buscada. Uno de los ejemplos más claros de esto, bien podría ser la reserva de hotel. Si una web que ofrece reservas de hotel tiene 100 visitas al día, pero sólo reservan 18, su número de conversiones (tasa de conversión) será del 18%.

Para intentar conseguir esto se puede recurrir a varias fórmulas o métodos, los cuales se comentan a continuación.

### Estructura lineal

La estructura lineal es aquella que presenta un único nivel de navegación, a partir del cual, los demás contenidos pueden ser accedidos haciendo un único clic en una dirección determinada. En otras palabras, es como cuando estamos haciendo un curso en Internet. Cuando terminamos de leer una sección o capítulo, el sistema siempre nos ofrece la posibilidad de acceder al anterior o al siguiente contenido.

Este tipo de estructura es especialmente útil cuando lo que se desea es que el usuario haga un itinerario o camino específico sin que pueda saltar a otros contenidos, es decir, es útil cuando lo que se desea es diseñar una guía de ayuda o tutorial.

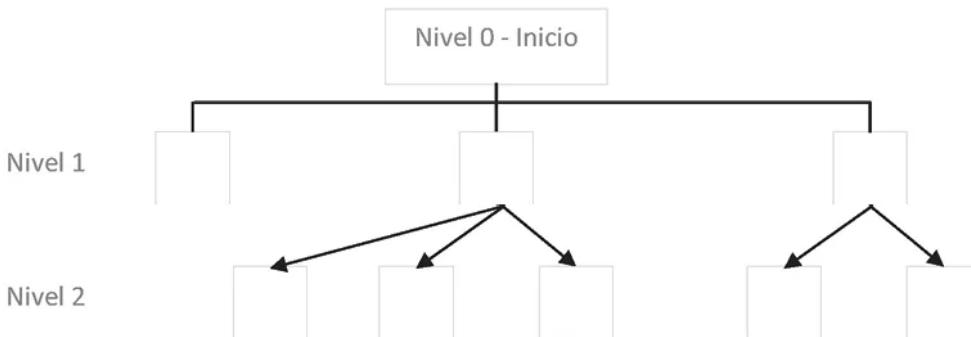


Estructuración lineal.

Cabe destacar que, no es recomendable que el número de contenidos encadenados sea muy elevado porque puede provocar sensación de fatiga y frustración y no permite regresar con facilidad al punto dónde se abandonó la última sesión. Además, suele producir poca relevancia en los contenidos porque los motores de búsqueda como Google, dan mayor importancia a la página de inicio.

### Estructura jerárquica

La estructura jerárquica define un nivel de navegación que representa los grandes grupos o categorías del negocio y, a partir de ahí, la navegación se realiza en cascada. Es decir, un nodo puede representar un contenido final o una categoría que lleva a una subcategoría que, a su vez, puede llevar a un contenido o subcategoría, y así sucesivamente.



Estructuración jerárquica.

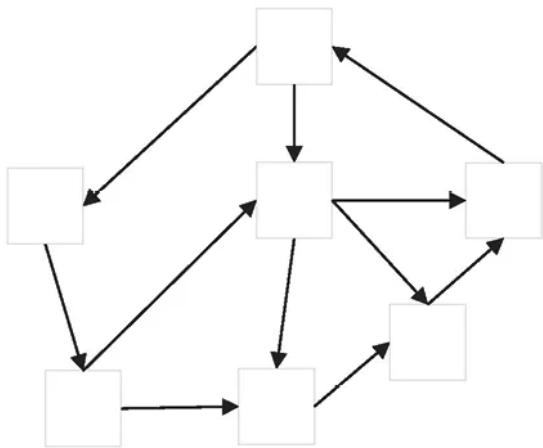
Este tipo de estructura es especialmente útil cuando lo que se desea es organizar contenidos que, desde un punto de vista estricto, no tienen nada que ver unos con otros. Es decir, la sección de servicios no tiene nada que ver el con blog o con la sección contactar. Por ello, es la más utilizada por las empresas y organizaciones, porque facilita la organización de grandes cantidades de información y ayuda a estructurar de forma lógica, es decir, de lo general a lo específico.

Cabe destacar que, esta estructura es una buena opción, siempre y cuando, no se tengan más de 3 o 4 niveles ya que, si tienen más niveles los usuarios pueden frustrarse y abandonar la página por tardar en encontrar lo que buscan o por no comprender bien la navegación.

Sin embargo, la relevancia de los contenidos se genera según su nivel de jerarquía porque los motores de búsqueda como Google, indexan los contenidos en función del nivel de profundidad con respecto a la página de inicio.

### Estructura en red

La estructura en red define un nivel de navegación que permite acceder a los contenidos de forma libre y flexible, es decir, no existen niveles como tal y desde cualquier contenido se puede ir a otros contenidos que llevan a unos contenidos diferentes.



Estructuración en red.

Este tipo de estructura puede llegar a ser útil cuando el contenido no es muy grande, fundamentalmente, porque la relación existente entre sus elementos puede generar confusión y desorientación a los usuarios. Esto es porque, al no haber ninguna jerarquía establecida, los usuarios pueden no saber de dónde han venido y puede que, ni siquiera, a dónde van, por lo que se frustran y abandonan el sitio.

No obstante, como las páginas se relacionan entre sí mediante enlaces internos, se favorece la indexación de las páginas del sitio y eso puede ayudar a que la indexación de los motores de búsqueda como Google posicione mejor los contenidos.

## LA INTERACCIÓN

Cuando hablamos de interacción, en general nos referimos a una relación, influencia o acción - reacción entre dos seres humanos, animales o cosas, pero, en el contexto online o web, la interacción se refiere únicamente a una acción - reacción que se produce entre una persona y un sistema o aplicación web.

Dicha interacción, puede producirse de varias maneras y, habitualmente, puede clasificarse en tres grandes grupos o categorías.

## Mecanismos lingüísticos

Los mecanismos lingüísticos son aquellos en los que el intercambio de información se produce a través de dispositivos de tipo palanca o interruptor, como pueda ser un teclado.

Dónde más se suele ver este tipo de interacción es cuando se trabaja con procesos que requieren de un dispositivo como el teclado y/o el ratón. Por ejemplo, es habitual utilizar este tipo de interacción cuando se está desarrollando un sistema o interfaz, cuando se manipula un control de versiones para llevar un histórico de los cambios o cuando se pulsa una combinación de teclas en una página web determinada que hace que se ejecute una acción asociada.



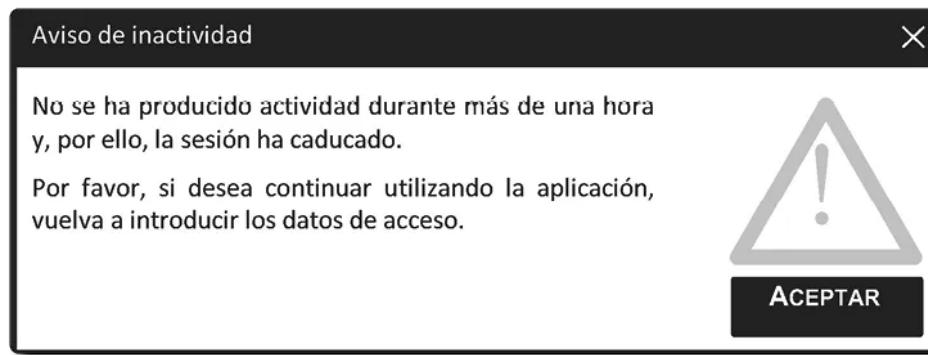
Símbolo de sistema de Windows

## Mecanismos contextuales

Los mecanismos contextuales son aquellos en los que el intercambio de información se produce alrededor de un hecho, evento o situación. Este hecho, evento o situación, puede producirse por una única razón, no obstante, lo normal es que se produzca por conjunto de circunstancias que les rodean o condicionan.

### DIÁLOGOS EMERGENTES

Uno de los mecanismos más recurrentes hoy día son los diálogos emergentes o ventanas modales, los cuales, pueden tener varias funcionalidades. Así, por ejemplo, no es lo mismo un diálogo emergente que solicita la intervención del usuario que un diálogo emergente que informa de una situación problemática, sea o no error.



Ejemplo de diálogo o popup

En este contexto, también hay que matizar dos situaciones que pueden darse, cuando el diálogo modal interrumpe la ejecución del sistema o interfaz y cuando no la interrumpe. En el contexto web, lo normal es que todos los diálogos emergentes obliguen al usuario a realizar una acción, la cual, si no se ejecuta, no permita continuar, pero habrá casos en que no.

Por ejemplo, cuando una página web nos realiza una pregunta a través de un cuadro de diálogo emergente o se nos muestra una alerta típica de JavaScript, la ejecución de todo lo que esté debajo de dicha acción quedará a la espera de confirmación o cancelación.

Los diálogos emergentes pueden ser una buena opción cuando se trata de ofrecer distintos grados de experiencia, debido a que son autoexplicativas y suelen ser fáciles de comprender, pero el uso abusivo de las mismas puede desoriar a los usuarios. Además, dependiendo de cómo se gestionen, pueden sobrecargar las páginas y empeorar el rendimiento global del sistema.

## FORMULARIOS

Otro de los mecanismos que más se suelen ver en sistemas y aplicaciones web son los formularios. Este tipo de interacción permite la introducción de datos de manera estructurada.

Los formularios son un mecanismo de interacción que solicita la intervención del usuario para recuperar algún tipo de información. Entre los diferentes elementos que puede componer un formulario tenemos cajas de texto, casillas de verificación, botones de radio, desplegables y botones de acción.

NOMBRE <b>Pedro Fernández Rodríguez</b>	FUNCIÓN	Profesión o fun-
NÚMERO DE TELÉFONO 999-999-999	EMAIL Introduzca su email de contacto	
EXPERIENCIA Y HABILIDADES Principales experiencias, habilidades y conocimientos		
ESTÁ ACTUALMENTE TRABAJANDO <input checked="" type="checkbox"/> SÍ	DISPONIBILIDAD EN DÍAS <b>15</b>	<b>ENVIAR SOLICITUD</b>

Ejemplo de formulario

Los formularios pueden ser uno de los recursos más utilizados puesto que se pueden utilizar en multitud de situaciones, desde la realización de encuestas, hasta la creación y edición de productos y servicios, sin embargo, antes de crear un formulario, es importante tener en cuenta las restricciones que se puedan dar para no incumplir el RGPD (Reglamento General de Protección de Datos).

## MENÚS

Otro mecanismo de interacción contextual que no suele faltar, son los menús de navegación. Este tipo de interacción permite presentar una lista de opciones seleccionables que el usuario puede ejecutar.

Dependiendo de cuál sea su función, los menús podrán ser de tipo comando, atributo o estado.

- Los de **tipo comando** son aquellos en los que se realiza una acción específica, como pueda ser copiar, cortar o pegar.
- Los de **tipo atributo** son aquellos en los que se manipula una característica sobre uno o varios elementos, como pueda ser cambiar el tipo de letra o el color del texto.
- Los de **tipo estado** son aquellos en los que se habilita o deshabilita una opción de configuración, como pueda ser habilitar un documento para sólo lectura o desactivar un botón de “enviar” hasta que no se hayan rellenado

todos los campos requeridos.



#### Ejemplo Menú de navegación

Cabe destacar que, un desplegable de un formulario puede ser considerado un menú, puesto que responde a la definición de lista con opciones seleccionable, no obstante, en sistemas de información web, lo habitual es diferenciarlos en función de si llevan o no una acción asociada.



#### Ejemplo Menú desplegable

### Mecanismos por manipulación directa

Los mecanismos por manipulación directa son aquellos en los que el intercambio de información se produce cuando se tienen que realizar o ejecutar tareas complejas, pero que no requieren una inversión de tiempo elevada en lo que aprendizaje se refiere.

La manipulación directa se basa en el reconocimiento visual y a la simplicidad por lo que, a menudo, los usuarios recuerdan mejor los conceptos principales sin tener que recurrir a la memorización de opciones y/o conceptos.

Además, la manipulación directa tiene una cosa bastante buena y es que, en general, es reversible. Es decir, si el usuario comete un error, siempre dispone de algún método para poder subsanarlo, lo que aumenta la seguridad y disminuye la ansiedad y frustración.



### Ejemplo manipulación directa

Si nos fijamos en la ilustración anterior, uno de los tipos de manipulación directa más extendidos y frecuentemente utilizados es el ícono. Sólo debemos fijar la mirada en la parte derecha de cualquiera de los recuadros y ver que hay uno o dos triángulos.

Cada una de estas figuras representa un ícono, entendiendo que, un ícono, es una entidad visual que representa un concepto. Por ejemplo, en el recuadro del margen de arriba, los triángulos nos indican que puede aumentarse o disminuirse su valor.

## Mecanismos de interacción social

Los mecanismos de interacción social son aquellos en los que el intercambio de información se produce entre dos o más personas bajo un mismo contexto o sistema.

La interacción social se basa en el intercambio de mensajes y opiniones sobre un tema o materia en particular y, habitualmente se utiliza para conseguir una mayor cercanía entre los usuarios y las empresas y organizaciones, aunque, también se suele utilizar para crear audiencia, notoriedad e influencia sobre productos y servicios.

Por su definición, este tipo de interacción se da de manera más notoria en las redes sociales a través de comunicaciones directas que incluyen el identificador de usuario, pero también suelen darse muchas comunicaciones indirectas que incluyen un identificador de hashtag.

## LAS BASES DE DATOS

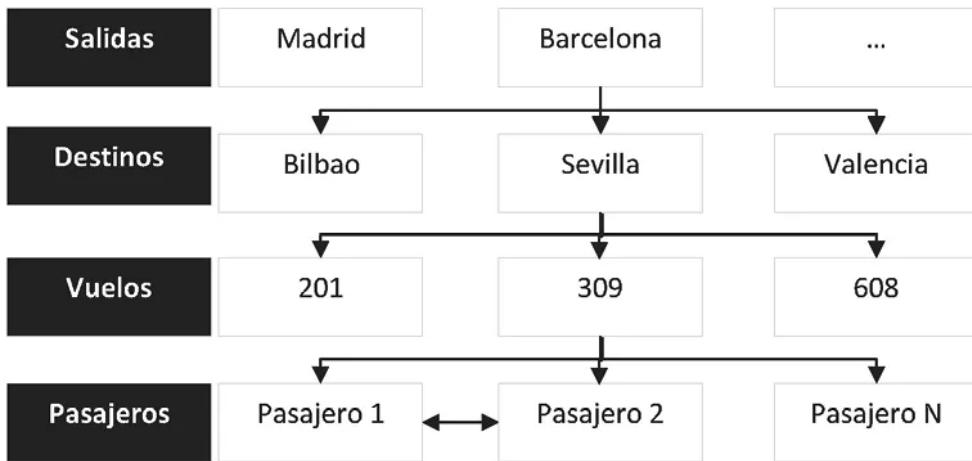
Una base de datos es una herramienta que permite la manipulación, organización y administración de información relacionada y estructurada de forma sencilla y sistemática a través de una interfaz gráfica o de línea de comandos.

A los programas o aplicaciones que gestionan estas herramientas se las suele denominar Sistemas Gestores de Bases de Datos (SGBD) y son consideradas una de las partes fundamentales dentro de un sistema o aplicación web.

Las bases de datos pueden ser de múltiples tipos, aunque las más extendidas son las que responden al modelo relacional y las que responden al modelo documental.

## Bases de datos jerárquicas

Una base de datos jerárquica es un modelo de gestión de bases de datos en dónde la información se estructura y almacena en forma de árbol.



Ejemplo de base de datos jerárquica

Se denominan jerárquicas porque poseen un nodo raíz que no participa como registro hijo y porque todos los nodos, a excepción del nodo raíz, admiten un único parente.

No obstante, la característica de jerarquía no tiene por qué estar de manera obligatoria, es decir, puede suceder que se establezca una relación entre elementos del mismo nivel, como es el caso entre los elementos hoja Pasajero 1 y Pasajero 2.

Entre las principales limitaciones que presenta este modelo podemos encontrar que:

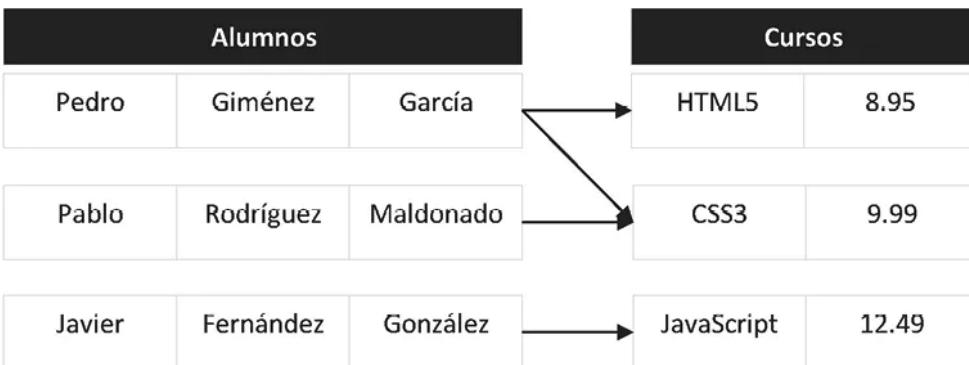
- No se garantiza la unicidad de datos porque pueden aparecer registros duplicados en el mismo nivel.
- No se garantiza la integridad referencial porque permite realizar dejar elementos inconexos o ejecutar acciones como borrar un elemento que contiene otros elementos.
- No se garantiza la normalización porque, por ejemplo, no existen los campos de identificador único y se permiten la combinación de relaciones o la creación de grupos repetidos.

Como nota adicional, sólo diremos que este modelo puede ser considerado como un caso especial o particular de base de datos en red.

## Base de datos de red

Una base de datos en red es un modelo de gestión de bases de datos en dónde la información está formada por un conjunto de registros que se conectan entre sí a través de vínculos de red.

Se denominan en red porque cada registro propietario (nodo parente), tiene uno o más registros miembros (nodos hijo). Además, pueden existir conjuntos singulares en los que, el propietario, puede ser el sistema.



### Ejemplo de base de datos en red

En lo referente a su estructura, una base de datos en red es bastante más compleja que la jerárquica, entre otras cosas, porque cualquier registro, sea propietario o miembro, puede pertenecer a uno o varios conjuntos y porque permite la declaración de un registro como propietario en un conjunto, pero como miembro en otro.

Cabe destacar que, los registros de este modelo son objetos similares a los utilizados o descritos en el modelo relacional, pero con la diferencia de que únicamente pueden almacenar un dato. No obstante, un registro miembro puede tener más de un registro propietario, a diferencia del modelo jerárquico que sólo admite un nodo hijo por padre.

Entre las principales limitaciones que presenta este modelo podríamos destacar las mismas que en el modelo jerárquico, a excepción de que, la adquisición de una buena seguridad de la información es más compleja.

## Bases de datos transaccionales

Una base de datos transaccional es un modelo de gestión de bases de datos en donde la información se recolecta, almacena y manipula a través de transacciones, es decir, mediante eventos que crean o actualizan información a partir de datos que se encuentran almacenados en otros sistemas de información.



### Ejemplo de base de datos transaccional

Las bases de datos transaccionales están pensadas para el almacenamiento de grandes cantidades de información y son, a menudo, utilizadas por empresas y organizaciones que pertenecen a las áreas de ventas, finanzas, marketing o recursos humanos. La razón es, esencialmente, que requieren una fuerte consistencia entre sistemas de información diferentes.

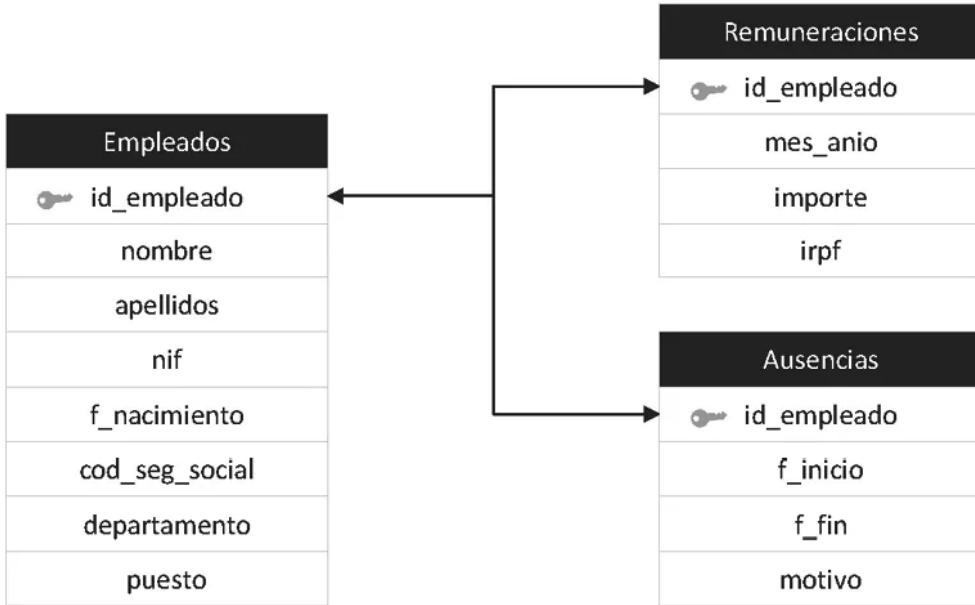
Pensemos, por ejemplo, en las transferencias bancarias. Cuando se realiza una transferencia bancaria de una cuenta a otra, independientemente de si es o no el mismo banco, la cantidad a restar de la cuenta emisora debe ser exactamente la misma que la cantidad a sumar en la cuenta receptora, de lo contrario se produciría una incoherencia que podría acabar en problemas mayores.

Entre las principales limitaciones que presenta este modelo podríamos destacar que presenta poca flexibilidad en las consultas, requieren un tiempo de respuesta entre peticiones muy pequeño (cuanto más pequeño mejor) y que requiere una alta fiabilidad de la información, con posibilidad de recuperación y respaldo de manera rápida y fácil.

## Bases de datos relacionales

Una base de datos relacional es un modelo de gestión de bases de datos en donde la información presenta relaciones predefinidas y se organiza como un conjunto de filas con una o varias columnas.

Las bases de datos relacionales se denominan así porque cumplen el modelo relacional, el cual se basa en la idea de que cada relación (o tabla) está compuesta por unos registros o filas denominadas tuplas.



### Ejemplo de base de datos relacional

Entre sus principales características, podemos destacar que poseen un modelo de datos único, que obliga a que la relación entre tablas se lleve a cabo a través de unos identificadores únicos bidireccionales (clave primaria y clave ajena) y que garantiza que, cada relación o tabla, se realice de forma única, es decir, no permite la existencia de dos tablas con el mismo nombre.

Si el identificador único se declara como clave primaria, actuará como clave principal de forma que ese valor no pueda repetirse. Además, los registros serán indexados a través de estos valores.

Si el identificador único se declara como clave foránea o ajena, deberá tener el mismo valor que su correspondiente clave primaria y actuará como clave de indexación de forma que ese valor quede vinculado a la clave primaria, independientemente de su relación (1:1 o 1:N).

Aunque las relaciones se establecen a través del par clave primaria – foránea, en general, todas las bases de datos relacionales permiten indexación con valores repetidos a partir de uno o varios campos. No obstante, este tipo de índices no es posible utilizarlos para establecer relaciones.

Además, el modelo de base de datos relacional utiliza el lenguaje de comunicación estándar SQL y garantiza el principio designado como ACID:

- **Atomicidad:** Es la propiedad que asegura si se ha realizado la transacción o no. Es decir, garantiza si todas las partes intermedias que conforman la transacción se realizaron con éxito o no.

- **Consistencia:** Es la propiedad que asegura que los datos involucrados en las transacciones sólo cambien la información afectada cuando se cumplan las reglas y directrices de integridad referencial. De este modo, se podrá garantizar que la información que se devuelve siempre sea la misma.
- **Aislamiento:** Es la propiedad que asegura que cada operación pueda ejecutarse de forma independiente sin generar errores ni afectar a otras operaciones anteriores o posteriores.
- **Durabilidad:** Es la propiedad que asegura que, una vez hechas las operaciones, los datos almacenados persistirán en el tiempo, aunque el sistema falle.

Sin ningún lugar a dudas, este es el modelo de base de datos más utilizado y extendido en Internet para gestionar y administrar datos. Ciento es que el modelo de base de datos documental está tomando auge, no obstante, nunca podrá superar ni reemplazar al modelo de base de datos relacional.

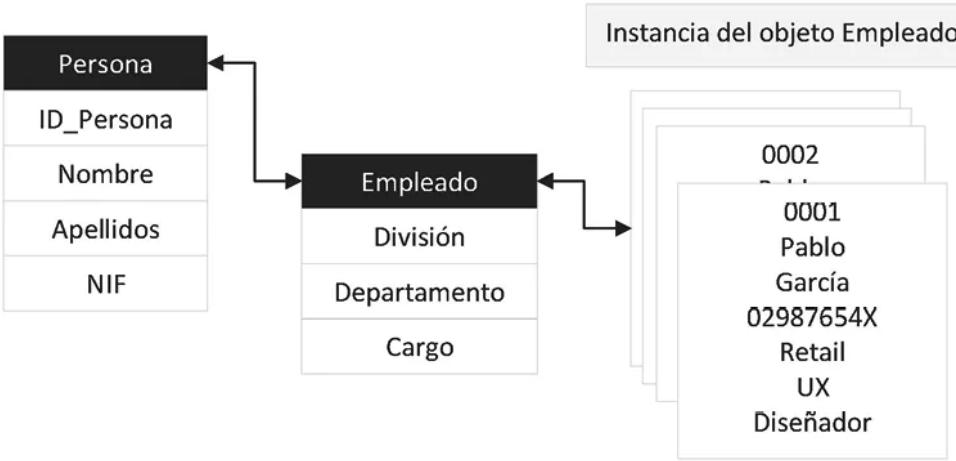
Entre los sistemas gestores de bases de datos que se basan en el modelo relacional, los más conocidos son MySQL, Oracle, PostgreSQL y SQLServer.

La utilización de uno u otro SGBD suele depender de la arquitectura y tecnologías que se deseen utilizar.

- **MySQL:** Está pensada para trabajar con, casi, cualquier arquitectura, aunque su mayor éxito se da en arquitecturas PHP y Python bajo sistemas Linux o Windows. Es, quizás, el sistema gestor de base de datos más extendido, primero, porque permite ahorrar costes iniciales, segundo, porque es compatible con todos los servicios de hosting, sean o no compartidos y, tercero, porque es uno de los lenguajes más sencillos de aprender.
- **Oracle:** Está pensada para trabajar con, casi, cualquier arquitectura, aunque su mayor éxito se da en arquitecturas Java y PHP bajo sistemas Linux o Windows.
- **PostgreSQL:** Está pensada para trabajar con arquitecturas que permiten su integración a modo de objeto, es decir, permite que los elementos puedan tratarse de forma similar a como lo hace un lenguaje de programación orientado a objetos.
- **SQLServer:** Está pensada para trabajar con, casi, cualquier arquitectura, aunque su mayor éxito se da en arquitecturas Java y .NET bajo sistemas Linux o Windows.

## Bases de datos orientadas a objetos

Una base de datos orientada a objetos es un modelo de gestión de bases de datos en donde la información es representada a través de objetos similares a los que se utilizan en los lenguajes de Programación Orientada a Objetos (POO).



### Ejemplo de base de datos orientada a objetos

Mientras que el modelo de base de datos relacional requiere de un lenguaje SQL externo para poder acceder a la información, el modelo de base de datos orientado a objetos permite integrarse con los lenguajes de programación de su misma tipología o paradigma, obteniendo así un Sistema Gestor de Base de Datos orientado a Objetos (SGBDO o ODBMS en inglés), el cual permite que los objetos de base de datos puedan ser manipulados de forma transparente y como si de un objeto del propio lenguaje se tratase.

Entre las principales características que posee, podemos destacar que:

- Permiten la encapsulación, es decir, la ocultación de información al resto de entidades u objetos.
- Permiten el polimorfismo, es decir, que una propiedad pueda aplicarse a objetos de distinta tipología, que presentan una fuerte herencia, lo que permite que se hereden comportamientos.
- Permiten su extensión o ampliación de forma sencilla.
- Permiten la declaración de operaciones sobre los datos como parte de la definición de la base de datos.

Entre los sistemas gestores de bases de datos que se basan en el modelo orientado a objetos, los más conocidos son Object Database++, ObjectStore, GemStone/S, Wakanda y ObjectDB.

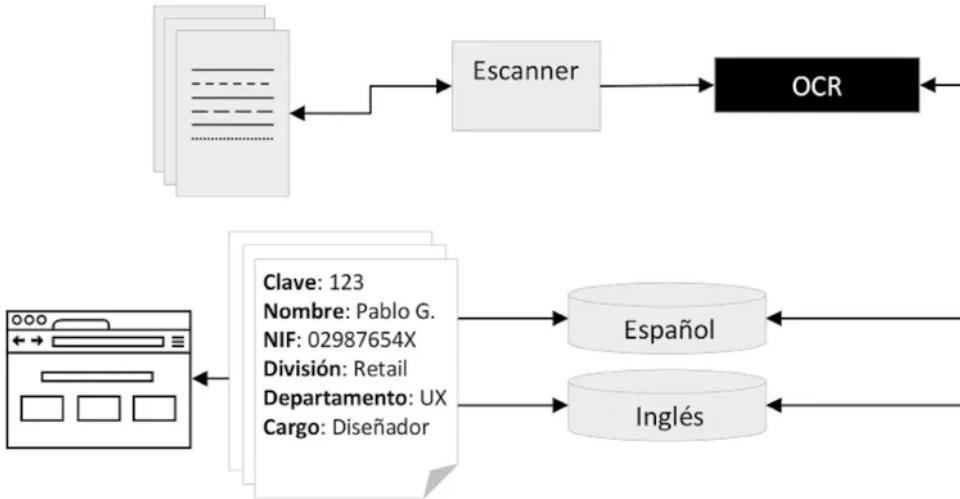
La utilización de uno u otro SGBD suele depender del objetivo y tecnologías utilizadas.

- **Object Database++**: Está más pensada para el alojamiento de aplicaciones en servidores remotos y suele utilizarse cuando los sistemas requieren muchas búsquedas con un fuerte control sobre las mismas.
- **ObjectStore**: Está más pensada para sistemas que requieren almacenamiento en caché y suele utilizarse cuando se requiere una migración y sincronización de datos entre sistemas diferentes.
- **GemStone/S**: Está más pensada para sistemas que requieren alta disponibilidad y suele utilizarse cuando se requiere una integración de aplicaciones de diferente índole de forma escalable y segura.
- **Wakanda**: Está más pensada para el desarrollo web y móvil con aplicaciones en JavaScript y suele utilizarse cuando se requiere un control más exhaustivo sobre el código.
- **ObjectDB**: Está más pensada para sistemas que trabajan en Java y suele utilizarse cuando se trabaja con máquinas virtuales y se requiere el uso de JPA o JDO.

## Bases de datos documentales

Una base de datos documental es un modelo de gestión de bases de datos en donde la información está formada con un conjunto de programas o aplicaciones que almacenan, recuperan y manipulan datos, de algún modo, estructurados.

Las bases de datos documentales son una de las principales tipologías que pertenecen al grupo de bases de datos NoSQL y se denominan así porque están diseñadas alrededor de lo que se define como noción abstracta de “Documento”, aunque también es porque esa información estructurada de algún modo, a menudo, se refiere a datos de documentos.



Ejemplo de base de datos documental

Como se puede observar en la ilustración anterior, los registros de datos dentro del documento son similares a las tuplas o registros utilizados en el modelo relacional, no obstante, son menos restrictivos porque no requieren de un formato estándar, ni que tengan los mismos atributos o campos.

Por ejemplo, mientras que un documento puede tener una lista sobre los hobbies y aficiones de un empleado, otro puede no disponer de esa información (ni siquiera los atributos) y contener una lista de sus experiencias laborales.

Entre los sistemas gestores de bases de datos que se basan en el modelo documental, los más conocidos son MongoDB, DynamoDB, CouchBase, RavenDB, Cassandra e IndexedDB.

La utilización de uno u otro SGBD suele depender del objetivo y tecnologías utilizadas.

- **MongoDB:** Es un SGBD estructurado bajo el esquema NoSQL que utiliza un formato de almacenamiento basado en BSON (similar al JSON). Posee un alto rendimiento en la realización de consultas y actualizaciones y suele utilizarse cuando se requiere un registro de datos diversos con altos volúmenes de información.
- **DynamoDB:** Es una base de datos NoSQL creada por Amazon que utiliza un modelo de datos basado en clave-valor. Cumple el principio ACID y puede llegar a manejar hasta 20 millones de solicitudes por segundo y suele utilizarse cuando las aplicaciones no disponen de servidor propio.
- **CouchBase:** Es una base de datos NoSQL que utiliza un formato de almacenamiento basado en JSON organizado a modo de pares de clave-valor. Posee una baja latencia con un rendimiento sostenido y suele utilizarse cuando se requiere una buena accesibilidad y compatibilidad entre dispositivos diferentes.
- **RavenDB:** Es una base de datos NoSQL que utiliza un formato de almacenamiento basado en JSON y permite la actualización a través de colas LINQ y métodos API. Está escrita en .NET, por lo que suele utilizarse en estos

entornos ya que facilita su manipulación.

- **Cassandra:** Es una base de datos NoSQL distribuida que utiliza el modelo de datos basado en clave-valor. Está escrita en Java y permite la adición y eliminación sin que afecte a la consulta o actualización y suele utilizarse cuando se requieren grandes volúmenes de datos en forma distribuida.
- **IndexedDB:** Es una base de datos NoSQL que utiliza un formato de almacenamiento basado en JSON. Posee una API de JavaScript y suele utilizarse cuando se requiere un acceso a datos en modo de fuera de línea (offline) con almacenamiento en caché.

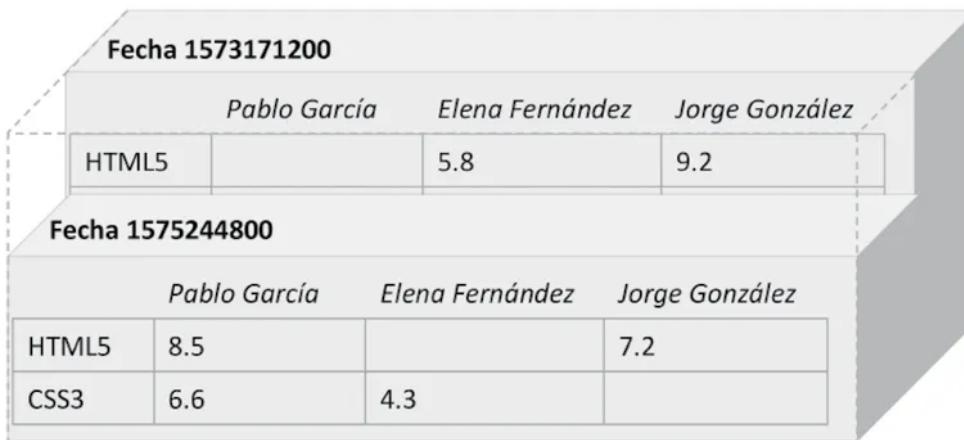
Cabe destacar que, aunque las bases de datos documentales pueden ser muy diferentes en lo referente a los detalles, en general, todas suelen cumplir con el principio de encapsulación de documentos y con alguna codificación estándar JSON o BSON, no obstante, también pueden llegar a utilizar los estándares de XML o YAML.

## Bases de datos multidimensionales

Una base de datos multidimensional es un modelo de gestión de bases de datos que está pensado para data warehouse y aplicaciones de procesamiento analítico en línea (OLAP). Aunque se suelen crear a partir de una base de datos relacional, la forma de gestionar y almacenar la información es muy distinta.

Mientras que las bases de datos relacionales almacenan los datos en objetos tipo tabla con filas y columnas, las bases de datos multidimensionales almacenan la información en un vector multidimensional donde, cada dimensión, representa un conjunto o rango de valores determinado.

ID	Curso	Alumno	Fecha Inicio	Nota
1	HTML5	Pablo García	1575244800	8.5
2	CSS3	Pablo García	1575244800	6.6
3	HTML5	Jorge González	1575244800	7.2
4	CSS3	Elena Fernández	1575244800	4.3
5	CSS3	Elena Fernández	1573171200	8.7
6	HTML5	Jorge González	1573171200	9.2
7	HTML5	Elena Fernández	1573171200	5.8



Ejemplo de base de datos multidimensional

Las bases de datos multidimensionales se basan en la idea de que los datos se pueden representar en tres dimensiones, como si de un cubo se tratase. Cada cubo representa una matriz que contiene un conjunto de dimensiones (categorías descriptivas) y un conjunto de medidas (valores cuantitativos) que coexisten con las dimensiones de análisis y proporcionan un contexto y descripción a través de atributos. Al conjunto de todas esas dimensiones es lo que se denomina jerarquías.

Dependiendo del número de dimensiones de este vector, el nombre puede diferir. Así, si el número de dimensiones del vector es de tres, se le suele denominar cubo, mientras que, si es igual o superior a cuatro, se le suele denominar hipercubo.

La forma de almacenar los datos es como si fuese una matriz en dónde las celdas son las medidas y las posiciones de cada celda son las jerarquías. Por decirlo de algún modo, cada celda de esta matriz representa un qué, y el valor de las dimensiones representa un cómo y cuándo.

La principal razón para utilizar una base de datos multidimensional es porque permiten analizar gran cantidad de datos muy diferentes con una gran agilidad y velocidad por lo que, el tiempo y los recursos empleados durante el análisis se reduce de manera bastante notoria.

Entre los sistemas gestores de bases de datos multidimensionales más populares podemos encontrar Microsoft Analysis Services, Hyperion Essbase y Cognos Power Cube. La utilización de uno u otro SGBD suele depender del gusto y criterio de cada uno.

## **Bases de datos deductivas**

Una base de datos deductiva, también conocida como base de datos lógica, es un modelo de gestión de bases de datos en dónde la información es representada a través de hechos, los cuales pueden ser deducidos o inferidos a partir de unas reglas.

Por decirlo así, los hechos son las tuplas o registros de relaciones que representan el conocimiento explícito y, las reglas, son los criterios deductivos que representan el conocimiento implícito. Tanto los hechos, como las reglas, están explícitamente almacenados en la base de datos.

Las bases de datos deductivas se denominan así porque funcionan en base a una lógica matemática preprogramada. Es decir, la base de datos representa un conjunto de fórmulas lógicas que permiten definir otras nuevas.

No obstante, no dejan de ser un sistema de base de datos relacional que está supervisado por un sistema de inferencia. El usuario manipula este sistema dual y recupera la información de la base de datos que contiene los hechos y reglas.

El lenguaje de definición que utiliza este sistema de inferencia suele ser Datalog (un subconjunto de Prolog) y, entre sus cualidades, podemos destacar que permite realizar consultas recursivas, utilizar algoritmos de evaluación eficientes y soporta objetos y conjuntos complejos.

## **PRINCIPALES TIPOLOGÍAS DE WEB**

A continuación, se muestran las principales topologías web que, actualmente, se encuentran en el mercado.

## **Según su diseño o estructura**

### **PÁGINAS WEB ESTÁTICAS O INFORMATIVAS**

Las páginas web estáticas, también conocidas como informativas, son aquellas en las que el usuario no puede interactuar con ellas, es decir, lo más que pueden hacer es acceder y leer el contenido propuesto. No utilizan sistemas gestores de bases de datos de ningún tipo, por lo que si hay que actualizar el contenido hay que cambiar los archivos desde el servidor de forma directa.

Este tipo de páginas suelen ser recomendables cuando sólo se desea tener presencia en Internet, rara vez se van a realizar cambios en el contenido y se dispone de pocos contenidos.

### **PÁGINAS WEB DINÁMICAS**

Las páginas web dinámicas son aquellas en las que el usuario puede interactuar con ellas de alguna forma, es decir, son webs que disponen de aplicaciones internas que permiten hacer diferentes funcionalidades. En términos generales, suelen requerir algún lenguaje de programación web (entre los más conocidos están Angular, Python, PHP, .Net y JSP) y mucho de su contenido se adquiere a través de sistemas gestores de bases de datos o a través de APIs externas.

Este tipo de páginas suelen ser recomendables cuando:

- Se espera tener una gran cantidad de contenido que puede o debe ser actualizado con cierta frecuencia de forma sencilla y rápida.
- Se desea que los usuarios puedan alterar el diseño o los contenidos de la web.
- Se desean implementar funcionalidades complejas como un buscador de contenidos dentro de la web.

## **Según su funcionalidad u objetivo**

Cabe destacar que casi todas las tipologías descritas a continuación son, según su diseño o estructura, dinámicas puesto que, en general, están pensadas para cumplir una, si no varias, de las necesidades descritas anteriormente.

### **CMS**

Un CMS Content Management System (sistema de gestión de contenido) es una herramienta web que permite, prácticamente, a cualquier usuario publicar, modificar y administrar todo el contenido de un sitio.

Los CMS se caracterizan porque suelen permitir una fácil instalación y actualización a través de asistentes, el uso de plantillas con diferentes tecnologías de Frontend, la gestión de perfiles de usuario y grupos, la optimización de direccionamiento web para ayudar al SEO y la manipulación de los contenidos a través de algún sistema gestor de base de datos. Estos contenidos pueden ser desde los propios artículos o anuncios, hasta el diseño del sitio o complementos externos.

Dentro de los CMS podemos encontrar varios tipos según su intencionalidad. Así, por ejemplo, los blogs están pensados para que, prácticamente, cualquier persona pueda publicar contenidos más o menos cortos de una forma sencilla y la posibilidad de que puedan comentar esos u otros contenidos.

Entre los diferentes sistemas que existen para crear blogs, los más conocidos posiblemente sean WordPress, Joomla, Drupal o Blogger, todos ellos basados en PHP y MySQL.

## **FOROS**

Un foro podría considerarse un CMS, no obstante, la principal diferencia es que, los contenidos, son provistos por los usuarios que visitan el sitio, no por quién lo ha instalado o mantiene. Además, esta herramienta está más pensada para que una gran cantidad de personas participen de forma activa entre los diferentes contenidos con discusiones entrelazadas.

Entre los diferentes sistemas que existen para crear foros, los más conocidos posiblemente sean phpBB, BBPress y MyBB, todos ellos basados en PHP y MySQL.

## **WIKIS**

Una Wiki, también conocida como una comunidad virtual, es un término hawaiano que significa rápido y que hace referencia a un tipo de sistema web que permite a los usuarios realizar todo tipo de operaciones de gestión sobre los contenidos con el objetivo de compartirlo y proporcionar un acceso libre. En pocas palabras, es un sistema que permite que los contenidos puedan ser creados y/o editados por múltiples usuarios a través de un navegador.

Entre los diferentes sistemas que existen para crear wikis, los más conocidos posiblemente sean mediaWiki, DokuWiki y TikiWiki, todos ellos basados en PHP y MySQL.

## **PORTALES WEB**

Un portal web es un tipo de sistema web que está pensado para crear comunidades. En este tipo de sistemas, se provee el acceso, de forma sencilla e integrada, a diversos recursos y servicios externos en los que se tratan temas relacionados o no de forma que puedan generar o suscitar el interés de los usuarios.

Entre los diferentes sistemas que existen para crear portales web, los más conocidos posiblemente sean Drupal, LifeRay, Microsoft Sharepoint y DotNetNuke, todos ellos basados en diferentes tecnologías como PHP, Java o .NET.

## **COMERCIOS ELECTRÓNICOS**

Un comercio electrónico, también conocido como e-commerce o tienda online, es un término que, originalmente, se utilizaba para hacer referencia a sistemas en línea que realizaban transacciones mediante medios electrónicos, no obstante, ahora se identifica más con un tipo de sistema web cuyo objetivo es comprar y vender a través de medios electrónicos.

Entre los diferentes sistemas que existen para crear comercios electrónicos, los más conocidos posiblemente sean PrestaShop, Magento, WooCommerce (que es un plugin de Wordpress) y openCart, todos ellos basados en PHP y MySQL.

## **WEBS CORPORATIVAS**

Una web corporativa, también conocida como web institucional, es un sistema web que está creado con el único objetivo de dar a conocer a una empresa. En este tipo de páginas, lo que se suele mostrar son los productos y servicios que ofrece e información sobre su creación e historia facilitando, además, la posibilidad de contactar con la empresa.

Las webs corporativas suelen estar creadas a medida a partir de las tecnologías que estén en auge en ese momento. Por dar algunos ejemplos, las tecnologías más utilizadas para la creación de webs corporativas son:

- PHP, Java, .Net y Python para el backend.
- MySQL, Oracle y SQL Server como posibles bases de datos.

- HTML5, CSS3 y JavaScript para el frontend.

No obstante, también es bastante frecuente que las webs corporativas se realicen bajo algún CMS o sistema de creación de portales web.

## **PORTFOLIOS**

Un portfolio es un tipo de sistema web en dónde los usuarios muestran sus habilidades, en especial, las que son susceptibles de ser cobradas o contratadas. Ejemplos de este tipo de páginas podrían ser un portfolio de fotografías, donde un fotógrafo muestra sus trabajos a modo de publicidad o, un currículum vitae, donde los trabajadores muestran sus conocimientos a modo de presentación.

Al igual que sucede con las webs corporativas, los portfolios suelen estar creadas a medida y a partir de las tecnologías que estén en auge en ese momento, aunque, tampoco es raro verlos a modo de CMS. Esto es:

- PHP, Java, .Net y Python para el backend.
- MySQL, Oracle y SQL Server como posibles bases de datos.
- HTML5, CSS3 y JavaScript para el frontend.

## **LANDING PAGES**

Una landing page, también conocida como página de aterrizaje, es una página web que tiene, como única finalidad, que los usuarios realicen algún tipo de acción determinada como pulsar en un botón concreto, suscribirse a una newsletter, hacer clic en un banner o anuncio, etcétera.

A diferencia de los tipos anteriores de web, las landing pages suelen ser creadas de forma estática y con diseño impactante para intentar aumentar la tasa de conversión o el porcentaje de éxito.

## **WEBS PERSONALES**

Una web personal es toda página que presenta, como su propio nombre indica, un carácter o enfoque personal. Es el tipo de web al que uno se refiere cuando el objetivo es presentar contenido de naturaleza personal como puedan una web sobre un grupo de música rock, un blog de moda o cocina, un diario o una página sobre los templarios.

Uno podría pensar que un portfolio de fotografía o un currículum vitae son páginas personales y, sí, tienen un carácter personal, no obstante, existe una gran diferencia, el objetivo. Las páginas personales no están pensadas para ofrecer servicios, más bien están pensadas para informar, entretenér o dar a conocer.

El uso de las tecnologías en este tipo de páginas es muy variable, desde sistemas más o menos complejos como un CMS, hasta páginas creadas artesanalmente a través de las tecnologías que estén en ese momento en auge.

## **LENGUAJES DE MARCADO**

Los lenguajes de marcado, también se les suele llamar lenguajes de marcas, aunque este tipo de referenciación no es nada habitual.

Históricamente, los lenguajes de marcado eran utilizados por las editoriales y medios de comunicación para imprimir instrucciones concretas en los márgenes. Para ello, utilizaban unos “marcadores” que indicaban el tipo de fuente, estilo, tamaño de letra y la corrección de errores. Una cosa llevó a la otra y, con el tiempo, se estandarizaron como un grupo de marcas que, más tarde, fueron reinterpretadas en lo que hoy conocemos como lenguajes de marcado.

El lenguaje de marcado es un término que hace referencia a una manera de codificar la información. Esta codificación se caracteriza porque los contenidos están envueltos y declarados a través de unas entidades que llamaremos etiquetas o marcas.

En este contexto, cuando se habla de etiquetas o marcas, además se les suele dar un significado que puede indicar qué es o qué debe contener, dónde utilizarlo o cómo presentarlo. No obstante, ninguna de estas marcas tiene la capacidad de ejecutar acciones ni de realizar operaciones de ninguna clase.

Esto último debe tenerse claro porque, en ocasiones, los lenguajes de marcado se confunden con los lenguajes de programación, y nada más lejos. Un lenguaje de programación es un idioma de computación que presenta varias peculiaridades como son la posibilidad de declarar variables, realizar operaciones matemáticas o crear fragmentos de código que suelen ser agrupados en procedimientos o funciones. Un lenguaje de marcado no hace nada de eso, sólo etiqueta la información aportándole un valor semántico.

## **Características de los lenguajes de marcado**

Un lenguaje de marcado se caracteriza porque tiene marcas o etiquetas que son declaradas a modo de elemento o entidad. Cada uno de estos elementos o entidades tiene un contenido y uno o varios atributos que definen su comportamiento, su función y su significado.

Un ejemplo de todo esto podría ser la declaración de un producto.

```
<producto>
<cabecera>
<titulo>Nombre producto</titulo>
<subtitulo>Categoría</subtitulo>
</cabecera>
<descripcion>
<texto>Descripción del producto</texto>
</descripcion>
<pie>
<negrita>Precio</negrita>
<accion>Comprar</accion>
</pie>
</producto>
```

Ejemplo de lenguaje de marcado

Como se puede observar en la ilustración anterior, se ha definido una entidad concreta a través de unos elementos que definen su significado y comportamiento y que, en ocasiones, tienen propiedades intrínsecas que aportan un valor a la presentación.

Un ejemplo de ello podría ser la etiqueta SUBTITULO, que indica que, su contenido textual, es un texto de cabecera o a modo de título y que está en el segundo nivel de la jerarquía en lo referente a títulos.

Si, además, este código fuese interpretado por una herramienta o sistema específico, podría llevar una interpretación específica que provocase que fuese presentado de forma concreta, con un tamaño, margen y estilo concretos, entre otras cosas.

## **Clasificación de los lenguajes de marcado**

Los lenguajes de marcado suelen ser clasificados en tres grupos, no obstante, pueden llegar a mezclarse propiedades de varios en uno, formando nuevas especificaciones.

## MARCADO DE PRESENTACIÓN

El marcado de presentación es aquel que está enfocado hacia la presentación de la información del documento, es decir, es aquel que sirve para dar formato al texto. Este tipo de marcado es suficiente para el proceso de lectura, pero insuficiente para el procesamiento automático de la información. Además, las etiquetas de marcado están ocultas al usuario, lo que hace que su contenido esté ofuscado y sea complicado de extraer si no es con la herramienta adecuada.

El marcado de presentación puede ser sencillo de elaborar, sin embargo, su mantenimiento y/o modificación pueden volverse complicados, por lo que su uso no está demasiado extendido.

Un ejemplo de marcado de presentación es RTF, en donde la composición de las etiquetas suele proporcionar un comportamiento y/o un significado distinto. Es decir, el título del documento podría venir predefinido por una combinación de varios saltos de línea en cualquier posición del documento y terminar con un carácter especial justo detrás del contenido textual.

```
{\rtfi\ansi\ansicpg1252\deff0\deflang3082
{\fonttbl{\fo\fswiss\fcharset0 Arial;}{\fi\fmodern\fprq1\fcharset0 Arial;}}
{\colortbl ;\red255\green0\blue0;\red0\green0\blue128;}
\viewkind4\uc1\pard\b\fo\fs20 RTF\bo es un \cf1\b\fs24 lenguaje de marcado de presentaci\f3 \fo\fs20 .\par}
```

Fragmento de código de documento RTF

## MARCADO DE PROCEDIMIENTOS

El marcado de procedimientos es aquel que está enfocado hacia la presentación del texto, al igual que sucede con el marcado de presentación. No obstante, las marcas o etiquetas que componen el documento no están ocultas, es decir, son visibles en todo momento para el usuario.

El marcado de procedimientos es sencillo de elaborar y de modificar. Además, es interpretado por orden de aparición, por lo que ayuda a predecir cómo será su resultado durante el proceso de lectura.

Un ejemplo de marcado de procedimientos es LaTeX y HTML. Ambos requieren de una marca o etiqueta que funciona a modo de identificador especial que le indica al sistema cómo se debe renderizar el contenido textual. Este renderizado suele venir definido a través de una serie de instrucciones que establecen el texto con un tipo de fuente, tamaño, alineación y estilos concretos.

```
<article>
<header>
<h2>Nombre producto</h2>
</header>
<div class="descripcion">
<p>Descripción del producto</p>
</div>
<footer>
<b>Precio</b>
<button>Comprar</button>
</footer>
</article>
```

Fragmento de código de documento HTML5

## MARCADO DESCRIPTIVO

El marcado descriptivo, también conocido como semántico, es aquel que utiliza las etiquetas para describir fragmentos de texto, pero sin especificar en qué orden o cómo deben representarse.

El marcado descriptivo suele ser sencillo de elaborar y modificar y, a menudo, sus etiquetas pueden ir acompañadas de atributos que definen su significado o comportamiento, pero su interpretación está separada de la presentación, es decir, sólo especifica la descripción del tipo y contenido de los documentos.

Al marcado descriptivo también se le atribuye la cualidad de simplificar el proceso de reformato del texto, principalmente, porque la información del formato está separada del propio contenido. Por esta razón, un fragmento indicado como cursiva a través de la etiqueta *I* (italic), podría emplearse para marcar énfasis o para indicar un contenido gráfico.

Cabe destacar que, si quisiéramos sortear esta última situación en el marcado de presentación y en el marcado de procedimientos, el proceso probablemente sería bastante tedioso, sin embargo, si los contenidos se hubiesen diferenciado descriptivamente mediante etiquetas distintas, podrían representarse de manera diferente sin esfuerzo.

Un ejemplo de marcado descriptivo es SGML y el XML. Ambos son muy flexibles ya que los fragmentos se etiquetan como son y no como deberían mostrarse.

```
<!doctype email system "email.dtd">
<email>
<to>info@ejemplo.com
<from>alumno@gmail.com
<date>mon, 29 jan 2020 12:00:02 - 0100 (est)
<subject>error de acceso
<contents>
Hola, buenas tardes. ¿Mi cuenta está bloqueada o deshabilitada?.
<url>http://ejemplo.com</url>
saludos
</contents>
</email>
```

Fragmento de código de documento SGML

## 2

# HTML5

El lenguaje HTML (HyperText Markup Language o lenguaje de marcado de hipertexto) es un lenguaje de marcado dedicado a la elaboración de páginas web. Fue definido por primera vez en 1991 y, en aquel entonces, se caracterizaba por tener algo más de una docena de etiquetas. Más tarde, en 1995 se publicó el primer estándar oficial de HTML al que denominaron HTML 2.0.

En 1997 entró en juego la W3C y desarrolló tres estándares más hasta llegar a lo que hoy conocemos como HTML5 en 2014.

Si bien HTML es un lenguaje formado por entidades que ayudan a estructurar y proporcionar significado a las diferentes partes del documento, cada una de estas entidades, usualmente denominadas elementos o etiquetas, están

formadas por un contenido y cero, uno o varios atributos.

```
<p>Esto es un párrafo</p>
<div class="layer">Esto es una capa</div>
```

Fragmento de código de documento HTML

Cada uno de los atributos tiene una función y puede estar o no asociado a un comportamiento o definición específica. Por ejemplo, el atributo ID habitualmente es utilizado para poder manipular el elemento a través de un nombre corto, sin embargo, también puede ser declarado para vincularse con otro elemento generando una entidad mayor, como es el caso del siguiente código.

```
<label for="nombre">Nombre</label>
<input id="nombre" placeholder="Inserte el nombre completo" />
```

Etiquetado de un campo de formulario en HTML

El atributo FOR, utiliza el atributo ID para vincular el LABEL con el INPUT y generar un elemento combinado o pequeño componente.

Cabe destacar que, aunque puede haber etiquetas sin cierre, como es el caso del elemento INPUT, lo normal es que todas las etiquetas o marcas tengan un principio y un final, como es el caso de la etiqueta LABEL.

En lo referente a las novedades de HTML5, como muchos sabrán, una de las más significativas es el valor semántico. La semántica es una característica que dota a los documentos web de mayor significado porque, entre otras cosas, proporciona una mayor estructuración y ayuda a la compresión gracias a lo que se denomina identificador semántico.

El identificador semántico es un término que hace referencia a lo que contiene o representa la etiqueta, es decir, cada etiqueta o elemento tiene un nombre asociado que representa o indica su objetivo. Por ejemplo, en general, la etiqueta SECTION siempre contendrá un conjunto de elementos agrupados que tendrán o guardarán una relación.

## **ELEMENTOS BÁSICOS DE HTML5**

### **Definición del tipo de documento DTD (!DOCTYPE)**

Cuando uno decide trabajar con HTML, lo primero que debe hacer es declarar es el elemento !DOCTYPE. Este elemento tiene, como objetivo, informar al navegador del tipo de documento que se va a definir.

La Declaración del Tipo de Documento (DTD) puede cambiar, y de hecho cambia, para cada versión de HTML. La versión del lenguaje de marcado puede ser muy diferente según qué tipo se utilice, y puede tener más o menos restricciones en función del modo y versión. Sin ir más lejos, el tipo de documento que se debe definir para indicar que es un documento XHTML es muy distinto al que se debe usar para indicar que es HTML5 o SVG.

A continuación, se muestran los principales DTD para documentos de HTML, SVG y MathML.

### **DTDS DE HTML**

#### **HTML5**

```
<!DOCTYPE html>
```

#### **HTML4.01 Estricto**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
```

## **HTML4.01 Transicional**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">
```

## **HTML 3.2**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

## **DTDS DE MATHML**

### **MathML 2.0**

```
<!DOCTYPE math PUBLIC "-//W3C//DTD MathML 2.0//EN"  
"http://www.w3.org/TR/MathML2/dtd/mathml2.dtd">
```

### **MathML 1.0**

```
<!DOCTYPE math SYSTEM "http://www.w3.org/Math/DTD/mathml1/mathml.dtd">
```

## **DTDS DE SVG**

### **SVG 1.1 Full**

```
<!DOCTYPE svg PUBLIC  
"-//W3C//DTD SVG 1.1//EN"  
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
```

### **SVG 1.1 Básico**

```
<!DOCTYPE svg PUBLIC  
"-//W3C//DTD SVG 1.1 Basic//EN"  
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-basic.dtd">
```

### **SVG 1.1 Reducido**

```
<!DOCTYPE svg PUBLIC  
"-//W3C//DTD SVG 1.1 Tiny//EN"  
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-tiny.dtd">
```

## **DTDS DE XHTML**

### **XHTML1.1**

```
<!DOCTYPE html PUBLIC  
"-//W3C//DTD XHTML 1.1//EN"  
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

### **XHTML1.0 Estricto**

```
<!DOCTYPE html PUBLIC  
"-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

### **XHTML1.0 Transicional**

```
<!DOCTYPE html PUBLIC  
"-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

## **Etiqueta html**

La etiqueta HTML es el elemento que representa la raíz o base de un documento HTML y supone el cierre automático de todo el resto de los elementos declarados posteriormente a él.

La etiqueta HTML admite varias propiedades o atributos, pero la única que no está en desuso es LANG. El atributo LANG define el lenguaje del documento y es muy útil cuando se disponen de documentos en distintos idiomas y para aquellos usuarios que dependen de herramientas de asistencia como lectores de pantalla.

```
<html lang="es">...</html>
```

## **Etiqueta head**

La etiqueta HEAD es el elemento o la estructura que proporciona información general acerca del documento. Esta información general viene definida a modo de metadatos, o lo que es lo mismo, datos que informan sobre los datos y, pueden ser de muy diferente índole. Esto es, el tipo de codificación, el título del documento, las palabras clave que lo describen, la descripción sobre lo que contiene, el autor del documento, etcétera.

No obstante, también suele contener uno o varios elementos LINK o STYLE que recogen todas las reglas CSS aplicables en el documento.

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Mi primer HTML</title>
<meta charset="utf-8">
<meta name="author" content="Pablo E. Fernández Casado">
<link rel="stylesheet" type="text/css" href="custom.css">
</head>
</html>
```

Aunque la etiqueta SCRIPT puede estar definida dentro del elemento HEAD, lo habitual es que esté al final de la etiqueta BODY para evitar bloqueos o retrasos en la muestra del primer renderizado.

## **Etiqueta body**

La etiqueta BODY es el elemento o la estructura que se define todo el contenido útil del documento. Aquí es dónde se definirán todos los textos, capas, botones, controles de entrada y salida, etcétera para que los usuarios puedan utilizarlo o consultarla.

Al final de esta estructura, habitualmente, suele contener uno o varios elementos SCRIPT que todas las funcionalidades que se ejecutan en el navegador, como validaciones o animaciones.

```
<!DOCTYPE html>
<html lang="es">
<head>
...
</head>
<body>
<!-- Esto es el cuerpo de la página, dónde se debe introducir --&gt;
<!-- el contenido a mostrar u ofrecer --&gt;
&lt;/body&gt;
&lt;/html&gt;</pre>
```

## Comentarios

Los comentarios en HTML se establecen a través de las marcas de <!-- y -->. Esta etiquetas o marcas indican al navegador que la información contenida no debe ser interpretada y, por tanto, tampoco renderizada. Un ejemplo podría ser:

```
<!-- Esto es un comentario de HTML -->
```

## ATRIBUTOS COMUNES O GENÉRICOS

### Atributo accesskey

El atributo ACCESSKEY especifica un atajo de teclado (o combinación de teclas) que el usuario puede pulsar para interactuar con el elemento.

```
<button accesskey="h">Ayuda</button>
```

Como se puede observar en la ilustración anterior, la tecla de activación es la letra H, no obstante, dependiendo del navegador y sistema operativo, se activará de una u otra manera.

Navegador	Windows	Linux	iOS
<b>Firefox</b>	Alt+Shift+Tecla	Alt+Shift+Tecla	Command+Alt+Tecla
<b>Edge</b>	Alt+Tecla	n/a	
<b>Internet Explorer</b>	Alt+Tecla	n/a	
<b>Chrome</b>	Alt+Tecla	Alt+Tecla	Command+Alt+Tecla
<b>Safari</b>	Alt+Tecla	N/A	Command+Alt+Tecla
<b>Opera 15+</b>	Alt+Tecla	Alt+Tecla	Command+Alt+Tecla

Cabe destacar que la utilización del atributo ACCESSKEY puede afectar de forma considerable a la accesibilidad web. Por ejemplo, un atajo de teclado programado con este atributo puede entrar en conflicto con la herramienta de asistencia que esté utilizando el usuario e, incluso, puede pasar que el usuario presente una discapacidad cognitiva y el atajo de teclado le resulte confuso.

### Atributo autocapitalize

El atributo AUTOCAPITALIZE especifica si el contenido que se va a representar debe capitalizarse de manera automática o no. Para ello, se nos proveen cuatro posibilidades.

- **OFF** o **NONE**: Se le indica al navegador que no se transforme el contenido.
- **ON** o **SENTENCES**: Se le indica al navegador que el contenido se transforme como si fuese una oración, es decir, el primer carácter del párrafo o elemento.
- **WORDS**: Se le indica al navegador que se capitalice cada una de las palabras del contenido.
- **CHARACTERS**: Se le indica al navegador que se transforme todo a mayúsculas.

```
<label>  
Nombre:  
<input type="text" name="name" autocapitalize="words">  
</label>
```

Por desgracia, parece que esta funcionalidad todavía no está soportada por casi ningún navegador. No obstante, es posible reproducir todas estas funcionalidades a través de las propiedades FONT-VARIANT y TEXT-TRANSFORM de CSS.

## Atributo autocomplete

El atributo AUTOCOMPLETE especifica si el contenido que se va a representar debe ser manipulado por el navegador y herramientas de asistencia de manera automática. Para ello, se nos proveen tres posibilidades.

- **OFF**: Se le indica al navegador que no se deben sugerir opciones automáticamente.
- **ON**: Se le indica al navegador que se deberían sugerir opciones automáticamente, pero sin indicar el tipo de dato que se espera recibir.
- **[LISTA DE IDENTIFICADORES]**: Se le indica al navegador que se deberían sugerir opciones automáticamente y especificando el tipo de dato que se espera recibir.

La lista de posibles valores que puede tomar este atributo es francamente amplia y muy utilizada en contextos de accesibilidad web, con más de 50 opciones. Todas ellas están descritas y pueden ser consultadas en MDN Web Docs Moz://a, en la URL <https://developer.mozilla.org/es/docs/Web/HTML/Atributos/autocomplete>.

```
<label>
Nombre:
<input type="text" name="name" autocomplete="off">
</label>
```

## Atributo class

El atributo CLASS especifica una definición de clase a la etiqueta o elemento. A diferencia del atributo ID, el valor de este atributo puede repetirse en cualquier parte del documento.

```
<p class="toLeft">
Este párrafo tiene una clase que hace que se alinee a la izquierda
</p>
<p class="toRight">
Este párrafo tiene una clase que hace que se alineé a la derecha
</p>
```

## Atributo contenteditable

El atributo CONTENTEDITABLE especifica al navegador que el elemento es editable, aunque originalmente no lo fuese. Es decir, permite que el elemento pueda ser modificado en lo referente a su contenido, formato y estilos y pueda ser manipulado a través de los eventos asociados a la edición, como puedan ser el evento FOCUS o el evento KEYPRESS.

```
<p contentEditable="true">
Este párrafo es modificable,
<span>y esta etiqueta también.</span>
</p>
```

Cabe destacar que el ámbito de aplicación de este atributo afecta a todos sus hijos. Es decir, que al estar el elemento SPAN (de la ilustración anterior) dentro de un elemento de párrafo editable, también se volverá editable.

Como nota adicional añadiremos que, desde JavaScript, es posible conocer si el elemento tiene esta propiedad establecida a través del método isContentEditable.

## Atributo draggable

El atributo DRAGGABLE especifica que el elemento indicado puede ser arrastrado a un contenedor.

```
<p draggable="true">Este es un párrafo arrastable</p>
```

Cabe destacar que, cuando se define un atributo DRAGGABLE, también es importante definir el atributo TITLE para proporcionar un identificador único cuando se realizan interacciones a través de herramientas de asistencia.

Además, en el elemento origen se debe declarar el evento ONDRAGSTART y, en el objeto destino, los eventos ONDROP y ONDRAFTER.

## Atributo contextmenu

El atributo CONTEXTMENU especifica el ID de un elemento MENU, que presupone la presentación de un menú contextual que aparece, por ejemplo, cuando se pulsa el botón derecho del ratón. Sin embargo, a agosto de 2020, este atributo no está soportado por casi ningún navegador, por lo que no haremos mucho más que mostrar un ejemplo.

```
<menu type="context" id="menumegusta">
<menuitem label="Copiar" onclick="copy()"></menuitem>
<menuitem label="Cortar" onclick="cut()"></menuitem>
<menuitem label="Pegar" onclick="paste()"></menuitem>
</menu>
```

## Atributo dir

El atributo DIR especifica la direccionalidad del texto, es decir, si los contenidos están escritos en un idioma que se lee de izquierda a derecha, o de derecha a izquierda.

```
<p dir="rtl">
Este texto aparecerá escrito de derecha a izquierda, además de estar
alineado a la derecha
</p>
```

## Atributo enterkeyhint

El atributo ENTERKEYHINT especifica la etiqueta de acción o ícono que representa la tecla de salto de línea (enter) en los teclados virtuales. Por tanto, sólo podrá ser definido en controles de formulario, como el elemento TEXTAREA, o en aquellos elementos que tengan establecida la propiedad CONTENTEDITABLE a true.

```
<input enterkeyhint="search">
```

Entre los posibles valores que puede tomar este atributo, tenemos las opciones de ENTER, DONE, GO, NEXT, PREVIOUS, SEARCH y SEND.

- **ENTER:** El navegador producirá un salto de línea.
- **DONE:** El navegador mostrará una señal de “hecho” que, por lo general, significa que el editor se cerrará indicando que no hay nada más que ingresar.
- **GO:** El navegador mostrará una señal de “ir” que, por lo general, significa mover al objetivo especificado.
- **NEXT:** El navegador mostrará una señal de “siguiente” que, por lo general, significa que se mueva el cursor al siguiente campo de texto enfocable.
- **PREVIOUS:** El navegador mostrará una señal de “anterior” que, por lo general, significa que se mueva el cursor al anterior campo de texto enfocable.

- **SEARCH:** El navegador mostrará una señal de “buscar” que, por lo general, significa que se muestren los resultados de búsqueda relacionados con el texto escrito.
- **SEND:** El navegador mostrará una señal de “enviar” que, por lo general, significa que se envíe el contenido a un servidor u objetivo destino.

Cabe destacar que, este atributo, a agosto de 2020 todavía se encuentra en fase experimental, por lo que puede que no sea efectivo en múltiples navegadores.

## Atributo hidden

El atributo HIDDEN especifica que el elemento no es relevante y, como consecuencia, no debe mostrarse en pantalla. Esto puede ser útil cuando se desea que los elementos del documento, o de una sección del mismo, se vuelvan visibles en función de una condición o circunstancia determinada.

En general, la manipulación del atributo HIDDEN se realiza a través de un lenguaje de programación, como pueda ser JavaScript, aunque también es posible establecerlo a través de otros medios, como pueda ser el renderizado parcial mediante peticiones Ajax al servidor.

```
<p hidden>Este párrafo permanecerá oculto hasta nueva orden.</p>
```

Cabe destacar que, la visualización u ocultación de elementos también es posible realizar a través de CSS, mediante la propiedad DISPLAY.

## Atributo inputmode

El atributo INPUTMODE sugiere, o proporciona pistas, sobre el tipo de datos que se pueden introducir en los elementos editables. Por tanto, sólo podrá ser definido en controles de formulario, como los elementos INPUT y TEXTAREA, o en aquellos elementos que tengan establecida la propiedad CONTENTEDITABLE a true.

Entre los diferentes valores que puede tomar, podemos escoger entre las opciones de none, text, decimal, numeric, tel, search, email y url.

- **NONE:** Indica que no tiene teclado virtual y que la página implementará su propio control de entrada.
- **TEXT:** Indica que se utilizará la entrada estándar definida.
- **DECIMAL:** Indica que la entrada será un valor numérico con decimales separados por coma o punto. La visualización del signo negativo dependerá del dispositivo donde se reproduzca.
- **NUMERIC:** Indica que la entrada será un valor numérico entero. La visualización del signo negativo dependerá del dispositivo donde se reproduzca.
- **TEL:** Indica que la entrada permitirá los dígitos del 0 al 9, el \* y #.
- **SEARCH:** Indica que el teclado virtual debe optimizarse para la búsqueda.
- **EMAIL:** Indica que la entrada es de tipo email.
- **URL:** Indica que la entrada es de tipo URL.

```
<input type="text" name="email" inputmode="email" />
<input type="text" name="searcher" inputmode="serach" />
```

Cabe destacar que, este atributo, a agosto de 2020 Firefox presenta un soporte parcial y que Safari e Internet Explorer no lo soportan.

## Atributo id

El atributo ID especifica un identificador que vincula la etiqueta HTML con ese nombre. Dada la importancia que tienen para los lenguajes de script, la accesibilidad web, la usabilidad web y el posicionamiento SEO, los identificadores de etiqueta no deben repetirse bajo el mismo contexto.

```
<label>
  Nombre completo
  <input id="name" />
</label>
```

## Atributo is

El atributo IS especifica que el elemento pertenece a un Web Component y, como consecuencia de ello, representa una extensión de otro elemento HTML estándar.

```
<script type="text/javascript">
class Autor extends HTMLElement {
  constructor() {
    super();
    // ...
  }
  customElements.define("custom-autor", Autor);
</script>
<span is="custom-autor"></span>
```

La creación y definición de Web Components está descrita en varios sitios web y, también, explicada en detalle en el libro “Domine JavaScript 4<sup>a</sup> Edición” de esta misma editorial.

Si este atributo no resulta del todo funcional o se tiene algún problema con su implementación, se puede recurrir a algún polyfill para habilitar dicha funcionalidad a través de JavaScript.

## Atributo itemid

El atributo ITEMID forma parte de lo que se denomina la sintaxis de microdatos y especifica el identificador global único que puede ser utilizado como microdato.

Los atributos ITEMID sólo se pueden utilizar en aquellos elementos que tengan establecidas las propiedades de ITEMSCOPE e ITEMTYPE.

```
<dl itemscope
  itemtype="http://vocab.ejemplo.com/libros"
  itemid="uri:1-001-34567-8">
</dl>
```

Cabe destacar que los identificadores de este atributo deben estar permitidos por la especificación del vocabulario proporcionado en el atributo ITEMTYPE, de lo contrario, pueden no producir el efecto deseado.

## Atributo itemprop

El atributo ITEMPROP forma parte de lo que se denomina la sintaxis de microdatos y especifica una o varias propiedades (separadas por espacios) que ayudan a identificar el propósito del dato.

```
<dl itemscope>
  <dt>Título</dt>
  <dd itemprop="title">Domine JavaScript 4a Edición</dd>
```

```

<dt>Autor</dt>
<dd itemprop="author">Pablo E. Fernández Casado</dd>
<dt>Fecha de publicación</dt>
<dd>
<time itemprop="pubdate" datetime="2020-02-14">14 Feb 2020</time>
</dd>
</dl>

```

Cabe destacar que, los identificadores de este atributo deben estar permitidos por la especificación del vocabulario proporcionado en el atributo ITEMTYPE.

## Atributo itemref

El atributo ITEMREF sólo se puede definir en aquellos elementos que tengan definido el atributo ITEMSCOPE y, al igual que ITEMID e ITEMPROP, forma parte de lo que se denomina la sintaxis de microdatos.

Su objetivo es especificar uno o varios valores (separados por espacios) que coinciden con los identificadores de elemento proporcionados por el atributo ID.

```

<div itemscope id="domineJS" itemref="title type"></div>
<p id="title">
Título: <span itemprop="title">Domine JavaScript 4ª Edición</span>
</p>
<div id="type" itemprop="libro" itemscope itemref="details"></div>
<div id="details">
<p>Autor: <span itemprop="author">Jazz Band</span> </p>
<p>Publicado el: <span itemprop="pubdate">14-02-2020</span></p>
</div>

```

Cabe destacar que, los identificadores de este atributo deben estar permitidos por la especificación del vocabulario proporcionado en el atributo ITEMTYPE.

## Atributo itemscope

El atributo ITEMSCOPE especifica si el elemento pertenece a un modelo de sintaxis de microdatos. Aunque sus posibles valores son TRUE o FALSE, con que esté presente, ya se define como elemento del modelo de sintaxis de microdatos.

```

<ul itemscope>
<li>Título</li>
<li itemprop="title">Domine JavaScript 4ª Edición</li>
</ul>

```

## Atributo itemtype

El atributo ITEMTYPE especifica una lista, con uno o varios valores, de direcciones web que definen el vocabulario válido para los atributos ITEMPROP. Si la lista contiene varios valores, todos los elementos deberán definir el mismo vocabulario.

```

<ul itemscope>
<li>Título</li>
<li itemprop="title">Domine JavaScript 4ª Edición</li>
</ul>

```

Cabe destacar que, la mayoría de los diccionarios de vocabulario pueden encontrarse en <https://schema.org>.

Según Schema.org, es una actividad comunitaria colaborativa con la misión de crear, mantener y promover esquemas para datos estructurados en Internet, en páginas web, en mensajes de correo electrónico y más.

El vocabulario de Schema.org se puede usar con muchas codificaciones diferentes, incluidas RDFa, Microdata y JSON-LD. Estos vocabularios cubren entidades, relaciones entre entidades y acciones, y pueden extenderse fácilmente a través de un modelo de extensión bien documentado. Más de 10 millones de sitios usan Schema.org para marcar sus páginas web y mensajes de correo electrónico. Muchas aplicaciones de Google, Microsoft, Pinterest, Yandex y otras ya utilizan estos vocabularios para impulsar experiencias ricas y extensibles.”

## Atributo lang

El atributo LANG especifica el idioma en el que está descrito el contenido del elemento. La anotación para indicar el idioma debe ser representada bajo el estándar ISO-639-1, el cual identifica el lenguaje a partir de dos caracteres.

```
<p lang="es">Mi nombre es Pablo</p>
<p lang="en">My name is Paul</p>
```

Dado que HTML tiene un carácter hereditario, su aplicación afectará al propio elemento y sus hijos, al igual que sucede cuando lo declaramos en la etiqueta HTML, que indica que todo el documento, y por tanto, todos sus elementos, están definidos en ese idioma.

## Atributo nonce

El atributo NONCE especifica un código criptográfico de uso único que puede ser utilizado por la Política de Seguridad de Contenido para indicar si el elemento va a ser cargado y aplicado al documento actual o si se permitirá que se continúe con una acción determinada.

Dicho de otra forma, el atributo NONCE resulta ser una manera de decirle a los agentes de usuario que el elemento SCRIPT o LINK que está definido en el documento, no fue injectado por un tercero (malicioso), sino que fue colocado de forma intencionada en el documento por un usuario legítimo que puede manipular el servidor.

```
<link rel="search" href="booksearch.asp"
nonce="bGlicm8tY29uc3JodWNjacOzbi1kZS1wYWdpbmFzLXdIYg==>
```

El código de este atributo, en general, se genera en el servidor, aunque resulta posible obtener un valor legítimo a través de las herramientas online que están disponibles en Internet. Su valor se corresponde con una codificación en Base64 de, al menos, 128 bits.

Para indicarle al servidor que ese NONCE es legítimo, en el encabezado de las peticiones se debe declarar una CSP (Política de Seguridad de Contenido) como la siguiente:

```
Content-Security-Policy: script-src 'nonce-
bGlicm8tY29uc3JodWNjacOzbi1kZS1wYWdpbmFzLXdIYg=='
```

Esto ayudará a evitar ciertos ataques, incluyendo los famosos ataques XSS (Cross-Site Scripting) y ataques por inyección de datos.

## Atributo slot

El atributo SLOT especifica o asigna un espacio dentro del árbol Shadow DOM de un componente web. Como dato importante, el valor de este atributo debe coincidir exactamente con el atributo NAME del elemento que define el atributo SLOT.

```
<span slot="book">Domine JavaScript 4a Edición</span>
```

La creación y definición de Web Components está descrita en varios sitios web y, también, explicada en detalle en el libro “Domine JavaScript 4<sup>a</sup> Edición” de esta misma editorial.

No obstante, esta especificación todavía está en fase experimental y, por tanto, es posible que se requiera de algún polyfill escrito en un lenguaje como JavaScript para poder utilizarlo.

## Atributo style

El atributo STYLE especifica las propiedades de estilo para un elemento o conjunto de ellos. Aunque es altamente aconsejable que las propiedades de estilo se definan en documentos CSS independientes, esta etiqueta puede ser útil para declarar estilos de uso específico o para realizar pruebas.

```
<h1 style="margin-top: 5px;">Título con margen superior</h1>
```

## Atributo title

El atributo TITLE especifica el mensaje, a modo de tooltip, que deberá ser mostrado cuando se sitúe el dispositivo puntero sobre dicho elemento.

```
<p title="Esto es un párrafo de prueba!">  
Esto es un párrafo con un mensaje emergente  
</p>
```

## Atributo tabindex

El atributo TABINDEX especifica si el elemento puede obtener el foco y su posición relativa en la secuenciación de obtención del foco.

Por ejemplo, si su valor es -1, el elemento no podrá tomar el foco. Los elementos no modificables suelen tener esta opción definida por defecto.

Si su valor es 0, podrá tomar el foco, pero por orden de llegada, es decir, por el orden que se establezca según se van declarando los elementos. Los elementos modificables tienen esta opción definida por defecto.

Ahora bien, si su valor es mayor que 0, lo que se indica al agente de usuario es que tiene prioridad con respecto a los elementos que tengan un TABINDEX menor.

```
<p tabindex="0">  
Este párrafo puede tomar el foco  
</p>
```

## Atributos personalizados

Los atributos personalizados podrían definirse como un contenedor de datos modificables que permiten su manipulación e integración con el DOM de los lenguajes de script como JavaScript.

Una de las razones por las que se dotó a HTML5 de esta capacidad es porque, hasta entonces, el almacenamiento de información asociada a elementos independientes se hacía a través del atributo CLASS, lo que muchas veces no resultaba muy elegante ni claro.

Hoy día, los atributos personalizados son útiles en muchas ocasiones. Por ejemplo, imaginemos una situación en la que se está ofreciendo una lista de elementos seleccionables a través de una tabla. En esta supuesta tabla, se están mostrando datos al usuario, pero no se desea mostrar el identificador de registro. Sin embargo, para conocer qué registro o elemento seleccionó el usuario, debemos recurrir a ese ID.

Pues bien, se puede almacenar en un atributo personalizable y, más tarde, con alguna funcionalidad creada en un lenguaje de script como JavaScript, recuperar ese ID y enviárselo al servidor a través de Ajax.

La forma de definir un atributo personalizado es mediante el prefijo DATA- y un sufijo que no puede empezar con la palabra XML, ni contener puntos, comas o mayúsculas.

```
<p id="p1" data-id="oo2">  
Esto es un párrafo con un atributo personalizado  
</p>
```

A modo informativo, también se puede recuperar el atributo personalizado a través de un lenguaje de script como JavaScript. La forma de recuperarlo es mediante el uso de la propiedad DATASET, la cual proporciona una interfaz para manipular los atributos personalizados definidos en HTML5. En este caso, para acceder al atributo anteriormente expuesto, podríamos hacer lo siguiente:

```
<script type="text/javascript">  
document.querySelector("#p1").dataset.id;  
</script>
```

## INFORMACIÓN DEL DOCUMENTO

Los elementos que proporcionan información sobre el documento son las etiquetas TITLE, BASE, LINK, STYLE y SCRIPT.

### Elemento base

El elemento BASE especifica la base de direccionamiento predeterminada para los elementos que utilicen un direccionamiento relativo, es decir, las direcciones de no tienen como primer carácter el símbolo de barra inclinada (/) y no empiezan por un identificador de dominio. Cabe destacar que, si se declaran varios elementos BASE, la última declaración, anulará todas las anteriores. Un ejemplo podría ser:

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
<base href="https://www.ejemplo.com/" target="_blank" />  
</head>  
<body>  
  
<a href="contenidos/tutorial.html">Tutorial</a>  
</body>  
</html>
```

### Elemento link

El elemento LINK especifica un enlace hacia una hoja de estilos externa. Un ejemplo podría ser:

```
<link rel="stylesheet" type="text/css" href="custom.css" />
```

El único atributo obligatorio del elemento LINK es REL, que es quién define la relación que existe entre el documento actual y el enlazado, es decir, si es una hoja de estilos, un ícono, un archivo de ayuda, un documento de alternativa, ...) y su valor más recurrente es STYLESHEET.

No obstante, aunque el establecimiento de este atributo es importante y ayuda a la semántica web, también puede influir en el rendimiento de forma notable haciendo que los recursos se carguen de maneras diferentes.

Por esta razón, es importante que se conozca bien la aplicación que se está creando y optimizar los recursos, porque un mal uso del atributo REL puede influir negativamente en el rendimiento de la página.

A continuación, se muestran algunos de los valores de REL que afectan al rendimiento:

<b>Valor de REL</b>	<b>Descripción</b>
<b>dns-prefetch</b>	<p>La cláusula DNS-PREFETCH indica al agente de usuario que se resuelva el DNS lo antes posible, es decir, que se resuelva el dominio del servidor, pero no realice ninguna descarga.</p> <p>Cabe destacar que, la resolución del dominio puede llevar entre 20 y 120 milisegundos, por lo que este tiempo hay que tenerlo en cuenta a la hora de realizar la primera petición. No obstante, podemos afirmar que solicitar la exploración previa del DNS minimizará el tiempo y agilizará la carga de la página.</p> <p>Es especialmente útil cuando se desean cargar archivos desde fuentes externas como pueda ser una fuente vectorial de Google Fonts, un script de un CDN o un JSON desde una API.</p>
<b>prefetch / preload</b>	<p>Las cláusulas PREFETCH y PRELOAD indican al agente de usuario que descargue y almacene en caché un recurso determinado, como pueda ser una hoja de estilos o un script.</p> <p>Si la cláusula es PREFETCH, la descarga se realizará con prioridad baja, es decir, sin afectar a los recursos más importantes o prioritarios, no obstante, puede no tener efecto si el agente de usuario así lo considera, por la razón que sea.</p> <p>Si la cláusula es PRELOAD, la descarga se realizará a la mayor brevedad posible, pudiendo afectar a los recursos más importantes o prioritarios.</p> <p>Para que la descarga se realice de manera correcta, las cláusulas PREFETCH y PRELOAD se alimentan del atributo AS, que proporciona una pista de la prioridad del recurso al agente de usuario. Los principales valores del atributo AS son STYLE, SCRIPT, FONT, IMAGE, AUDIO, VIDEO, TRACK, IMAGE, OBJECT y FETCH (para indicar que es un recurso descargado con XMLHttpRequest o la API FETCH de JavaScript).</p> <p>Es especialmente útil cuando se desean cargar fuentes vectoriales desde servidores externos como Google Fonts.</p>
<b>preconnect</b>	<p>La cláusula PRECONNECT indica al agente de usuario que se conecte a un servidor, es decir, que resuelva el DNS y realice la preconexión con los protocolos TCP y TLS, si procede.</p> <p>Cabe destacar que el uso de esta cláusula puede llevar varios cientos de milisegundos, por lo que habrá que tenerlo en cuenta a la hora de realizar la primera petición. No obstante, podemos afirmar que solicitar la exploración previa a través de este método minimizará bastante el tiempo y agilizará la carga de la página.</p> <p>Al igual que DNS-PREFETCH, es especialmente útil cuando se desean cargar archivos desde fuentes externas como pueda ser una fuente vectorial de Google Fonts, un script de un CDN o un JSON desde una API.</p> <p><b>NOTA</b></p> <p>El uso excesivo de esta cláusula puede provocar pérdidas sustanciales en el rendimiento global, por lo que no se recomienda usarlo más de 4 o 6 veces por página.</p>
<b>prerender</b>	<p>La cláusula PRERENDER indica al agente de usuario que se renderice el recurso en segundo plano. Esto puede ser una buena idea cuándo se está seguro de que, el usuario, realizará alguna acción que requiera esta precarga.</p> <p>El uso de PRERENDER puede aumentar el rendimiento hasta un 70%, no obstante, este tipo de procesamiento es muy costoso, tanto a nivel de memoria, como a nivel de tráfico</p> <p><b>NOTA</b></p> <p>A agosto de 2020, esta cláusula no está del todo soportada por Chrome y, Firefox y Safari, no la soportan, por lo que su utilización no está recomendada.</p>

Con respecto a su posible personalización, permite, entre otras cosas, establecer el idioma en el que está escrito a través del atributo HREFLANG y el dispositivo o medio para el que está optimizado mediante el atributo MEDIA, que habitualmente es ALL, SCREEN o PRINT.

## Elemento meta

Los metadatos se pueden definir como datos acerca de los datos. Es como una información que nos proporciona datos clave sobre el contenido que va a ser representado.

Por tanto, podríamos decir que los metadatos proporcionan información a los robots y usuarios que lo necesiten sobre el documento actual. No obstante, esta información puede ser de muy diversa índole, desde información referente a la apariencia, hasta datos que pueden ser utilizados por los agentes de usuario.

Dicho esto, el elemento META especifica una información sobre los datos que se encuentran dentro del actual documento HTML. Un ejemplo podría ser:

```
<meta name="keywords" content="HTML5, HTML, XHTML">
```

El elemento META tiene varios atributos que permiten personalizar, entre otras cosas, la codificación de caracteres, respuestas al documento o el esquema que se debe emplear, no obstante, se verá con mucho más detalle en otro capítulo más adelante.

## Elemento style

Permite definir información de estilo a través de selectores y reglas CSS. Un ejemplo podría ser:

```
<style media="screen" type="text/css">
html{
font-family: Arial, sans-serif;
font-size: 14px;
font-style: normal;
}
</style>
```

El elemento STYLE tiene dos atributos, el tributo MEDIA y el atributo TYPE. El atributo MEDIA, permite especificar el dispositivo o medio para el que está optimizado el recurso y el atributo TYPE, especifica el tipo de medio o dispositivo, sin embargo, actualmente sólo admite el valor expuesto en el ejemplo, es decir, TEXT/CSS.

## Elemento script

HTML puede ser de gran ayuda cuando se trata de describir el contenido, sin embargo, la inmensa mayoría de veces se suele requerir de una funcionalidad que no nos proporciona el lenguaje. Sirva como ejemplo que, si lo que se desea es saber si el valor de un campo de entrada es válido, se debe recurrir a algún fragmento de código externo en lenguaje script para poder realizar las verificaciones pertinentes.

Dicho esto, el elemento SCRIPT permite insertar un código, o fragmento de código, ejecutable dentro de un documento HTML.

```
<script src="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=7ca17def877aec14cofc1d8c98038e29"></script>
```

# ESTRUCTURA DEL DOCUMENTO

Antes de entrar en materia, quisiera hacer un inciso sobre qué es la semántica web y para qué sirve.

La semántica web es un conjunto de recomendaciones y estándares desarrolladas por la W3C (World Wide Web Consortium) que están pensadas para hacer que los datos se vuelvan más legibles. Se basa en la idea de que los metadatos son y pueden ser semánticos y ontológicos, es decir, que los elementos que conforman las páginas pueden aportar un significado con carácter formal y legible.

El objetivo de la semántica web es mejorar el intercambio de información entre sistemas a través de agentes inteligentes. En otras palabras, que los sistemas y programas informáticos puedan encontrar información sin la intervención del factor humano.

Una de las formas para conseguir este objetivo es establecer una jerarquía con una estructuración y organización específica y, en eso, entra en juego el HTML5.

## Elemento article

Especifica un contenido que, habitualmente, es considerado como una entidad independiente o autónoma que cobra sentido por sí sola sin depender de los demás elementos colindantes. Un ejemplo podría ser:

```
<article>
<header>
<h2>HTML5</h2>
</header>
<div>
<p> El lenguaje HTML5 (HyperText Markup Language Versión 5) es un lenguaje de marcado de hipertexto que está vigente desde el año 2014 y puede ser utilizado para...</p>
</div>
<footer>
<a href="html-usos.html">Seguir leyendo</a>
</footer>
</article>
```

El elemento ARTICLE es uno los elementos clave para el posicionamiento SEO, y es de vital importancia para la accesibilidad y semántica web. Por ello, cuando se declara este elemento, se debe especificar, al menos, una etiqueta HEADER con el título y una etiqueta FOOTER con las posibles acciones, si procede.

Además, también es importante tener claro que ARTICLE no puede contener elementos SECTION contenidos en él puesto que, de lo contrario, se perdería el valor de entidad única, independiente y significativa.

## Elemento aside

Especifica un contenido que, habitualmente, es considerado como una entidad independiente a todo elemento colindante. Su uso está especialmente arraigado para listas u opciones de navegación, nubes de etiquetas y menús Off Canvas, aunque se puede utilizar para cualquier cometido mientras no se pierda su significado u ontología. Un ejemplo podría ser:

```
<p>Descripción de un contenido</p>
<aside>
<h3>Artículos relacionados</h3>
<ul>
<li><a href="#">Artículo 1</a></li>
<li><a href="#">Artículo 2</a></li>
<li><a href="#">Artículo 3</a></li>
</ul>
</aside>
```

## Elemento div

El elemento DIV especifica que el contenido que se va a representar es una división o sección.

Aunque el uso de la etiqueta DIV no tiene restricciones, su aplicación está más pensada para realizar divisiones que no tengan, o tengan poco, valor semántico, es decir, su uso debe deberse darse cuando no se puedan utilizar

elementos de mayor significado como puedan ser MAIN, NAV, SECTION, ARTICLE, HEADER o FOOTER.

```
<div>
<p>Esto puede ser un texto descriptivo sobre HTML5</p>

</div>
```

## Elemento footer

El elemento FOOTER puede especificar un pie de página o un pie de artículo, dependiendo de donde se declare. Si es un descendiente directo del elemento ARTICLE, se considerará pie de artículo. Si es un descendiente directo del elemento BODY, se considerará pie de página o documento.

El elemento FOOTER sólo puede ser declarado una única vez por artículo y por página, sin embargo, dependiendo de dónde se defina, su intencionalidad es muy diferente.

Si está dentro de una estructura ARTICLE, lo habitual es que contenga las acciones asociadas o en relación con el artículo que se está describiendo, pero, si es un pie de página, lo habitual es que contenga los datos de contacto, acceso a documentos importantes como la Política de Privacidad, Términos de uso o la Declaración de cookies, información de copyright, los enlaces hacia sus redes sociales o, incluso, otras acciones relacionadas con el contexto de la web y la empresa u organización como apuntarse a la newsletter.

Un ejemplo podría ser:

```
<footer>
<ul>
<li>Copyright ©2020</li>
<li>Política de Privacidad</li>
<li>Versión para móviles</li>
</ul>
</footer>
```

## Elemento header

El elemento HEADER puede especificar una cabecera de página o una cabecera de artículo, dependiendo de donde se declare. Si es un descendiente directo del elemento ARTICLE, se considerará cabecera de artículo. Si es un descendiente directo del elemento BODY, se considerará cabecera de página o documento.

El elemento HEADER sólo puede ser declarado una única vez por artículo y por página, sin embargo, dependiendo de dónde se defina, su intencionalidad es muy diferente.

Si está dentro de una estructura ARTICLE, lo habitual es que contenga el título del artículo que se está describiendo, pero, si es una cabecera de página, lo habitual es que contenga el logo de la empresa u organización, el menú principal de navegación, el acceso a la zona privada y registro, los enlaces hacia sus redes sociales o, incluso, otras acciones relacionadas con el contexto de la web y la empresa u organización de primer orden como un buscador.

También, es habitual encontrar en la definición de cabecera, la declaración de elementos de cabecera H1...H6. Estos elementos, bien utilizados, aportan al documento un orden de jerarquía y valor semántico.

Un ejemplo podría ser:

```
<article>
<h2>HTML5</h2>
<div>

</div>
<footer>
<a href="html-usos.html">Acceder a HTML5</a>
</footer>
</article>
```

## Elementos h1..h6

Los elementos H1...H6 especifican diferentes niveles de encabezado. El más relevante en la jerarquía o de mayor peso es H1 (al cual se le suele atribuir el título del documento) y, el menos relevante en la jerarquía o de menor peso es H6. Un ejemplo podría ser:

```
<h1>Esto es un encabezado de nivel 1</h1>
<h2>Esto es un encabezado de nivel 2</h2>
<h3>Esto es un encabezado de nivel 3</h3>
<h4>Esto es un encabezado de nivel 4</h4>
<h5>Esto es un encabezado de nivel 5</h5>
<h6>Esto es un encabezado de nivel 6</h6>
```

Los elementos H1...H6 pueden ser elementos clave para el posicionamiento SEO, y son de vital importancia para la accesibilidad y semántica web. Es muy importante que no se especifique más de un encabezado de primer nivel y que no se realicen saltos en la jerarquía por temas de apariencia o gusto, es decir, después de un H1 siempre debe ir un H2, después de un H2 siempre debe ir un H3, y así sucesivamente.

También, es importante matizar que, la utilización de elementos de jerarquía dentro de las etiquetas HEADER puede ser una técnica muy interesante para definir el contenido y reforzar su importancia y significado.

## Elemento hr

El elemento HR especifica que lo que se va a representar es una línea horizontal separadora. En el pasado, se solía recurrir a este elemento para realizar un cambio de contexto, sin embargo, cuando apareció HTML5, se perdió su uso pasando a ser un elemento puramente decorativo.

Cabe destacar que, por defecto, muchos agentes de usuario añaden saltos de línea y otros estilos a este elemento, sin embargo, esto es posible subsanarlo fácilmente a través de CSS, como ya veremos.

```
<p>Párrafo 1</p>
<hr />
<p>Párrafo 2</p>
```

## Elemento hgroup

El elemento HGROUP especifica un bloque de encabezado de sección que representa la estructura del documento HTML. Habitualmente, este elemento se utiliza para agrupar encabezados consecutivos a modo de subtítulos o eslóganes. Un ejemplo podría ser:

```
<hgroup>
<h1>Fast and Furious 4</h1>
<h2>Aún más rápido</h2>
</hgroup>
```

## Elemento main

El elemento MAIN especifica el bloque de contenido principal del documento.

La norma para utilizar correctamente este elemento es que:

- Sólo puede haber un elemento MAIN en todo el documento,
- No puede ser un descendiente de ASIDE, SECTION, ARTICLE, HEADER, FOOTER o NAV.
- **Salvo excepciones, no debe incluir secciones laterales, cabeceras de página o pies de página, menús de navegación principales, formularios de búsqueda ni ningún elemento que, por definición o contexto, deban estar fuera de la sección principal del documento.**

Un ejemplo podría ser:

```
<main>
<h1>CSS</h1>
<p>CSS es un lenguaje de marcado para proveer estilos al contenido.</p>
<section>
<article>
<h2>El Sistema Solar</h2>
<p>...</p>
</article>
<article>
<h2>La nebulosa de Orión</h2>
<p>...</p>
</article>
</section>
</main>
```

## Elemento nav

El elemento NAV especifica un conjunto de enlaces de navegación. Este conjunto de enlaces debe estar destinado únicamente para el bloque principal, es decir, no se debe usar el elemento NAV para acciones, botones o enlaces que no pertenezcan al menú principal de navegación. Un ejemplo podría ser:

```
<nav>
<ul class="nav navbar-nav navbar-right">
<li><a href="#home">Inicio</a></li>
<li><a href="#about">Acerca de Nosotros</a></li>
<li><a href="#features">Servicios</a></li>
<li><a href="#portfolio">Portfolio</a></li>
<li><a href="#blog">Blog</a></li>
<li><a href="#support">Contactar</a></li>
<li><a href="javascript:showSearchLayer()">Buscar</a></li>
</ul>
</nav>
```

Cabe destacar que, la utilización correcta de este elemento es importante para la usabilidad web, la semántica web, el posicionamiento SEO y, especialmente, para la accesibilidad web.

La razón de su importancia en accesibilidad es que las herramientas de asistencia, como los lectores de pantalla, usan este elemento para determinar si omitir o no la representación inicial del contenido del documento.

## Elemento section

El elemento SECTION especifica un contenido que, habitualmente, contiene entidades independientes como artículos y tiene una temática definida.

Al igual que sucede con el elemento ARTICLE, el elemento SECTION suele llevar asociado un elemento de encabezado por su naturaleza semántica, aunque no es una cualidad requerida. Un ejemplo podría ser:

```
<section>
<article>
<h3>Atmósfera de Mercurio</h3>
<p>La atmósfera de Mercurio contiene un 31.7% de Potasio, un 24.9% de Sodio, un 9.5% de Oxígeno atómico, un 7.0% de Argón, un 5.9% de Helio, un 5.6% de Oxígeno molecular, un 5.2% de Nitrógeno, un 3.6% de Dióxido de carbono, un 3.4% de Agua y un 3.2% de Hidrógeno.</p>
</article>
</section>
```

El elemento SECTION es otro los elementos clave para el posicionamiento SEO, y es de vital importancia para la accesibilidad y semántica web. Por ello, cuando se declara este elemento, se debe especificar, al menos, una etiqueta H1...H6 con el título.

Además, también es importante tener claro que SECTION puede contener varios elementos SECTION a su vez, o tener varios elementos ARTICLE, pero un elemento ARTICLE no puede contener elementos SECTION contenidos en él.

## FORMATADO DEL TEXTO

### Elemento abbr

El elemento ABBR especifica que el contenido que se va a representar es una abreviatura. Aunque una abreviatura no es un acrónimo, es frecuente confundir el uso de esta etiqueta con la etiqueta ACRONYM. La etiqueta ABBR sólo debe usarse para aquellos contenidos en los que se quiere destacar o indicar que se trata de una abreviatura y no de un acrónimo.

```
<abbr title="Cascading Style Sheets">CSS</abbr>
```

El elemento ABBR admite varios atributos, sin embargo, el único obligatorio es el atributo TITLE que indica el significado de dicha abreviatura.

### Elemento acronym

El elemento ACRONYM especifica que el contenido que se va a representar es un acrónimo.

Aunque su uso no está aprobado para HTML5, se debe aclarar que, un acrónimo es una expresión resultado de la unión de las iniciales de varias palabras, por lo que no es una abreviatura. Por ello, la etiqueta ACRONYM sólo debe usarse para aquellos contenidos en los que se quiere destacar o indicar que se trata de un acrónimo y no de una abreviatura.

```
<acronym title="Para tu información">PTI</acronym>
```

El elemento ACRONYM admite varios atributos, sin embargo, el único obligatorio es el atributo TITLE que indica el significado de dicho acrónimo.

### Elemento address

Especifica una información de contacto para el documento actual. Un ejemplo podría ser:

```
<address>
Escrito por Pablo E. Fernández
Visita <a href="https://ejemplo.com">Ejemplo.com</a>
Castellana 58, local
28046 Madrid
```

España  
</address>

## Elementos b y strong

Los elementos B y STRONG especifican que el contenido que se va a representar debe mostrarse en negrita.

Esto es un texto normal.

```
<br>
<b>Este, sin embargo, es un texto en negrita</b>
<br>
<strong>Al igual que este.</strong>
```

## Elemento blockquote

El elemento BLOCKQUOTE especifica que el contenido que se va a representar es una cita a modo de bloque. Esto es, el contenido se representará agregando un salto de línea y estilos preprogramados.

```
<blockquote cite="https://blog.com/einstein">
    Hay dos cosas infinitas, el Universo y la estupidez humana
</blockquote>
```

Aunque el elemento BLOCKQUOTE puede utilizarse para indicar citas cortas, lo aconsejable es usarlo, únicamente, para citas largas, es decir, lo que viene siendo un párrafo o unas líneas.

El elemento BLOCKQUOTE admite varios atributos, sin embargo, el único obligatorio es el atributo CITE que indica el enlace de la fuente.

## Elemento q

El elemento Q especifica que el contenido que se va a representar es una cita a modo de línea. Esto es, el contenido se representará sin agregar un salto de línea, aunque se le asignarán unos estilos preprogramados.

```
<q cite="https://blog.com/einstein">
    Hay dos cosas infinitas, el Universo y la estupidez humana
</q>
```

Aunque el elemento Q puede utilizarse para indicar citas largas, lo aconsejable es usarlo sólo para citas cortas.

El elemento Q admite varios atributos, sin embargo, el único obligatorio es el atributo CITE que indica el enlace de la fuente.

## Elemento cite

El elemento CITE especifica que el contenido que se va a representar es una cita a modo de título o encabezado. Esto es, el contenido se debe utilizar para citas de tipo nombre de un autor, de un libro, de una canción o película, de un cuadro o fotografía, de una escultura, etcétera.

```
<p>
<cite>
    Hay dos cosas infinitas, el Universo y la estupidez humana
</cite>, dicho por Albert Einstein
</p>
```

Cabe destacar que, el elemento CITE nunca debe contener el nombre del autor, la fecha, ni ningún dato que no sea la cita en sí. Hacerlo, puede provocar una mala indexación en los motores de búsqueda puesto que se considera una violación del estándar.

## Elemento code

El elemento CODE especifica que el contenido que se va a representar es un fragmento de código.

```
<code>
<script type="text/javascript">
document.querySelector("body").style.fontFamily = "Arial";
document.querySelector("body").style.fontSize = "14px";
</script>
</code>
```

Cabe destacar que, el elemento CODE suele insertarse o colocarse dentro de un elemento PRE para conseguir que se respeten los espacios en blanco y los saltos de línea.

## Elementos ins y del

Los elementos INS y DEL especifican que el contenido que se va a representar ha sufrido una alteración que afecta a un texto, o parte de un texto, anteriormente escrito.

```
<p>
El cometa <del>C/2020 F3</del> <ins>Neowise</ins>, descubierto ...
</p>
```

Ambos elementos, INS y DEL, admiten los atributos CITE y DATETIME. El primero, especifica la URL dónde se explica la razón de porqué se ha realizado el cambio y, el segundo, especifica cuándo se realizó dicho cambio.

```
<p>
El cometa <del>C/2020 F3</del>
<ins cite="https://universes.com/neowise#update"
datetime="2020-07-14">Neowise</ins>, descubierto ...
</p>
```

## Elemento em

El elemento EM especifica que el contenido que se va a representar debe aparecer enfatizado. Normalmente, este énfasis suele ser el resultado de aplicar un estilo en cursiva, por lo que se puede confundir con la etiqueta I.

```
<p>
Este texto no tiene énfasis,
<em>pero este texto sí está con énfasis</em>
</p>
```

## Elemento i

El elemento I especifica que el contenido que se va a representar debe aparecer en cursiva. Cabe destacar que, este estilo cursivo puede confundirse con el resultado de la aplicación de la etiqueta EM.

```
<p>
Este texto no está en cursiva,
<i>pero este texto sí está en cursiva o en formato itálico</i>
</p>
```

## Elemento pre

El elemento PRE especifica que el contenido que se va a representar es un texto preformatado. En general, este elemento se suele representar con una fuente Courier o Monospace y conserva todos los espacios y saltos de línea.

```
<pre>
<p>
```

Los espacios repetidos y  
Saltos de línea de este elemento se muestran tal cuál!  
</p>  
</pre>

## Elemento sub

El elemento SUB especifica que el contenido que se va a representar debe aparecer como subíndice, es decir, por debajo de la línea normal y en una fuente de menor tamaño.

```
<p>La fórmula del agua es H<sub>2</sub>O</p>
```

## Elemento sup

El elemento SUP especifica que el contenido que se va a representar debe aparecer como superíndice, es decir, por encima de la línea normal y en una fuente de menor tamaño.

```
<p>E = MC<sup>2</sup></p>
```

## LISTAS

### Elemento UL

El elemento UL especifica que el contenido que se va a representar es una lista desordenada, en general, representada a través de unas viñetas.

Los elementos de esta lista desordenada se definen mediante el elemento LI, el cual representa el contenido útil del elemento UL.

```
<ul style="list-style-type: disc">
  <li>
    Planetas del Sistema Solar
    <ul>
      <li>Mercurio</li>
      <li>Venus</li>
      <li>La Tierra</li>
      <li>Marte</li>
      <li>Júpiter</li>
      <li>Saturno</li>
      <li>Urano</li>
      <li>Neptuno</li>
      <li>Phattie (hipotético planeta helado)</li>
    </ul>
  </li>
  <li>
    Planetas enanos del Sistema Solar
    <ul>
      <li>Eris</li>
      <li>Plutón</li>
      <li>Makemake</li>
      <li>Haumea</li>
      <li>Ceres</li>
    </ul>
  </li>
</ul>
```

Como se puede observar en el ejemplo, el elemento UL permite crear listas anidadas de elementos. La forma de realizarlo es declarar un nuevo elemento UL dentro de un elemento LI.

Los estilos de las listas desordenadas pueden definirse a través de la propiedad de estilo LIST-STYLE-TYPE. Entre los múltiples valores, los más utilizados son:

Propiedad	Significado
<b>circle</b>	Especifica que las viñetas deben ser un círculo (○).
<b>disc</b>	Especifica que las viñetas deben ser un círculo relleno (●).
<b>inherit</b>	Especifica que utilicen las viñetas por defecto.
<b>none</b>	Especifica que las viñetas no se deben mostrar.
<b>square</b>	Especifica que las viñetas deben ser un cuadrado (■).

## Elemento OL

El elemento OL especifica que el contenido que se va a representar es una lista ordenada, en general, representada a través de una lista de valores numéricos o letras.

Los elementos de esta lista ordenada se definen mediante el elemento LI, el cual representa el contenido útil del elemento OL.

```
<ol style="list-style-type: decimal">
<li>...</li>
<li>
Satélites de Plutón
<ol>
<li>Nix</li>
<li>Caronte</li>
<li>Hydra</li>
</ol>
<li>...</li>
</ol>
```

El elemento OL permite crear listas anidadas de elementos. La forma de realizarlo es declarar un nuevo elemento OL dentro de un elemento LI.

Los estilos de las listas desordenadas pueden definirse a través de la propiedad de estilo LIST-STYLE-TYPE. Entre los múltiples valores, los más utilizados son:

Propiedad	Significado
<b>decimal</b>	Especifica que la numeración es decimal sin ceros a la izquierda (1, 2).
<b>decimal-leading-zero</b>	Especifica que la numeración es decimal con ceros a la izquierda (01, 02).
<b>lower-alpha</b>	Especifica que la numeración es alfabética en minúsculas (a, b).
<b>lower-roman</b>	Especifica que la numeración es romana en minúsculas (i, ii).
<b>upper-alpha</b>	Especifica que la numeración es alfabética en mayúsculas (A, B).
<b>upper-roman</b>	Especifica que la numeración es romana en mayúsculas (I, II).

## Elemento DL

El elemento DL especifica que el contenido que se va a representar es una lista de definición, también conocidas como descriptiva o dinámica.

Los elementos de esta lista descriptiva se definen a través de los elementos DT y DD, que representan el grupo de terminología y su descripción respectivamente.

```
<h4>Términos de astronomía</h4>
<dl>
<dt>Planeta</dt>
<dd>Es aquel cuerpo celeste que orbita alrededor del Sol, posee una masa como para que su propia gravedad domine las fuerzas presentes como cuerpo rígido, lo que implica una forma eférica determinada por el equilibrio hidrostático y es claramente dominante en su vecindad, habiendo limpiado su órbita de cuerpos similares a él.</dd>
<dt>Planeta enano</dt>
<dd> Es aquel cuerpo celeste que orbita alrededor del Sol, posee una masa como para que su propia gravedad domine las fuerzas presentes como cuerpo rígido, lo que implica una forma eférica determinada por el equilibrio hidrostático y no es dominante en su vecindad y no es un satélite de otro planeta o cuerpo estelar.</dd>
</dl>
```

El elemento DL suele ser utilizado para definir glosarios, listas de metadatos o pares de clave-valor. No obstante, puede utilizarse en cualquier otra situación que se requiera puesto que posee un gran valor semántico.

Aunque los elementos están dispuestos de una forma muy diferente a como podríamos estar acostumbrados a trabajar con HTML, la representación se realiza de forma bastante clara, asignando únicamente un margen la izquierda de cada elemento DD. Además, los elementos DD no tienen por qué ser únicos, es decir, a un elemento DT le pueden seguir dos o más elementos DD.

Los estilos de las listas de definición no admiten viñetas o numeraciones como sucede con los elementos UL y OL, no obstante, si son personalizables a través de CSS.

### NOTA

Utilizar las listas de definición para crear sangrías o efectos de jerarquización puede afectar a la usabilidad web, accesibilidad web y/o posicionamiento SEO de manera negativa.

## ENLACES

### Elemento A

El elemento A especifica que el contenido que se va a representar es un hipervínculo que, habitualmente, lanzará una acción a otro lugar del documento actual o a otro documento diferente.

Por defecto, los enlaces se estilizan de la misma forma para ayudar a la accesibilidad y usabilidad web. Es por esta razón que, en general, todos los agentes de usuario suelen mostrar los enlaces no visitados en azul y subrayado, los enlaces visitados en morado y subrayado y, los enlaces activos en rojo y subrayado.

```
<a href="https://www.google.es">Visitar Google España</a>
```

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>download</b>	Especifica que el contenido al que apunta el enlace debe ser descargado. Aunque casi todos los navegadores lo soportan, no es funcional con ningún navegador de Microsoft hasta la versión 18 de Microsoft Edge.
<b>href</b>	Especifica el destino hacia dónde se irá cuando se pulse en el enlace. Si este valor empieza por el símbolo almohadilla, indicará que se desea ir a otra sección del documento actual. De no ser así, indicará la dirección hacia otro documento

	diferente.
<b>hreflang</b>	Especifica el idioma del documento vinculado.
<b>media</b>	Especifica para qué medios y/o dispositivos está optimizado el documento vinculado.
<b>referrerpolicy</b>	Especifica la cabecera de HTTP que debe incluirse en la petición de envío cuando se haga clic. No es frecuente verlo en los enlaces de HTML, aunque sí en peticiones Ajax.
<b>rel</b>	Especifica la relación existente entre el documento actual y el vinculado. Entre los posibles valores que puede tomar, los más frecuentes son NOREFERRER, para indicar que no se envíe ningún encabezado, NOFOLLOW, para indicar que el enlace no sea rastreado por los crawlers y SEARCH, para indicar que el documento es una página de búsqueda.
<b>target</b>	Especifica dónde se abrirá el vínculo. Entre los posibles valores que puede tomar, los más frecuentes son _BLANK, para indicar que se abra en una nueva pestaña, _SELF, para indicar que se abra en la misma pestaña y _TOP, para que se abra en el primer elemento BODY de la ventana.
<b>type</b>	Especifica el tipo de medio, antes conocido como MIME type, del documento vinculado. Los posibles valores que puede tomar este atributo están descritos en la página oficial de IANA, en la URL <a href="http://www.iana.org/assignments/media-types/">http://www.iana.org/assignments/media-types/</a> .

Cabe destacar que, si el atributo HREF no está presente, los atributos DOWNLOAD, HREFLANG, MEDIA, REL, TARGET y TYPE no tendrán ningún efecto y serán ignorados.

## PRÁCTICA 1: ESTRUCTURA BÁSICA DE UNA PÁGINA HTML5

### Código del ejemplo

<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-01>



### Objetivo de la práctica

Obtener una idea aproximada de la estructura básica de una web y los elementos que contiene.

### Resolución

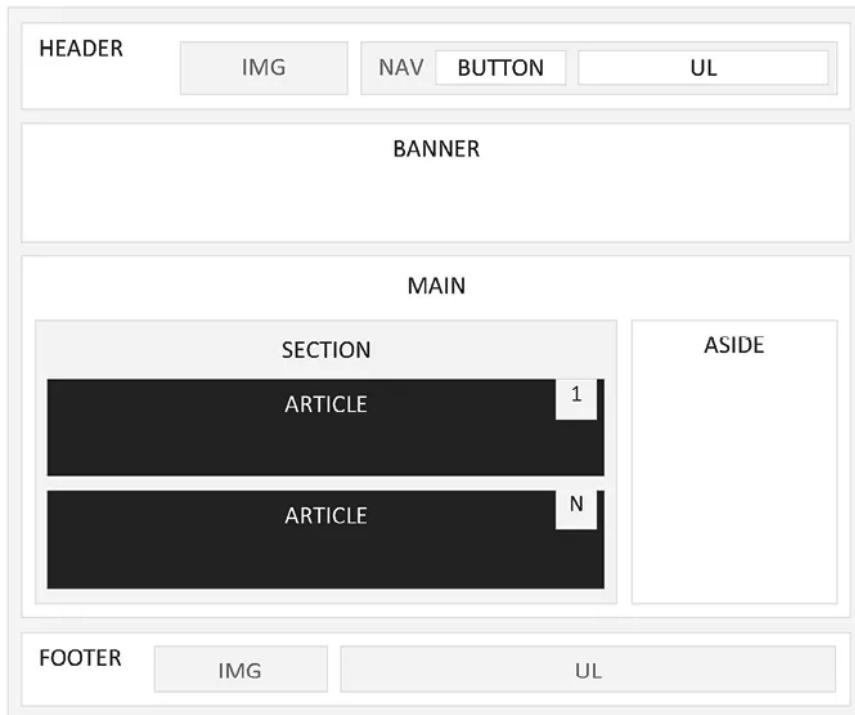
Como hemos dicho anteriormente, para definir un documento HTML5, lo primero que se ha de hacer es definir su tipo a través de la declaración <!DOCTYPE> y la raíz o apertura del mismo, incluyendo el idioma.

Acto seguido, lo que se deberá declarar son las estructuras HEAD, que define la sección de cabecera con todos los datos que se utilizan e identifican el documento, y la estructura BODY, que define todo el contenido que se le va a presentar al usuario.

En el HEAD, definiremos el título del documento, la codificación de caracteres y los metadatos de descripción y VIEWPORT. Además, insertaremos la fuente Roboto de Google Fonts y la referencia a la hoja de estilos, aunque, ahora mismo, estará vacía hasta que lleguemos a la práctica 3 (en donde se proporcionarán los primeros estilos a una página web, después de haber visto el capítulo introductorio de CSS3).

En lo referente al BODY, pueden darse varios tipos de estructura, pero en términos generales, se podría afirmar que un documento HTML5 genérico suele contener:

- Una estructura de cabecera etiquetada con el elemento HEADER que contiene el logotipo identificativo y el menú principal de navegación.
- Una sección identificada como espacio publicitario definida a través de un elemento DIV para mostrar uno o varios productos o servicios a destacar.
- Un contenido principal que suele englobar un elemento MAIN y que contiene una o varias secciones SECTION con uno o varios elementos ARTICLE y, si procede, un elemento ASIDE que representa un acceso a los contenidos relacionados a través de una barra lateral.
- Un FOOTER o pie de página con el logotipo y los elementos principales de finalización como es el identificador de copyright, aviso legal, política de privacidad y/o contactar.



Como se ha mencionado antes, la estructura expuesta en el código anterior no es la única posibilidad que cumple los requisitos mínimos de usabilidad web y semántica web, sin embargo, sí que podemos afirmar que es una de las más extendidas.

## PRÁCTICA 2: ESTRUCTURA DE “UNIVERSES”

### Código del ejemplo

<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-02>



## Objetivo de la práctica

Obtener una idea de lo que supone crear una página de inicio para una web. En esta ocasión, el formato elegido de la web se corresponde con la página de inicio de un blog.

## Resultado

UniversES

UniversES

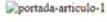
- UniversES
  - Inicio
  - Dónde estamos
  - Novedades
  - Contactar
  - Acceder

UniversES

Todo el conocimiento del universo a tu alcance

**Inicio**

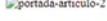
**Lore ipsum**

portada-articulo-1

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Seguir leyendo

**Dolor sit amet**

portada-articulo-2

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Seguir leyendo

**Consectetur adipiscing elit**

## Resolución

Si nos fijamos en el resultado de la práctica, podemos apreciaremos que sólo es un montón de texto con enlaces y sin formato alguno, pero eso cambiará, y mucho, a partir del próximo capítulo.

Dicho esto, ahora que sabemos cómo es una estructura típica de HTML a grandes rasgos, lo que vamos a hacer es crear la estructura para la página de inicio de una web con contenidos sobre el Universo.

Como en el caso anterior, lo primero que se debe hacer es definir el tipo de documento a través de la declaración `<!DOCTYPE>` e indicar el idioma utilizado.

Acto seguido, se deberá definir la sección HEAD del documento dónde se especificarán los principales metadatos de la página (los cuales se verán en detalle más adelante), las fuentes vectoriales de FontAwesome y la hoja de estilos que contiene las reglas aplicables al documento actual.

Las fuentes vectoriales como FontAwesome, pueden cargarse desde cualquier CDN, por ejemplo, <https://cdnjs.com/>. Cuando se accede a esta web, aparece un buscador en donde se puede escribir lo que se desea buscar, en este caso “font-awesome” sin comillas. Después, entre los resultados seleccionamos el enlace de la versión deseada y lo copiamos a la propiedad HREF del elemento LINK deseado.

En lo referente al archivo STYLES.CSS, cabe mencionar que, como todavía no hemos entrado en ese capítulo, es posible que el navegador muestre un error porque no encuentra la hoja de estilos definida por la declaración LINK, pero no hay que preocuparse demasiado porque se verán en detalle en el próximo capítulo.

Una vez que tengamos definido la sección HEAD del documento, podremos definir la estructura del contenido del documento. Como en la práctica anterior, definiremos una sección de cabecera que contendrá un elemento que reflejará el nombre de la empresa u organización y un menú de navegación con un botón y una lista de opciones.

Como queremos que sea similar a cómo son las páginas web actuales, le introduciremos un banner debajo del menú de navegación. Esta sección, de momento, sólo contendrá el nombre de la empresa u organización y un texto a modo

de slogan.

Una vez definida la cabecera visual de la página, es hora de definir el contenido principal del documento. Este contenido principal se caracterizará por tener una sección MAIN que contendrá una sección SECTION que, a su vez, contendrá un elemento de cabecera H1 y cuatro elementos ARTICLE, cada uno de ellos con un HEADER y un DIV.

En los elementos HEADER, se definirán los títulos de cada artículo a través de un elemento de cabecera H2 y una imagen a modo de portada. En los elementos DIV, se definirán las entradillas para cada artículo y un elemento A, que hará de enlace para acceder al contenido completo.

No obstante, como toda web que se diseña a modo de blog, suele tener una sección lateral en dónde se ofrecen acceso al histórico por mes, entre otras funcionalidades, lo que haremos es añadir una sección ASIDE con dos listas. Una lista con los últimos comentarios de los usuarios y otra con los enlaces a los artículos de los últimos meses agrupados por mes.

Sólo nos resta definir el pie de página. En esta sección tan característica es dónde se deberá proporcionar acceso a los principales datos informativos como son los datos del copyright o los documentos sobre aviso legal y la política de privacidad.

No obstante, también es muy frecuente insertar en el pie de página, cosas como el logotipo de la empresa u organización, una forma de contactar a través de un enlace o formulario, los principales iconos con un vínculo a nuestras redes sociales y otros datos de interés.

Por último, quiero mencionar que, aunque en este curso no se va a ver, en ocasiones introduciremos código en JavaScript para hacer que las páginas de ejemplo funcionen debidamente. En esta ocasión añadiremos unas funcionalidades para hacer que, el ícono del menú en modo receptivo cambie en función de si está abierto o no y para hacer que la barra superior de navegación se quede fija en pantalla cuando se supera la altura del banner.

# 3

## CSS3

El lenguaje CSS (Cascading Style Sheets) es un lenguaje de marcado de presentación dedicado a la elaboración de páginas web. Fue definido por primera vez en 1994 y, poco después, fue añadido al grupo de trabajo de la W3C como parte del proceso de desarrollo y estandarización.

El crecimiento de especificación CSS ha sido francamente irregular con respecto a otras evoluciones de lenguajes de marcado. De hecho, sólo hay que investigar un poco para ver que en 1998 se publicó la recomendación 2.1 y, poco después, en 1999, ya apareció el primer borrador de la versión actual de CSS. Desde entonces, ha ido evolucionando hasta la especificación que es la actualmente denominamos CSS3.

El lenguaje de marcado CSS3 está pensado para separar el contenido y estructura de la presentación en un documento HTML, XML o XHTML. Esta separación permite, además, mejorar la accesibilidad y usabilidad de los documentos porque proporciona una mayor flexibilidad, una mejor reutilización de código y una reducción de la complejidad gracias, entre otras cosas, a que se evita la repetición de código.

Entre sus características iniciales podemos destacar que es un lenguaje sencillo que permite diferentes métodos de renderizado y que todo se realiza a través de reglas que se aplican en función de unos selectores previamente definidos.

## SOPORTE A LOS NAVEGADORES

El soporte de CSS3 en los navegadores actuales está más o menos cubierto, no obstante, Internet Explorer sigue siendo el que más problemas da a la hora de hacer que los documentos se comporten de igual forma.

En líneas generales podemos decir que Chrome, Edge, Firefox, Internet Explorer, Opera y Safari son los navegadores más utilizados por los usuarios y que, prácticamente todos, a excepción de Internet Explorer 10 e inferiores, dan soporte a todos los selectores, pseudo-clases y gran parte de las propiedades.

En la dirección [https://www.w3schools.com/cssref/css3\\_browsersupport.asp](https://www.w3schools.com/cssref/css3_browsersupport.asp) es posible encontrar el soporte específico para todas las propiedades CSS de manera individual.

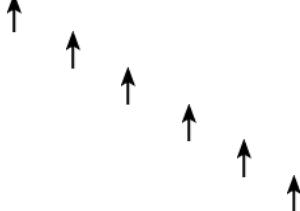
## CÓMO FUNCIONA CSS

Básicamente, el lenguaje CSS es una especificación que define un conjunto de reglas, cada una de ellas, definida a través de un selector y una declaración.

Selector Declaración

-----  
div { background: white; font-family: Arial; font-size: 14px }

Propiedad Valor Propiedad Valor Propiedad Valor



Como se puede apreciar en la ilustración anterior, la regla CSS está compuesta por un selector y una declaración.

El selector, puede ser un nombre de etiqueta HTML, una clase, un identificador, un comodín o, incluso, una combinación de ellos. Su objetivo es localizar el elemento o grupo de elementos dónde se debe aplicar la declaración.

La declaración, está definida a modo de bloque mediante unas llaves que representa el conjunto de pares de propiedad-valor.

Cada uno de estos pares siempre termina con un punto y coma, aunque si es el final de la declaración puede omitirse, como es el caso de la ilustración. El conjunto de estos pares de propiedad-valor es el que definirá el comportamiento

visual del objeto que se haya seleccionado por el selector previamente definido.

## DEFINICIÓN DE SELECTOR Y CLASIFICACIÓN

Como se acaba de comentar, un selector, puede ser un nombre de etiqueta HTML, una clase, un identificador, un comodín o, incluso, una combinación de ellos y se utilizan para localizar o seleccionar el conjunto de elementos HTML que se desea manipular.

Aunque los selectores pueden ser de muy diversos tipos, se puede establecer una clasificación general en la que se dividen en universales, simples, combinados, de atributo, pseudo-clases y pseudo-elementos. A continuación, se describen cada uno de ellos con brevedad ya que, posteriormente, se verán en detalle:

**Los selectores simples** son aquellos que permiten seleccionar los elementos a partir del nombre de una etiqueta, identificador o clase. Por ejemplo:

```
button { background: #fofofo; border: 1px solid #ccc; }
#banner { background: black; }
.form-group { color: #ooo; height: 32px; line-height: 32px; }
```

**Los selectores combinados** son aquellos que permiten seleccionar los elementos a partir de una relación preestablecida entre ellos. Por ejemplo:

```
div p { font-family: Arial; font-size: 14px; }
#banner .title { font-family: Verdana; font-size: 21px; }
```

**Las pseudo-clases** son un tipo especial de selectores que permiten seleccionar los elementos a partir del estado en el que se encuentran y siempre van precedidos del símbolo dos puntos. Por ejemplo:

```
button:hover { font-weight: bold; }
a:active { color: red; }
input:focus { background: #ooo; color: white; }
```

**Los pseudo-elementos** son otro tipo especial de selectores que permiten seleccionar partes concretas de los elementos o del documento y siempre van precedidos del símbolo dos puntos repetido dos veces. Por ejemplo:

```
::selection { background: #003366; color: rgba(255,255,255,1); }
::placeholder { background: #333333; color: lightgray; }
.icon::before { content: "\f249"; font-size: 22px; color: #006699; }
```

**Los selectores de atributo** son aquellos que permiten seleccionar los elementos a partir de sus atributos o propiedades y siempre van declarados a través de corchetes. Por ejemplo:

```
a[href^="https"] { color: rgba(255,255,255,1); }
a[href*="blog"] { color: lightgray; }
a[href$=".php"] { color: #006699; }
```

El **selector universal** permite selección todos los elementos del documento o del elemento padre que se haya definido.

```
* { font-family: Arial; font-size: 14px; }
article * { color: black; }
```

## UNIDADES DE MEDIDA

Existen dos tipos de unidad de medida, las absolutas y las relativas.

## **Unidades absolutas**

Las unidades absolutas o fijas son aquellas que no están relacionadas de ninguna forma ni con el contexto, ni con otra entidad. Entre las diferentes propuestas que ofrece CSS como unidades de medida absolutas tenemos:

### **UNIDAD DE MEDIDA PX**

El término PX hace referencia a los puntos de la pantalla, por lo que cada punto en la pantalla es un píxel. Sus posibles valores pueden ser negativos y positivos, dependiendo de la propiedad y contexto.

### **UNIDAD DE MEDIDA PT**

El término PT hace referencia al tamaño del punto en una pantalla o papel. En lo referente a su equivalencia, un punto se suele traducir a 0.35mm, o lo que es lo mismo, a 1.33 píxeles. Sus posibles valores pueden ser negativos y positivos, dependiendo de la propiedad y contexto.

### **UNIDAD DE MEDIDA IN**

El término IN hace referencia a la longitud del sistema inglés pulgada, es decir, 2.54cm. En lo referente a sus posibles equivalencias, una pulgada se suele traducir en a 96 píxeles o, lo que es lo mismo, 72 puntos. Sus posibles valores pueden ser negativos y positivos, dependiendo de la propiedad y contexto.

### **UNIDADES DE MEDIDA CM, MM Y PC**

Los términos CM, MM y PC hacen referencia las conocidas medidas de longitud. En lo referente a su equivalencia, podríamos decir que una pica es equivalente a 12 puntos, o lo que es lo mismo, 4.23 milímetros.

## **Unidades relativas**

Las unidades relativas son aquellas que toman como punto de referencia otra entidad o contexto. Entre las diferentes propuestas que ofrece CSS como unidades de medida absolutas tenemos:

### **UNIDAD DE MEDIDA %**

El término % hace referencia al porcentaje del marco o contexto actual. Sus valores no tienen por qué ir de cero a cien, ya que también son válidos los valores superiores.

### **UNIDAD DE MEDIDA CH**

El término CH hace referencia a la anchura, en píxeles, del carácter o del elemento padre, esto es, depende del tamaño de la fuente y letra base realizar la equivalencia.

El tamaño que se toma como referencia es la anchura del carácter o, por tanto, si se establece el ancho de un elemento a 80CH, lo que se está indicando es que, será de ancho como 80 caracteres o, todos seguidos sin espacios en blanco.

### **UNIDAD DE MEDIDA EM**

El término EM hace referencia a la anchura, en píxeles, de la letra M mayúscula del elemento padre, esto es, si el elemento padre tiene aplicado un tamaño de fuente de 14PX, 1.2EM será equivalente a 16.8PX.

## **UNIDAD DE MEDIDA EX**

El término EX hace referencia a la anchura, en píxeles, de la letra X minúscula del elemento padre, esto es, depende del tamaño de la fuente y letra base realizar la equivalencia.

El tamaño que se toma como referencia es la anchura de la X minúscula, por tanto, si se establece el ancho de un elemento a 80EX, lo que se está indicando es que, será de ancho como 80 caracteres x, todos seguidos sin espacios en blanco.

## **UNIDAD DE MEDIDA REM**

El término REM hace referencia a la anchura, en píxeles, de la letra M mayúscula del elemento HTML (o raíz), esto es, si el elemento HTML tiene aplicado un tamaño de fuente de 14PX, 1.2REM será equivalente a 16.8PX.

## **UNIDAD DE MEDIDA VMAX**

El término VMAX hace referencia a la centésima parte de la altura o anchura de la ventana gráfica o área útil.

La regla de para saber qué valor se está aplicando es “cuál de los valores de ancho y alto es mayor”, esto es, si la altura es mayor que la anchura, 1VMAX será equivalente a 1VH, pero si la anchura es mayor que la altura, 1VMAX será equivalente a 1VW.

## **UNIDAD DE MEDIDA VMIN**

El término VMIN hace referencia a la centésima parte de la altura o anchura de la ventana gráfica o área útil.

La regla de para saber qué valor se está aplicando es “cuál de los valores de ancho y alto es menor”, esto es, si la altura es menor que la anchura, 1VMIN será equivalente a 1VH, pero si la anchura es menor que la altura, 1VMIN será equivalente a 1VW.

## **UNIDAD DE MEDIDA VH**

El término VH hace referencia a la centésima parte de la altura del viewport, esto es, un 1% de la altura de la ventana gráfica o área útil. Por tanto, si se establece el alto de un elemento a 100VH, lo que se está indicando es que sea el 100% del alto del viewport, o lo que es lo mismo, 100% del alto de la ventana del navegador.

## **UNIDAD DE MEDIDA VW**

El término VW hace referencia a la centésima parte de la anchura del viewport, esto es, un 1% de la anchura de la ventana gráfica o área útil. Por tanto, si se establece el ancho de un elemento a 100VW, lo que se está indicando es que sea el 100% del ancho del viewport, o lo que es lo mismo, 100% del ancho de la ventana del navegador.

## **CODIFICACIÓN DE COLORES**

Los colores en HTML pueden establecerse a través de los nombres preestablecidos por el agente de usuario, mediante codificación RGB, HSL o hexadecimal. Tanto RGB, como HSL, admiten una variación que permite especificar el canal alfa.

### **Codificación RGB y RGBA**

El modelo de color RGB (Red - Green - Blue) es un modelo matemático abstracto que describe cómo realizar la codificación o representación de colores a partir de unos componentes de intensidad sobre los colores primarios.

Dado que su formulación es únicamente a través de rojo, verde, azul, la codificación RGB no presenta la posibilidad de definir un color con transparencia.

El espacio de color RGBA (Red - Green - Blue - Alpha) es una combinación del modelo RGB con un cuarto componente denominado canal de alfa. Este canal alfa es el que posibilita que los colores puedan tener una transparencia.

El valor de los componentes rojo, verde y azul se establecen a partir de un valor binario de 8 bits bajo el sistema decimal, es decir, con valores que van de 0 a 255. Sin embargo, el valor del canal alfa, se establece con un valor en tanto por uno.

```
RGB(0,0,255); /* Color azul al 50% de transparencia */  
RGB(0,0,255, 0.5); /* Color azul al 50% de transparencia */
```

En lo referente a su compatibilidad, el formato RGB y RGBA resulta ser compatible con IE 9+, Firefox 3+, Chrome, Safari 3.1+ y Opera 10+.

En lo referente a su compatibilidad, la entrada de colores en formato es compatible con todos los agentes de usuario conocidos.

## Codificación Hexadecimal

La codificación de colores en hexadecimal funciona exactamente igual que el modelo RGB, es decir, los colores pueden formalizarse o construirse a partir de unos valores de rojo, verde, azul, no obstante, su anotación es diferente.

Cabe destacar que existen algunos navegadores que permiten definir colores en hexadecimal y con transparencia a través de un código de ocho caracteres hexadecimales, en vez de seis. No obstante, este tipo de codificación con transparencia no es compatible ni con Internet Explorer, ni con Microsoft Edge.

El valor de los componentes rojo, verde y azul van de 00 a FF, así como el canal alfa, se establecen a partir de un valor binario de 8 bits bajo el sistema hexadecimal, es decir, con valores que van de 00 a FF.

En general, esta especificación se aplica mediante la concatenación del carácter almohadilla y una cadena de seis caracteres, aunque, también es posible codificar los colores mediante una cadena de longitud tres. Cuando se utiliza este tipo de anotación, el color generado se construye a partir de la duplicación y anidación de cada carácter. Es decir, cuando se especifica un color codificado como oFo, en realidad, se está generando el color ooFFoo.

```
#0000FF; /* Color azul con 6 caracteres */  
#oof /* Color azul con 3 caracteres */
```

En lo referente a su compatibilidad, el formato hexadecimal resulta ser compatible con IE 3+, Firefox, Chrome, Safari y Opera 3.5+.

## Codificación HSL y HSLA

El modelo de color HSL (Hue - Saturation - Lightness) es un modelo matemático abstracto que describe cómo realizar la codificación o representación de colores a partir de unos componentes de matiz, saturación y luminosidad. Dado que su formulación es únicamente a través de estos tres componentes, la codificación HSL no presenta la posibilidad de definir un color con transparencia.

El espacio de color HSLA (Hue - Saturation - Lightness - Alpha) es una combinación del modelo HSL con un cuarto componente denominado canal de alfa. Este canal alfa es el que posibilita que los colores puedan tener una

transparencia.

El componente de matiz se mide en grados, por lo que sus posibles valores van de 0 a 360. El valor 0 equivale al rojo, 120 al verde y 240 al azul.

Los componentes de saturación y luminosidad se establecen en términos porcentaje, es decir, con valores que van de 0 a 100 y, el canal alfa se establece en términos de tanto por uno.

```
HSL(0, 240%, 50%); /* Color azul al 50% de transparencia */  
HSLA(0, 240%, 50%, 0.5); /* Color azul al 50% de transparencia */
```

En lo referente a su compatibilidad, el formato hexadecimal resulta ser compatible con IE 9+, Firefox 3+, Chrome, Safari 3.1+ y Opera 10+.

## PROPIEDADES

A continuación, se muestra una lista con las propiedades de CSS3 agrupadas por funcionalidad, aunque algunas ellas se verán en otros capítulos más adelante.

### Texto, fuentes y tipos de letra

#### PROPIEDAD COLOR

Especifica el color del texto. Esta propiedad admite una larga lista de colores a través de su nombre en inglés o mediante su equivalente en RGB(A), HSL(A) o hexadecimal.

##### Ejemplos:

```
p { color: black; }  
p { color: #000000; }  
p { color: rgb(0, 0, 0); }  
p { color: hsl (0, 0%, 0%); }  
p { color: rgba(0,0,0,0.0); } /* color transparente */  
p { color: transparent; } /* color transparente */
```

#### PROPIEDAD DIRECTION

Dirección de la lectura. Puede obtener los valores LTR y RTL para indicar que el sentido de la lectura es de izquierda a derecha o al revés, respectivamente.

##### Ejemplos:

```
p { direction: rtl; }  
article { direction: ltr; }
```

#### PROPIEDAD FONT-FAMILY

Especifica el espacio entre caracteres. Entre sus posibles valores podemos encontrar:

- **NORMAL**: Indica el valor prefijado por el agente de usuario.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

Si se desea indicar varias fuentes, se puede hacer a través del carácter coma, no obstante, su aplicación será en función de la disponibilidad y de izquierda a derecha, es decir, si la primera no está disponible o no se puede seleccionar, se seguirá intentando con las siguientes.

##### Ejemplos:

```
p { font-family: Arial, sans-serif; }
a { font-family: "Open Sans", sans-serif; } /* Fuente de Google Fonts */
b { font-family: Times New Roman, serif; }
```

## PROPIEDAD FONT-SIZE

Especifica el tamaño del texto. Entre sus posibles valores podemos encontrar:

- **XX-SMALL**: Indica un tamaño de fuente baladí, equivalente a 9PX.
- **X-SMALL**: Indica un tamaño de fuente muy pequeño, equivalente a 10PX.
- **SMALL**: Indica un tamaño de fuente pequeño, equivalente a 13PX.
- **MEDIUM**: Indica un tamaño de fuente medio, equivalente a 16PX. Es el valor por defecto.
- **LARGE**: Indica un tamaño de fuente grande, equivalente a 18PX.
- **X-LARGE**: Indica que el tamaño de fuente muy grande, equivalente a 24PX.
- **XX-LARGE**: Indica un tamaño de fuente enorme, equivalente a 32PX.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

### Ejemplos:

```
html, body { font-size: 14px; }
h1 { font-size: 1.5rem; }
h2 { font-size: large; }
li { font-size: 1em; }
```

## PROPIEDAD FONT-STYLE

Especifica el estilo del texto. Entre sus posibles valores podemos encontrar:

- **NORMAL**: Indica que se muestre con un estilo de fuente normal.
- **ITALIC**: Indica que se muestre con un estilo de letra cursiva.
- **OBLIQUE**: Indica que se muestre con un estilo de fuente oblicuo.

### Ejemplos:

```
p { font-style: normal; }
em { font-style: italic; }
a { font-style: oblique; }
```

## PROPIEDAD FONT-STRETCH

Especifica el nivel de condensación o expansión del texto. Entre sus posibles valores podemos encontrar:

- **ULTRA-CONDENSED**: Indica que el texto sea lo más estrecho posible.
- **EXTRA-CONDENSED**: Indica que el texto sea algo menos estrecho que ULTRA-CONDENSED.
- **NARROW**: Indica que el texto sea un poco menos estrecho que su predecesor.
- **CONDENSED**: Indica que el texto sea un poco menos estrecho que NARROW.
- **SEMI-CONDENSED**: Indica que el texto debe ser un poco menos estrecho que CONDENSED, cercano a NORMAL.
- **NORMAL**: Indica que el texto no se debe estirar ni estrechar. Es el valor por defecto.

- **EXPANDED**: Indica que el texto sea un poco más ancho de lo normal.
- **EXTRA-EXPANDED**: Indica que el texto sea un poco más ancho que EXPANDED.
- **ULTRA-EXPANDED**: Indica que el texto sea lo más ancho posible.

### Ejemplos:

```
p { font-stretch: semi-condensed; }
em { font-stretch: narrow; }
a { font-stretch: ultra-expanded; }
```

### NOTA

Sólo las fuentes que ofrezcan caras adicionales con variaciones de caracteres condensados y/o expandidos podrán hacer uso de esta propiedad.

### PROPIEDAD FONT-VARIANT

Especifica la variación del texto. Esta propiedad presenta múltiples valores, pero el único que, de verdad se utiliza, es **SMALL-CAPS** y **ALL-SMAL-CAPS** para indicar que el texto se muestre con las letras minúsculas convertidas a mayúsculas, pero con un tamaño menor.

### Ejemplos:

```
p { font-variant: small-caps; }
em { font-variant: all-small-caps; }
```

### PROPIEDAD FONT-WEIGHT

Especifica el grosor del texto. Entre sus posibles valores podemos encontrar:

- **NORMAL**: Indica que se muestre la letra con un grosor normal. Es el valor por defecto.
- **BOLD**: Indica que se muestren los caracteres gruesos.
- **100, 200, 300, 400, 500, 600, 700, 800, 900**: Indica que se muestre el grosor de los caracteres en función de estos valores.

Muchas fuentes vectoriales utilizan esta propiedad para definir diferentes pesos o grosores de carácter. En general, el valor 100 se corresponde con un tipo de letra muy fina o delgada, el valor 400 a un tipo NORMAL y 700 o superior a un tipo BOLD.

### Ejemplos:

```
p { font-weight: 100; }
em { font-weight: 400; }
a { font-weight: bold; }
```

### PROPIEDAD LETTER-SPACING

Especifica el espacio entre caracteres que se debe aplicar al texto. Entre sus posibles valores podemos encontrar:

- **NORMAL**: Indica que se muestre la letra con un espacio entre caracteres normal. Es el valor por defecto.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

### Ejemplos:

```
p { letter-spacing: normal; } /* Ejemplo a normal */
```

```
em { letter-spacing: 10px; } /* Ejemplo a 10 px */
```

## PROPIEDAD LINE-HEIGHT

Especifica el espacio entre líneas de un texto. Entre sus posibles valores podemos encontrar:

- **NORMAL**: Indica que se muestre la letra con un espacio entre caracteres normal definido por defecto.
- **[NÚMERO]**: Indica un factor de multiplicación para el tamaño de fuente actual.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

### Ejemplos:

```
p { line-height: normal; }
h1 { line-height: 2; } /* Si font-size es 10px, equivaldrá a 20px */
h2 { line-height: 18px; }
```

## PROPIEDAD SHAPE-OUTSIDE

Especifica cómo se distribuirá el texto alrededor de un elemento flotante, es decir, indica cómo se recolocará el texto alrededor de un elemento que tiene la propiedad FLOAT establecida con un valor diferente a NONE.

Entre sus posibles valores podemos encontrar:

- **CIRCLE**: Indica que el texto debe flotar siguiendo una forma circular. El radio y valores de posición pueden establecerse a través de cualquiera de las unidades de medida estándar de CSS y, adicionalmente, esta función permite el uso de CLOSEST-SIDE y FARTHEST-SIDE que sirven para indicar que el círculo debe llegar hasta el lado más cercano o hasta la esquina más lejana, respectivamente. Su sintaxis es:

```
shape-outside: circle();
shape-outside: circle(radio at posX posY);
```

- **ELLIPSE**: Indica que el texto debe flotar siguiendo una forma elíptica. El radio y valores de posición pueden establecerse a través de cualquiera de las unidades de medida estándar de CSS y, adicionalmente, esta función permite el uso de CLOSEST-SIDE y FARTHEST-SIDE que sirven para indicar que el círculo debe llegar hasta el lado más cercano o hasta la esquina más lejana, respectivamente. Su sintaxis es:

```
shape-outside: ellipse();
shape-outside: ellipse(radio en X Y at posX posY);
```

- **POLYGON**: Indica que el texto debe flotar siguiendo una forma poligonal. La especificación de pares de coordenadas X e Y se deben separar a través de comas y usando las unidades de medida estándar de CSS. Su sintaxis es:

```
shape-outside: polygon(X Y, X Y, ..., X Y);
```

- **INSET**: Indica que el texto debe flotar siguiendo una forma rectangular. Adicionalmente, admite la palabra clave ROUND seguido del valor de redondeo para los valores de la diagonal superior izquierda y diagonal superior derecha que hacen que la forma se vuelva redondeada. Para especificar el valor de las coordenadas X e Y se puede usar cualquiera de las unidades de medida estándar de CSS, separando, esos sí, los valores a través de comas. Su sintaxis es:

```
shape-outside: inset(top right bottom left);
shape-outside: inset(top right bottom left round r1 r2);
```

- **URL:** Indica que el texto debe flotar siguiendo las transparencias de la imagen proporcionada por parámetro. Para especificar el valor de esta propiedad se puede utilizar codificación en Base64 o una URI. Su sintaxis es:

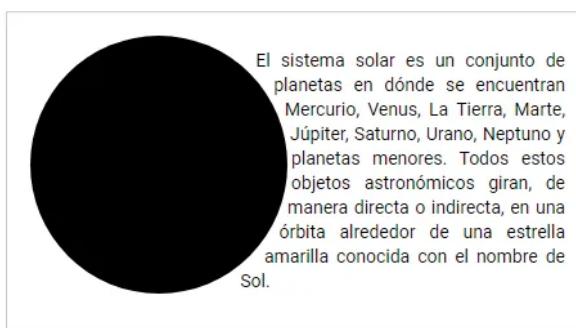
```
shape-outside: url([Base64]);
shape-outside: url([URI]);
```

Aunque las opciones anteriores son las más utilizadas, esta propiedad también permite dar forma a partir de las siguientes palabras clave:

- **BORDER-BOX:** Indica debe flotar por fuera del borde del elemento flotante.
- **CONTENT-BOX:** Indica debe flotar alrededor del contenido del elemento flotante.
- **MARGIN-BOX:** Indica debe flotar por fuera del margen externo del elemento flotante.
- **PADDING-BOX:** Indica debe flotar por fuera del margen interno del elemento flotante.

## Ejemplo gráfico 1

```
<style>
article { border:1px solid #ccc; line-height:1.42; margin:0 auto;
padding:1em; text-align: justify; width: 450px; }
article div{ background: #000000; border-radius: 100%;
float:left; height: 200px; margin: 0.25rem;
shape-outside: circle(); width: 200px; }
</style>
<article>
<div></div>
<p>El sistema solar es un conjunto de planetas en dónde se encuentran Mercurio, Venus, La Tierra, Marte, Júpiter, Saturno, Urano, Neptuno y planetas menores. Todos estos objetos astronómicos giran, de manera directa o indirecta, en una órbita alrededor de una estrella amarilla conocida con el nombre de Sol.</p>
</article>
```



## Ejemplo gráfico 2

```
<style>
article { border:1px solid #ccc; line-height:1.42; margin: 1rem auto;
padding:1em; text-align: justify; width: 450px; }
article img{ float: left; height: 150px; object-fit: contain;
padding: 0 0.8rem; shape-image-threshold: 0.5;
shape-outside: url(https://api.perlego.com/assets/asset?session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=ff8107104cd45co999bcfd5d3566af44); width: auto; }
</style>
<article>

<p>El sistema solar es un conjunto de planetas en dónde se encuentran Mercurio, Venus, La Tierra, Marte, Júpiter, Saturno, Urano, Neptuno y planetas menores. Todos estos objetos astronómicos giran, de manera directa o indirecta, en una órbita alrededor de una estrella amarilla conocida con el nombre de Sol.</p>
</article>
```

</article>



Imagen extraída de [https://es.m.wikibooks.org/wiki/Archivo:Draw\\_Solar\\_System.png](https://es.m.wikibooks.org/wiki/Archivo:Draw_Solar_System.png)

Cabe destacar que, cuando se recurre a este tipo de efectos o presentaciones, también se suele indicar la propiedad **SHAPE-IMAGE-THRESHOLD** y **SHAPE-MARGIN**. La primera, define el rango de valores válidos del canal alfa para extraer la forma y, la segunda, establece un margen alrededor de la forma.

También es importante entender que, no todas las imágenes con transparencias generan el efecto deseado. Esto suele ocurrir porque las imágenes no siempre guardan las transparencias de manera correcta. La forma de solventar este problema suele pasar por un editor de imágenes como PhotoShop, seleccionar todo el contenido externo a la figura de la imagen con la herramienta de varita mágica y eliminar el fragmento de imagen pulsando la tecla suprimir (o la opción de menú Edición / Borrar).

## PROPIEDAD TEXT-ALIGN

Especifica la alineación del texto. Entre sus posibles valores podemos encontrar **LEFT**, **RIGHT**, **CENTER** y **JUSTIFY** que indican que el texto se alinee a la izquierda, a la derecha, de manera centrada o de forma justificada, respectivamente.

Cabe destacar que la alineación justificada no suele ser una buena opción si se desea que la página sea accesible.

### Ejemplos:

```
p { text-align: left; }  
td.number { text-align: right; }
```

## PROPIEDAD TEXT-DECORATION

Especifica la decoración agregada al texto. Aunque esta propiedad presenta múltiples valores, los más recurrentes y frecuentemente utilizados son:

- **UNDERLINE**: Indica que el texto se muestre subrayado por debajo del texto.
- **OVERLINE**: Indica que el texto se muestre subrayado por encima del texto.
- **LINE-THROUGH**: Indica que el texto se muestre tachado.

### Ejemplos:

```
a { text-decoration: underline; }  
h1 { text-decoration: overline; }  
.error { text-decoration: line-through; }
```

## PROPIEDAD TEXT-INDENT

Especifica la sangría que se debe aplicar a la primera línea del texto. Entre sus posibles valores podemos encontrar:

- **[PORCENTAJE]**: Indica que define un tanto por ciento de sangría con respecto al ancho del elemento actual.

- **[VALOR]:** Indica un valor establecido en una de las medidas permitidas de CSS.

Aunque no es frecuente, el valor de la sangría puede ser negativo. Si esto es así, el texto se sangrará a la izquierda en vez de hacia la derecha.

### Ejemplos:

```
li { text-indent: 15px; }
div { text-indent: -1.5em; }
span { text-indent: 6%; }
```

## PROPIEDAD TEXT-SHADOW

Es una propiedad compuesta que agrega una sombra al texto. Su sintaxis es:

```
text-shadow: PosX PosY radio_desenfoque color;
```

Si el valor de la posición horizontal es positivo, la sombra avanzará en sentido hacia la derecha, por lo que, si es negativo, avanzará en sentido hacia la izquierda. Algo similar pasa con el segundo parámetro. Si el valor de la posición vertical es positivo, la sombra avanzará en sentido hacia abajo, por lo que, si es negativo, avanzará en sentido hacia arriba.

Cabe destacar que, salvo excepciones, la propiedad TEXT-SHADOW no se debe utilizar porque, además de dificultar su lectura y disminuir la legibilidad, puede proporcionar una imagen corporativa “desaliñada”.

### Ejemplos:

```
P { text-shadow: 0 0 3px #333; }
div { text-shadow: -1px -1px 5px HSL(0, 0%, 0%); }
span { text-shadow: 1% 1% 5px red; }
```

## PROPIEDAD TEXT-TRANSFORM

Especifica la capitalización del texto. Entre sus posibles valores, los más recurrentes y frecuentemente utilizados son:

- **NONE:** Indica que no se debe capitalizar el texto. Es el valor por defecto.
- **CAPITALIZE:** Indica que se capitalice el primer carácter de cada palabra.
- **UPPERCASE:** Indica que se transforme todo el texto a mayúsculas.
- **LOWERCASE:** Indica que se transforme todo el texto a minúsculas.

### Ejemplos:

```
li { text-transform: uppercase; }
div { text-transform: lowercase; }
span { text-transform: capitalize; }
```

## PROPIEDAD VERTICAL-ALIGN

Especifica la alineación vertical para el elemento y es particularmente útil cuando se trabaja con texto e imágenes. Entre sus posibles valores podemos encontrar:

- **BASELINE:** Indica que el texto debe alinearse con respecto a la línea base del elemento padre. Es el valor por defecto.
- **SUB:** Indica que el texto debe alinearse con respecto a la línea base del subíndice del elemento padre.
- **SUPER:** Indica que el texto debe alinearse con respecto a la línea base del superíndice del elemento padre.

- **TOP:** Indica que el texto debe alinearse en la parte superior con respecto al elemento más alto de la línea.
- **TEXT-TOP:** Indica que el texto debe alinearse en la parte superior con respecto al elemento padre.
- **MIDDLE:** Indica que el texto debe alinearse en el medio del elemento padre.
- **BOTTOM:** Indica que el texto debe alinearse en la parte inferior con respecto al elemento más bajo de la línea.
- **TEXT-BOTTOM:** Indica que el texto debe alinearse en la parte inferior con respecto al elemento padre.

### Ejemplos:

```
img { vertical-align: top; }
div { vertical-align: middle; }
span { vertical-align: bottom; }
```

## PROPIEDAD WHITE-SPACE

Especifica como se debe gestionar el espacio en blanco en el elemento textual actual. Entre sus posibles valores podemos encontrar:

- **NORMAL:** Indica que las secuencias de espacios en blanco y tabulaciones se reducirán a un único elemento y el texto saltará a la línea siguiente cuando sea necesario. Es el valor por defecto.
- **NOWRAP:** Indica que las secuencias de espacios en blanco y tabulaciones se reducirán a un único elemento, pero el texto nunca saltará a la siguiente línea hasta que encuentre un elemento BR.
- **PRE:** Indica que las secuencias de espacios en blanco y tabulaciones se conservarán, pero el texto nunca saltará a la siguiente línea hasta que encuentre un elemento BR. Actúa como el elemento PRE de HTML.
- **PRE-LINE:** Indica que las secuencias de espacios en blanco y tabulaciones se reducirán a uno y el texto saltará a la siguiente línea cuando sea necesario.
- **PRE-WRAP:** Indica que las secuencias de espacios en blanco y tabulaciones se conservarán y el texto saltará a la siguiente línea cuando sea necesario.

### Ejemplos:

```
li { white-space: normal; }
div { white-space: nowrap; }
span { white-space: pre; }
```

## PROPIEDAD WORD-BREAK

Especifica como se deben cortar las palabras cuando no entran en el espacio asignado al elemento. Entre sus posibles valores podemos encontrar:

- **NORMAL:** Indica que las palabras deben cortarse en función de las reglas predeterminadas. En general, esto se convierte en que no se cortan y la palabra entera cae hacia la siguiente línea. Es el valor por defecto.
- **BREAK-ALL:** Indica que las palabras deben cortarse por cualquier carácter para que entren en el espacio del elemento.
- **KEEP-ALL:** Indica que las palabras NO deben cortarse bajo ninguna circunstancia, siempre y cuando, sean textos CJK (chino, japonés y coreano). Para el resto, es equivalente a NORMAL.
- **BREAK-WORD:** Indica que las palabras pueden cortarse en puntos arbitrarios para que entren en el espacio del elemento. Es el comportamiento habitual para textos no CJK.

## Ejemplos:

```
li { word-break: break-all; }
div { word-break: normal; }
span { word-break: break-word; }
```

## PROPIEDAD WORD-SPACING

Especifica el espacio que debe haber entre las palabras del texto. Entre sus posibles valores podemos encontrar:

- **NORMAL:** Indica que las palabras deben estar separadas por un espacio equivalente a 0.25EM. Es el valor por defecto.
- **[VALOR]:** Indica un valor establecido en una de las medidas permitidas de CSS.

## Ejemplos:

```
li { word-spacing: normal; }
div { word-spacing: 0.25em; }
span { word-spacing: 0.6vw; }
```

## PROPIEDAD WORD-WRAP

Especifica que las palabras del texto pueden ajustarse o separarse a la siguiente línea. Entre sus posibles valores podemos encontrar:

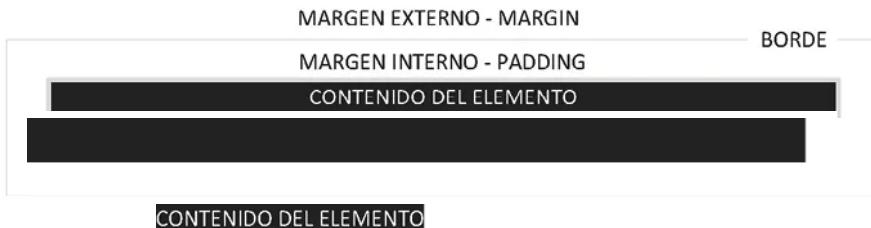
- **NORMAL:** Indica que se deben cortar las palabras sólo en los puntos de ruptura permitidos, lo que se suele convertir en que no se corten. Es el valor por defecto.
- **BREAK-WORD:** Indica que se pueden cortar las palabras en cualquier punto.

## Ejemplos:

```
li { word-wrap: break-word; }
p { word-wrap: normal; }
div { word-wrap: normal; }
```

## Márgenes internos y externos

Aunque se haga referencia con terminología similar en nuestro idioma, el uso de los márgenes internos y externos tiene diferentes objetivos. Mientras que el margen interno puede estar enfocado a provocar un efecto decorativo, el externo puede estar enfocado a provocar una mejor legibilidad.



## PROPIEDADES MARGIN Y PADDING

Las propiedades MARGIN y PADDING son unas propiedades compuestas que especifican el margen externo e interno respectivamente.

Ambas, admiten entre uno y cuatro valores que el agente de usuario interpretará de manera distinta, dependiendo del número de los mismos. Entre sus posibles valores podemos encontrar:

- **AUTO:** Indica que se debe dejar que los valores sean provistos por el agente de usuario y, únicamente, es aplicable a MARGIN.
- **[VALOR]:** Indica un valor establecido en una de las medidas permitidas de CSS.

Si se declaran los cuatro valores, la secuencia de asignación será valor superior, valor derecho, valor inferior y valor izquierdo. Por ejemplo, si quisiéramos asignar 15PX de margen externo a los elementos LI en todos sus lados podríamos hacer:

```
li { margin: 15px 15px 15px 15px; }
```

Si se declaran tres valores, la secuencia de asignación será valor superior, valor a los lados y valor inferior. Por ejemplo, si quisiéramos asignar 1EM de margen interno a las partes superior e inferior y 0.5EM a los lados derecho e izquierdo en todos los elementos P podríamos hacer:

```
p { padding: 1em 0.5em 1em; }
```

Si se declaran dos valores, la secuencia de asignación será valor superior e inferior y valor derecho e izquierdo. Por ejemplo, si quisiéramos eliminar el margen externo superior e inferior de los elementos DIV y asignar valores automáticos a los laterales, podríamos hacer:

```
div { margin: 0 auto; }
```

Si se declara un único valor, la secuencia de asignación será para todos los valores, es decir, se asignará el mismo valor para el margen superior, para el margen derecho, para el margen inferior y para el margen izquierdo. Por ejemplo, si quisiéramos asignar 15PX de margen interno a los elementos LI en todos sus lados podríamos hacer:

```
li { padding: 15px; }
```

En lo referente a los valores por defecto, la propiedad MARGIN tiene asignado el valor AUTO a todos sus lados y, la propiedad PADDING tiene asignado el valor 0 a todos sus lados.

Además de la compuesta, CSS permite especificar los valores de margen de forma independiente a través de la adición de los sufijos TOP, RIGHT, BOTTOM y LEFT. La concatenación de las palabras clave MARGIN o PADDING con uno de estos sufijos, provocará que se asigne el valor para el lado especificado.

### Ejemplos:

```
ul { margin-top: 2ch; }
li { margin-right: auto; }
p { padding-bottom: 1vw; }
div { padding-left: 1rem; }
```

## Bordes

### PROPIEDAD BORDER

Es una propiedad compuesta que especifica el borde del elemento. Se compone de un valor de ancho de borde, un estilo de borde y un color de borde.

Todas estas características pueden asignarse a través de sus propiedades individuales BORDER-COLOR, que permite asignar el color del borde, BORDER-STYLE, que permite asignar el estilo del borde y BORDER-WIDTH, que permite asignar el ancho o tamaño de cada uno de los bordes.

La secuencia de asignación para esta propiedad es ancho, estilo y color.

### Ejemplo:

```
div { border: 2px solid #ff0000; }
```

## PROPIEDAD BORDER-COLOR

Es una propiedad compuesta que especifica el color de los bordes del elemento. Al igual que sucede con los márgenes internos y externos, el orden de asignación es valor superior, valor derecho, valor inferior y valor izquierdo. Entre sus posibles valores podemos encontrar:

- **TRANSPARENT**: Indica que los bordes deben ser de color transparente.
- **[COLOR]**: Es un valor que puede ser uno de los valores preestablecidos por los navegadores, como pueda ser BLACK o WHITE o, un código de color en hexadecimal, RGBA o HSLA.

Si se declaran los cuatro valores, la secuencia de asignación será valor superior, valor derecho, valor inferior y valor izquierdo. Por ejemplo, si quisiéramos asignar un color rojo a los elementos LI en todos sus lados podríamos hacer:

```
li { border-color: red #FF0000 rgba(255, 0, 0); }
```

Si se declaran tres valores, la secuencia de asignación será valor superior, valor a los lados y valor inferior. Por ejemplo, si quisiéramos asignar un color azul claro a los lados superior e inferior y un color azul oscuro a los lados derecho e izquierdo en todos los elementos P podríamos hacer:

```
p { border-color: lightblue darkblue lightblue; }
```

Si se declaran dos valores, la secuencia de asignación será valor superior e inferior y valor derecho e izquierdo. Por ejemplo, si quisiéramos asignar un color negro a los lados superior e inferior y un color gris a los lados derecho e izquierdo en todos los elementos DIV podríamos hacer:

```
div { border-color: black gray; }
```

Si se declara un único valor, la secuencia de asignación será para todos los valores, es decir, se asignará el mismo valor al borde superior, borde derecho, borde inferior y borde izquierdo. Por ejemplo, si quisiéramos asignar un color naranja a los elementos LI en todos sus lados podríamos hacer:

```
li { border-color: orange; }
```

Por último, cabe mencionar que, además de la compuesta, CSS permite especificar los valores de forma independiente a través de la adición de los prefijos TOP, RIGHT, BOTTOM y LEFT. La construcción de BORDER-COLOR con estos nombres, provocará que se asigne el valor para el lado especificado.

### Ejemplos:

```
div { border-top-color: rgb(0, 0, 0); }
div { border-right-color: rgba(0, 0, 0, 1); }
div { border-bottom-color: hsl(0, 0%, 0%); }
div { border-left-color: hsla(0, 0%, 0%, 1); }
```

## PROPIEDAD BORDER-IMAGE

Es una propiedad compuesta que especifica la imagen que se debe mostrar en vez de los bordes del elemento. Se compone de una propiedad requerida, que es la imagen o gradiente, aunque admite cuatro parámetros más que son el tamaño del corte (o SLICE) el tamaño de la imagen, el punto de inicio y las opciones de repetición.

### Ejemplo:

```
div { border-image:url(./borde-cool.png) 25 25 repeat; }
```

No obstante, esta propiedad no se suele utilizar de forma compuesta puesto que genera bastantes dudas su especificación debido, fundamentalmente, a que tiene una sintaxis variable. En su lugar, se suele recurrir a sus variaciones específicas que resultan más claras de comprender cuando se están interpretando por una persona.

Entre sus propiedades podemos encontrar:

- **BORDER-IMAGE-SOURCE:** Indica la imagen o gradiente a utilizar.
- **BORDER-IMAGE-SLICE:** Indica la posición de corte de la imagen provista por BORDER-IMAGE-SOURCE. Para ello, la imagen se divide en 9 secciones o partes que comprenden las cuatro esquinas, los cuatro bordes y el centro. El centro suele tratarse como transparente, a no ser que se indique el valor FILL. El tamaño de estas secciones o partes se calculará en función del valor aquí establecido. Así, un valor sin unidad de medida será interpretado en píxeles para imágenes rasterizadas o como coordenadas para imágenes vectoriales. Si se especifica un valor de porcentaje, además se podrá utilizar el valor clave FILL para indicar que se rellene el centro del elemento también.
- **BORDER-IMAGE-WIDTH:** Indica el ancho de la imagen. Sus posibles valores pueden ser un número, un tanto por ciento o la palabra clave AUTO.
- **BORDER-IMAGE-OUTSET:** Indica cómo de lejos se debe extender el área de la imagen más allá de las secciones de borde. Su valor por defecto es 0.
- **BORDER-IMAGE-REPEAT:** Indica cómo se debe repetir la imagen para llenar el área del borde. Entre sus valores está STRETCH, que indica que la imagen se debe estirar todo lo necesario para llenar el elemento, REPEAT, que indica que se debe utilizar la imagen a modo de mosaico con o sin cortes, ROUND que indica que se debe utilizar la imagen a modo de mosaico de forma escalada para evitar la división de los mismos y, SPACE, que indica que se debe utilizar la imagen a modo de mosaico sin cortes y que los espacios sobrantes deben distribuirse equitativamente en todo el área del elemento.

### Ejemplos:

```
div {  
border-image: url(./borde-cool.png);  
border-image-slice: 10%;  
border-image-width: auto;  
border-image-repeat: round;  
}
```

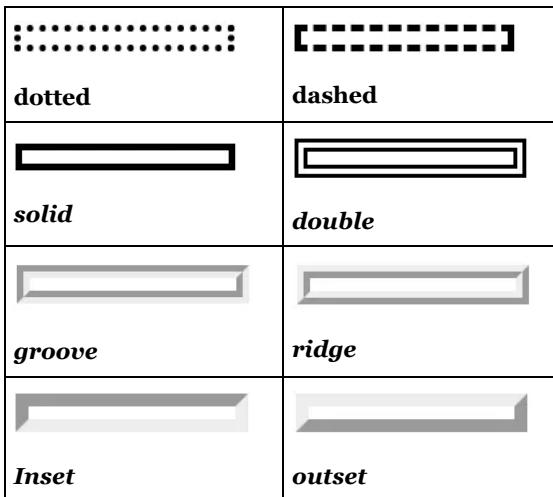
Cabe destacar que esta propiedad es válida para todos los elementos exceptuando los elementos internos del elemento TABLE y sólo cuando la propiedad BORDER-COLLAPSE está establecida a COLLAPSE.

## PROPIEDAD BORDER-STYLE

Es una propiedad compuesta que especifica el estilo de los bordes del elemento. Al igual que sucede otras propiedades, el orden de asignación es valor superior, valor derecho, valor inferior y valor izquierdo. Entre sus posibles valores podemos encontrar:

- **NONE**: Indica que NO se desean bordes en ninguno de los lados.
- **HIDDEN**: Indica que los bordes deben estar ocultos.
- **DOTTED**: Indica que los bordes deben ser únicos y punteados.
- **DASHED**: Indica que los bordes deben ser únicos y rayados.
- **SOLID**: Indica que los bordes deben ser únicos y continuos.
- **DOUBLE**: Indica que los bordes deben ser dobles y continuos.
- **GROOVE**: Indica que los bordes deben tener un efecto 3D acanalado.
- **RIDGE**: Indica que los bordes deben tener un efecto 3D ondulado.
- **INSET**: Indica que los bordes deben tener un efecto 3D presionado.
- **OUTSET**: Indica que los bordes deben tener un efecto 3D en relieve.

A continuación, se muestran ejemplos de todos y cada uno de los estilos de borde:



Si se declaran los cuatro valores, la secuencia de asignación será valor superior, valor derecho, valor inferior y valor izquierdo. Por ejemplo, si quisiéramos asignar un estilo sólido a los elementos LI en todos sus lados podríamos hacer:

```
li { border-style: solid solid solid solid ; }
```

Si se declaran tres valores, la secuencia de asignación será valor superior, valor derecho e izquierdo y valor inferior. Por ejemplo, si quisiéramos asignar un estilo de borde sólido a los lados superior e inferior y uno punteado a los lados derecho e izquierdo en todos los elementos P podríamos hacer:

```
p { border-style: solid dotted solid; }
```

Si se declaran dos valores, la secuencia de asignación será valor superior e inferior y valor derecho e izquierdo. Por ejemplo, si quisiéramos asignar un estilo rayado a los lados superior e inferior y uno punteado a los lados derecho e izquierdo en todos los elementos DIV podríamos hacer:

```
div { border-style: dashed dotted; }
```

Si se asigna un único valor, la secuencia de asignación será para todos los valores, es decir, se asignará el mismo valor al borde superior, borde derecho, borde inferior y borde izquierdo. Por ejemplo, si quisiéramos asignar un estilo 3D ondulado a los elementos LI en todos sus lados podríamos hacer:

```
li { border-style: ridge; }
```

Por último, cabe mencionar que, además de la compuesta, CSS permite especificar los valores de forma independiente a través de la adición de los prefijos TOP, RIGHT, BOTTOM y LEFT. La construcción de BORDER-STYLE con estos nombres, provocará que se asigne el valor para el lado especificado.

### Ejemplos:

```
div { border-top-style: solid; }
div { border-right-style: solid; }
div { border-bottom-style: solid; }
div { border-left-style: solid; }
```

## PROPIEDAD BORDER-RADIUS

Es una propiedad compuesta que especifica el tamaño de la curva (o radio) que une los bordes del elemento. Para esta propiedad, el orden de asignación es valor esquina superior izquierda, valor esquina superior derecha, valor esquina inferior derecha y valor esquina inferior izquierda.

Sus posibles valores sólo pueden asignarse a través de un valor establecido en una de las medidas permitidas de CSS.

Si se declaran los cuatro valores, la secuencia de asignación será la anteriormente comentada. Así, por ejemplo, si quisiéramos asignar un radio de borde de 15PX a los elementos LI en todas sus esquinas podríamos hacer:

```
li { border-radius: 15px 15px 15px 15px; }
```

Y si lo ejecutásemos, el resultado sería algo como:

ESTE ELEMENTO TIENE UN BORDER-RADIUS DE 15PX 15PX 15PX 15PX

Si se declaran tres valores, la secuencia de asignación será valor esquina superior izquierda, valor esquinas superior derecha e inferior izquierda y valor esquina inferior derecha. Por ejemplo, si quisiéramos asignar un radio de borde de 15PX a las esquinas superior izquierda e inferior derecha y un radio de borde de 0PX a las esquinas superior derecha e inferior izquierda en todos los elementos P podríamos hacer:

```
p { border-radius: 15px 0px 15px; }
```

Y si lo ejecutásemos, el resultado sería algo como:

ESTE ELEMENTO TIENE UN BORDER-RADIUS DE 15PX 0PX 15PX

Si se declaran dos valores, la secuencia de asignación será valor esquinas superior izquierda e inferior derecha y valor esquinas superior derecha e inferior izquierda. Por ejemplo, si quisiéramos asignar un radio de borde de 15PX a las esquinas superior izquierda e inferior derecha y un radio de borde de 0PX a las esquinas superior derecha e inferior izquierda en todos los elementos DIV podríamos hacer:

```
div { border-width: 15px 0px; }
```

Y si lo ejecutásemos, el resultado sería algo como:

ESTE ELEMENTO TIENE UN BORDER-RADIUS DE 15PX 0

Si se declara un único valor, la secuencia de asignación será para todos los valores, es decir, se asignará el mismo valor para las esquinas superior izquierda, superior derecha, inferior derecha e inferior izquierda. Por ejemplo, si quisiéramos asignar un radio de borde de 15PX a los elementos LI en todas sus esquinas podríamos hacer:

```
li { border-width: 15px; }
```

Y si lo ejecutásemos, el resultado sería algo como:

```
ESTE ELEMENTO TIENE UN BORDER-RADIUS DE 15 PÍXELES
```

## PROPIEDAD BORDER-WIDTH

Es una propiedad compuesta que especifica el tamaño de los bordes del elemento. Al igual que sucede con los márgenes internos y externos, y otras propiedades de su mismo contexto, el orden de asignación es valor superior, valor derecho, valor inferior y valor izquierdo. Entre sus posibles valores podemos encontrar:

- **THIN**: Indica que los bordes deben tener un tamaño fino o delgado, equivalente a 1 píxel.
- **MEDIUM**: Indica que los bordes deben tener un tamaño medio, equivalente a 3 píxeles. Es el valor por defecto.
- **THICK**: Indica que los bordes deben tener un tamaño grueso, equivalente a 5 píxeles.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

Si se declaran los cuatro valores, la secuencia de asignación será valor superior, valor derecho, valor inferior y valor izquierdo. Por ejemplo, si quisiéramos asignar un tamaño de borde de 2PX a los elementos LI en todos sus lados podríamos hacer:

```
li { border-width: 2px 2px 2px 2px; }
```

Si se declaran tres valores, la secuencia de asignación será valor superior, valor derecho e izquierdo y valor inferior. Por ejemplo, si quisiéramos asignar un tamaño de borde de 2PX a los lados superior e inferior y un tamaño de borde de 1PX a los lados derecho e izquierdo en todos los elementos P podríamos hacer:

```
p { border-width: 2px 1px 2px; }
```

Si se declaran dos valores, la secuencia de asignación será valor superior e inferior y valor derecho e izquierdo. Por ejemplo, si quisiéramos asignar un tamaño de borde de 2PX a los lados superior e inferior y un tamaño de borde de 4PX a los lados derecho e izquierdo en todos los elementos DIV podríamos hacer:

```
div { border-width: 2px 4px; }
```

Si se declara un único valor, la secuencia de asignación será para todos los valores, es decir, se asignará el mismo valor al borde superior, borde derecho, borde inferior y borde izquierdo. Por ejemplo, si quisiéramos asignar un tamaño de borde de 2PX a los elementos LI en todos sus lados podríamos hacer:

```
li { border-width: 2px; }
```

Por último, cabe mencionar que, además de la compuesta, CSS permite especificar los valores de forma independiente a través de la adición de los prefijos TOP, RIGHT, BOTTOM y LEFT. La construcción de BORDER-WIDTH con estos nombres, provocará que se asigne el valor para el lado especificado.

## Ejemplos:

```
div { border-top-width: 2px; }  
div { border-right-width: 1px; }  
div { border-bottom-width: 1px; }  
div { border-left-width: 2px; }
```

## Colores de fondo

### PROPIEDAD BACKGROUND

Aunque la propiedad BACKGROUND admite múltiples opciones, uno de los usos más extendidos es, únicamente, la asignación de un color de fondo o degradado. De hecho, su definición permite establecer un color de fondo, una imagen o efecto gradiente, una posición, un tamaño, una opción de repetición, un origen, una opción de extensión y un de desplazamiento.

Todos estos posibles valores son interpretados de forma unívoca e independiente y, por decirlo así, todos son opcionales, lo cual permite que se puedan asignar uno o varios valores en cualquier orden.

Al igual que sucede con otras propiedades, también permite la asignación a través de sus propiedades individuales y, aunque todas ellas se verán, de forma independiente, en el capítulo de imágenes y multimedia, aquí las comentaremos por encima, sobre todo, para obtener una idea de su capacidad y utilidad.

- **BACKGROUND-COLOR:** Permite definir el color del fondo.
- **BACKGROUND-IMAGE:** Permite definir la imagen o efecto gradiente.
- **BACKGROUND-POSITION:** Permite definir la posición de la imagen o fondo.
- **BACKGROUND-SIZE:** Permite definir el tamaño de la imagen o fondo.
- **BACKGROUND-REPEAT:** Permite definir las opciones de repetición del fondo.
- **BACKGROUND-ORIGIN:** Permite definir el origen de la imagen o fondo.
- **BACKGROUND-CLIP:** Permite definir las opciones de extensión para el fondo.
- **BACKGROUND-ATTACHMENT:** Permite definir opciones de desplazamiento.

Como se puede observar, su sintaxis es bastante compleja, sin embargo, su uso está muy recomendado, debido, fundamentalmente, a que está diseñada para ahorrar tiempo de proceso y carga.

### Ejemplos:

```
div { background: purple; }
div { background: url(./imagen-fondo.png) repeat-x; }
div { background: padding-box black; }
div { background: no-repeat center/cover url(./imagen-fondo.png); }
li { background: #FFF url("plus.png") no-repeat fixed left center; }
```

### PROPIEDAD BACKGROUND-COLOR

Especifica el color del fondo que será aplicado en el elemento, el cual afecta (o incluye) el espacio asignado al margen interno y al borde, pero no al margen externo. Entre sus posibles valores podemos encontrar:

- **[COLOR]:** Indica el color de fondo en formato HSL, RGBA o hexadecimal.
- **TRANSPARENT:** Indica que el color de fondo es transparente. Es equivalente a RGBA (0, 0, 0, 0).

### Ejemplo:

```
div { background-color: rgba(128, 0, 128, 1); }
```

## Listas

### PROPIEDAD LIST-STYLE

Es una propiedad compuesta que especifica el tipo de lista que se desea utilizar en el elemento. Se compone de un valor de estilo de lista, una posición y una posible imagen.

Todas estas características pueden asignarse a través de sus propiedades individuales, las cuales pasamos a describir a continuación:

- **LIST-STYLE-IMAGE:** Permite definir la imagen que desea utilizar como viñeta.
- **LIST-STYLE-POSITION:** Permite definir la posición de la viñeta.
- **LIST-STYLE-TYPE:** Permite definir el tipo de viñeta cuando no se utiliza una imagen como marcador.

Dado que se van a comentar todas y cada una de las propiedades independientes que pueden asignarse a esta propiedad compuesta, sólo expondremos un ejemplo.

#### Ejemplo:

```
ul { list-style: circle inside url("circle.png"); }
```

### PROPIEDAD LIST-STYLE-IMAGE

Especifica la imagen que se debe utilizar como viñeta o marcador. Entre sus posibles valores podemos encontrar:

- **NONE:** Indica que no se utilizará ninguna imagen y que, por tanto, será la propiedad LIST-STYLE-TYPE quién definirá el tipo de marcador de la lista. Es el valor por defecto.
- **[URL]:** Indica la dirección de la imagen a utilizar.

#### Ejemplo:

```
ul { list-style-image: url("../img/circle.png"); }
```

### PROPIEDAD LIST-STYLE-POSITION

Especifica la posición de la viñeta en los elementos de tipo lista. Entre sus posibles valores podemos encontrar:

- **INSIDE:** Indica que las viñetas deben estar dentro del propio elemento, por lo que, para verse correctamente, se deberá aplicar un margen interno o padding.
- **OUTSIDE:** Indica que las viñetas deben estar fuera del elemento, por lo que, para verse correctamente, se deberá aplicar un margen externo o margin. Es el valor por defecto.

#### Ejemplo:

```
ul { list-style-position: outside; }
```

### PROPIEDAD LIST-STYLE-TYPE

Especifica el tipo de viñeta que se debe utilizar en los elementos de tipo lista. Esta propiedad presenta multitud de posibles valores, pero los recurrentes o frecuentemente utilizados son:

- **CIRCLE:** Indica que la viñeta debe ser un círculo hueco.
- **DISC:** Indica que la viñeta debe ser un círculo relleno.

- **DECIMAL:** Indica que la viñeta debe ser 1, 2, 3, 4, ....
- **DECIMAL-LEADING-ZERO:** Indica que la viñeta debe ser 01, 02, 03, 04, ....
- **LOWER-ALPHA:** Indica que la viñeta debe ser a, b, c, d, ....
- **LOWER-ROMAN:** Indica que la viñeta debe ser i, ii, iii, iv, ....
- **NONE:** Indica que NO se debe mostrar la viñeta.
- **SQUARE:** Indica que la viñeta debe ser un cuadrado.
- **UPPER-ALPHA:** Indica que la viñeta debe ser A, B, C, D, ...
- **UPPER-ROMAN:** Indica que la viñeta debe ser I, II, III, IV, ...

### Ejemplos:

```
ul { list-style-type: circle; }
ul { list-style-type: square; }
ul { list-style-type: upper-alpha; }
ul { list-style-type: lower-roman; }
ul { list-style-type: disc; }
```

## Posicionamiento

### PROPIEDAD CLEAR

Especifica en qué lado debe romperse la línea de posicionamiento flotante proporcionada por la propiedad FLOAT. El efecto de romper el posicionamiento flotante viene a significar que, los elementos que estén por la parte que indica esta propiedad, caerán hacia la línea siguiente. Entre sus posibles valores podemos encontrar:

- **BOTH:** Indica que debe romperse por ambos lados, independientemente del valor de la propiedad FLOAT.
- **LEFT:** Indica que romperse por el lado izquierdo. Esto sólo ocurrirá, siempre y cuando, el valor de la propiedad FLOAT sea LEFT.
- **NONE:** Indica que se permiten elementos flotantes a ambos lados del elemento actual. Es el valor por defecto.
- **RIGHT:** Indica que romperse por el lado derecho. Esto sólo ocurrirá, siempre y cuando, el valor de la propiedad FLOAT sea RIGHT.

Por dejar el tema algo más claro, por ejemplo, si tenemos tres elementos con la propiedad FLOAT establecida a LEFT o RIGHT, y establecemos la propiedad CLEAR a BOTH en todos ellos, cada elemento debería posicionarse en una nueva línea.

Ahora, si esos mismos elementos tienen la propiedad FLOAT establecida a LEFT o RIGHT, y establecemos la propiedad CLEAR al mismo valor que FLOAT en el segundo, el primero debería permanecer en la línea de posicionamiento actual y, el resto, deberían posicionarse en una nueva línea. Sin embargo, si al segundo elemento se le establece la propiedad CLEAR a otro valor que no sea el de FLOAT, no tendrá ningún efecto.

### Ejemplos:

```
aside { clear: right; }
div { clear: both; }
p { clear: left; }
```

## NOTA

Los elementos que tengan un posicionamiento absoluto, es decir, tengan la propiedad POSITION establecida a ABSOLUTE, ignorarán esta propiedad.

## PROPIEDAD FLOAT

Especifica un posicionamiento flotante para el elemento. Entre sus posibles valores podemos encontrar:

- **LEFT:** Indica que el elemento debe estar flotando a la izquierda.
- **NONE:** Indica que NO debe flotar, es decir, que se mostrará justo dónde aparezca el elemento. Es el valor por defecto.
- **RIGTH:** Indica que el elemento debe estar flotando a la derecha.

### Ejemplos:

```
aside { float: right; }
div { float: left; }
p { float: none; }
```

## NOTA

Los elementos que tengan un posicionamiento absoluto, es decir, tengan la propiedad POSITION establecida a ABSOLUTE, ignorarán esta propiedad.

## PROPIEDAD POSITION

Especifica el tipo de posicionamiento utilizado para el elemento. Entre sus posibles valores podemos encontrar:

- **ABSOLUTE:** Indica que el elemento se posiciona con respecto al primer ancestro que tenga un posicionamiento relativo o estático.
- **FIXED:** Indica que el elemento se posiciona con respecto a la ventana del navegador.
- **RELATIVE:** Indica que el elemento debe posicionarse de forma relativa con respecto a su elemento hermano o anterior.
- **STATIC:** Indica que los elementos se procesan y posicionan en el orden en el que llegan o aparecen. Es el valor por defecto.
- **STICKY:** Sólo es efectivo cuando el elemento presenta una barra de desplazamiento e indica que el elemento debe tener un comportamiento mixto de relativo y fijo. Por ejemplo, si un elemento contenedor tiene la barra de desplazamiento habilitada y uno de sus elementos hijo se establece como STICKY, mientras esté por encima del punto indicado por la propiedad TOP, se comportará como si tuviese un posicionamiento relativo. Sin embargo, en el momento en que se sobrepase el valor establecido por TOP, se quedará adherido como si tuviese posicionamiento fijo.

### Ejemplos:

```
body { position: relative; }
aside { position: sticky; }
div { position: absolute; }
p { position: fixed; }
span { position: inherit; }
```

## NOTA

Cuando se trabaja con posicionamientos absolutos (ABSOLUTE), la coordenada (0,0) casi nunca será el mismo punto físico porque dependerá de su último ancestro con posicionamiento estático (STATIC) o relativo (RELATIVE). Sin embargo, cuando se trabaja con posicionamientos fijos (FIXED), la coordenada (0,0) siempre será el mismo punto físico.

## PROPIEDAD BOTTOM

Especifica que el elemento debe estar posicionado con respecto a su propiedad inferior. Entre sus posibles valores podemos encontrar:

- **AUTO**: Indica que el navegador es quién debe calcular la posición. Es el valor por defecto.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

La propiedad BOTTOM es muy sencilla, no obstante, posee unas reglas que hay que tener claras. Por ejemplo, si el posicionamiento es ABSOLUTE, el punto de origen desde el cual, el elemento se empezará a desplazar, será el equivalente al valor de la propiedad BOTTOM de su primer ancestro con posicionamiento relativo.

Si el posicionamiento es ABSOLUTE, FIXED o RELATIVE, su desplazamiento será hacia arriba o hacia abajo en función de si es o no positivo, respectivamente.

Si el posicionamiento es STICKY (adherido), se comportará como si tuviese un posicionamiento relativo hasta que llegue a ese valor establecido. A partir de ese momento, se quedará adherido como si tuviese posicionamiento fijo.

Ahora bien, si su posicionamiento es STATIC, la propiedad BOTTOM no tendrá ningún efecto.

### Ejemplo:

```
aside { bottom: 15vh; }
```

## PROPIEDAD LEFT

Especifica que el elemento debe estar posicionado con respecto a su propiedad izquierda. Entre sus posibles valores podemos encontrar:

- **AUTO**: Indica que el navegador es quién debe calcular la posición. Es el valor por defecto.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

La propiedad LEFT es muy sencilla, no obstante, posee unas reglas que hay que tener claras. Por ejemplo, si el posicionamiento es ABSOLUTE, el punto de origen desde el cual, el elemento se empezará a desplazar, será el equivalente al valor de la propiedad LEFT de su primer ancestro con posicionamiento relativo.

Si el posicionamiento es ABSOLUTE, FIXED o RELATIVE, su desplazamiento será hacia la derecha o hacia la izquierda en función de si es o no positivo, respectivamente.

Si el posicionamiento es STICKY (adherido) se comportará como si tuviese un posicionamiento relativo hasta que llegue a ese valor establecido. A partir de ese momento, se quedará adherido como si tuviese posicionamiento fijo. Eso sí, sólo funcionará si la barra de desplazamiento horizontal está visible y habilitada.

Ahora bien, si su posicionamiento es STATIC, la propiedad LEFT no tendrá ningún efecto.

### Ejemplo:

```
aside { left: 0; }
```

## **PROPIEDAD RIGHT**

Especifica que el elemento debe estar posicionado con respecto a su propiedad derecha. Entre sus posibles valores podemos encontrar:

- **AUTO:** Indica que el navegador es quién debe calcular la posición. Es el valor por defecto.
- **[VALOR]:** Indica un valor establecido en una de las medidas permitidas de CSS.

La propiedad RIGHT es muy sencilla, no obstante, posee unas reglas que hay que tener claras. Por ejemplo, si el posicionamiento es ABSOLUTE, el punto de origen desde el cual, el elemento se empezará a desplazar, será el equivalente al valor de la propiedad RIGHT de su primer ancestro con posicionamiento relativo.

Si el posicionamiento es ABSOLUTE, FIXED o RELATIVE, su desplazamiento será hacia la izquierda o hacia la derecha en función de si es o no positivo, respectivamente.

Si el posicionamiento es STICKY (adherido) se comportará como si tuviese un posicionamiento relativo hasta que llegue a ese valor establecido. A partir de ese momento, se quedará adherido como si tuviese posicionamiento fijo. Eso sí, sólo funcionará si la barra de desplazamiento horizontal está visible y habilitada.

Ahora bien, si su posicionamiento es STATIC, la propiedad RIGHT no tendrá ningún efecto.

### **Ejemplo:**

```
div { right: 2em; }
```

## **PROPIEDAD TOP**

Especifica que el elemento debe estar posicionado con respecto a su propiedad superior. Entre sus posibles valores podemos encontrar:

- **AUTO:** Indica que el navegador es quién debe calcular la posición. Es el valor por defecto.
- **[VALOR]:** Indica un valor establecido en una de las medidas permitidas de CSS.

La propiedad TOP es muy sencilla, no obstante, posee unas reglas que hay que tener claras. Por ejemplo, si el posicionamiento es ABSOLUTE, el punto de origen desde el cual, el elemento se empezará a desplazar, será el equivalente al valor de la propiedad TOP de su primer ancestro con posicionamiento relativo.

Si el posicionamiento es ABSOLUTE, FIXED o RELATIVE, su desplazamiento será hacia abajo o hacia arriba en función de si es o no positivo, respectivamente. Eso sí, esto sólo funcionará si la barra de desplazamiento vertical está visible y habilitada.

Si el posicionamiento es STICKY (adherido), se comportará como si tuviese un posicionamiento relativo hasta que llegue a ese valor establecido. A partir de ese momento, se quedará adherido como si tuviese posicionamiento fijo.

Ahora bien, si su posicionamiento es STATIC, la propiedad TOP no tendrá ningún efecto.

### **Ejemplo:**

```
p { top: auto; }
```

## **PROPIEDAD Z-INDEX**

Especifica el orden de apilamiento del elemento. Entre sus posibles valores podemos encontrar:

- **AUTO:** Indica que el navegador es quién debe calcular la posición. Es el valor por defecto.

- **[NÚMERO]**: Es un valor entero que indica el orden de presentación.

Si el valor es positivo, se presentará según el orden de apilamiento. Esto es, cuanto mayor sea el valor, más más arriba de la pila estará y más posibilidades de mostrarse primero tendrá.

Sin embargo, si el valor es negativo, se establecerá por detrás del orden de apilamiento general de sus hermanos o ancestros, por lo que puede que se oculte o vuelva invisible.

#### NOTA

Solamente tendrá efecto cuando el posicionamiento sea absoluto, relativo o fijo.

#### Ejemplos:

```
aside { z-index: 2; }  
p { z-index: -1; }
```

## Comportamientos y tamaños

### PROPIEDAD BOX-SIZING

Especifica cómo deben asignarse y calcularse el alto y ancho de los elementos. Esto es, si deben incluir los márgenes internos (padding) y/o los bordes, o no. Entre sus posibles valores podemos encontrar:

- **CONTENT-BOX**: Indica que se debe incluir sólo el contenido, e ignorar los márgenes internos y bordes. Es el valor por defecto.
- **BORDER-BOX**: Indica que se deben incluir el contenido, padding y bordes.

#### NOTA

En general, se puede afirmar que, trabajar con cajas o capas incluyendo los márgenes internos y los bordes es más fácil de manejar y facilita el diseño adaptativo, aunque no siempre.

#### Ejemplo:

```
aside { box-sizing: border-box; }
```

### PROPIEDAD DISPLAY

Especifica cómo se debe representar el elemento. Entre sus posibles valores podemos encontrar:

- **BLOCK**: Indica que el elemento debe mostrarse en bloque. Esto es, el elemento comenzará en una nueva línea y que ocupará todo el ancho disponible.
- **CONTENTS**: Indica que el elemento debe mostrarse como si fuese un contenido. Esto significa que se pierde el concepto de caja o capa y que, en su lugar, será reemplazado por un contenedor virtual que contiene sus elementos descendientes. Es experimental.
- **FLEX**: Indica que el elemento debe comportarse como un elemento de bloque flexible. Esto es, los elementos se recolocarán en cualquier dirección y podrán ampliar o reducir sus tamaños en función del espacio disponible para llenarlo de forma eficiente sin provocar desbordamientos ni cortes.
- **GRID**: Indica que el elemento debe comportarse como un elemento de bloque de una cuadrícula, similar a una tabla, por lo que los elementos se recolocarán a modo de filas y columnas.

- **INLINE:** Indica que el elemento debe mostrarse en línea. Esto conllevará que las propiedades de WIDTH y HEIGHT sean ignoradas y, por lo tanto, no tengan ningún efecto.
- **INLINE-BLOCK:** Indica que el elemento debe mostrarse como un conjunto de bloques en línea. Esto es, el elemento se trata como si estuviese en INLINE, pero admite la aplicación de las propiedades de WIDTH y HEIGHT.
- **INLINE-FLEX:** Indica que el elemento debe comportarse como un contenedor flexible, aunque con el concepto de línea. Esto es, el elemento se trata como si estuviese en INLINE, pero con las ventajas de una visualización en FLEX.
- **INLINE-GRID:** Indica que el elemento debe comportarse como un conjunto de contenedores de cuadrícula en línea. Esto es, el elemento se trata como si estuviese en INLINE, pero con las ventajas de una visualización en GRID.
- **INLINE-TABLE:** Indica que el elemento debe comportarse como si fuese un conjunto de tablas en línea. Esto es, el elemento se trata como si estuviese en INLINE, pero con las ventajas de una visualización en TABLE.
- **LIST-ITEM:** Indica que el elemento debe comportarse como si fuese una lista, con sus viñetas y demás características.
- **NONE:** Indica que el elemento NO debe mostrarse.
- **TABLE:** Indica que el elemento debe comportarse como si fuese una tabla. Esto es, sus descendientes serán tratados como una matriz bidimensional compuesta por filas y columnas.
- **TABLE-CAPTION:** Indica que el elemento debe comportarse como si fuese el elemento CAPTION de una tabla.
- **TABLE-CELL:** Indica que el elemento debe comportarse como si fuese la celda de una tabla, es decir, como el elemento TD.
- **TABLE-COLUMN:** Indica que el elemento debe comportarse como si fuese la columna de una tabla, es decir, como el elemento COL.
- **TABLE-COLUMN-GROUP:** Indica que el elemento debe comportarse como si de un grupo de columnas de una tabla se tratase, es decir, como el elemento COLGROUP.
- **TABLE-FOOTER-GROUP:** Indica que el elemento debe comportarse como si fuese el pie de una tabla, es decir, como el elemento TFOOTER.
- **TABLE-HEADER-GROUP:** Indica que el elemento debe comportarse como si fuese la cabecera de una tabla, es decir, como el elemento THEAD.
- **TABLE-ROW:** Indica que el elemento debe comportarse como si fuese la fila de una tabla, es decir, como el elemento TR.
- **TABLE-ROW-GROUP:** Indica que el elemento debe comportarse como si fuese el cuerpo de una tabla, es decir, como el elemento TBODY.

## NOTA

Mientras que, en documentos HTML, el valor por defecto suele ser BLOCK, en los documentos XML y SVG suele ser INLINE.

## Ejemplos:

```
aside { display: flex; }
```

```
div { display: inline-block; }
p { display: table; }
```

## PROPIEDAD HEIGHT

Especifica la altura del elemento. Entre sus posibles valores podemos encontrar:

- **AUTO**: Indica que el navegador es quién debe calcular y asignar el alto. Es el valor por defecto.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

### Ejemplos:

```
aside { height: auto; }
div { height: 100vh; }
```

## PROPIEDAD MAX-HEIGHT

Especifica la altura máxima del elemento. Entre sus posibles valores podemos encontrar:

- **NONE**: Indica que no hay máximo establecido. Es el valor por defecto.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

### Ejemplos:

```
aside { max-height: none; }
div { max-height: 100vh; }
```

## PROPIEDAD MAX-WIDTH

Especifica la anchura máxima del elemento. Entre sus posibles valores podemos encontrar:

- **NONE**: Indica que no hay máximo establecido. Es el valor por defecto.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

### Ejemplos:

```
aside { max-width: none; }
div { max-width: 100vw; }
```

## PROPIEDAD MIN-HEIGHT

Especifica la altura mínima del elemento. Entre sus posibles valores podemos encontrar:

- **AUTO**: Indica que no hay mínimo establecido. Es el valor por defecto.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

### Ejemplos:

```
aside { min-height: auto; }
div { min-height: 10vh; }
```

## PROPIEDAD MIN-WIDTH

Especifica la anchura mínima del elemento. Entre sus posibles valores podemos encontrar:

- **AUTO**: Indica que no hay mínimo establecido. Es el valor por defecto.

- **[VALOR]:** Indica un valor establecido en una de las medidas permitidas de CSS.

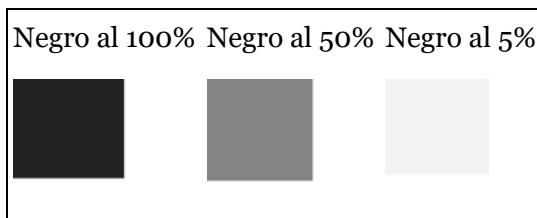
### Ejemplos:

```
aside { min-width: auto; }
div { min-width: 10vw; }
```

## PROPIEDAD OPACITY

Especifica la opacidad del elemento. Entre sus posibles valores podemos encontrar:

- **[NÚMERO]:** Es un valor decimal que indica, en tanto por uno, el porcentaje de visibilidad del elemento.



### Ejemplos:

```
/* Negro al 100%, 0% transparencia */
aside { opacity: 1; }

/* Negro al 50%, 50% transparencia */
div { opacity: 0.50; }

/* Negro al 5%, 95% transparencia */
p { opacity: 0.05; }
```

## PROPIEDAD OVERFLOW

Especifica cómo se debe comportar el elemento si su contenido se desborda, es decir, si su contenido no entra en el espacio asignado. Entre sus posibles valores podemos encontrar:

- **AUTO:** Indica que las barras de desplazamiento se mostrarán u ocultarán en función de si el contenido desborda o no, y que el contenido nunca debe verse fuera de los límites del elemento.
- **HIDDEN:** Indica que las barras de desplazamiento se mantengan ocultas, independientemente de que el contenido desborde o no, y que el contenido nunca debe verse fuera de los límites del elemento.
- **SCROLL:** Indica que las barras de desplazamiento se mantengan visibles, independientemente de que el contenido desborde o no, y que el contenido nunca debe verse fuera de los límites del elemento.
- **VISIBLE:** Indica que las barras de desplazamiento se mantengan ocultas y que el contenido se muestre, aunque se desborde. Es el valor por defecto.

### NOTA

La propiedad OVERFLOW no tiene ningún efecto en elementos que no tengan definida una altura específica.

### Ejemplos:

```
body { overflow: auto; }
aside { overflow: hidden; }
div { overflow: scroll; }
p { overflow: inherit; }
```

Por último, cabe mencionar que, existen unas variaciones de esta propiedad que permiten establecer los valores de forma independiente. Estas propiedades son OVERFLOW-X, que establece el comportamiento para los límites

derecho e izquierdo del elemento y, OVERFLOW-Y, que establece el comportamiento para los límites superior e inferior del elemento.

## PROPIEDAD VISIBILITY

Especifica si el elemento debe estar visible o no. Entre sus posibles valores podemos encontrar:

- **COLLAPSE**: Indica que el elemento debe aparecer oculto y sin ocupar espacio en pantalla. Este valor sólo es aplicable para los elementos TR, TBODY, COL y COLGROUP.
- **HIDDEN**: Indica que el elemento debe aparecer oculto, pero sin colapsar el espacio que ocupa.
- **VISIBLE**: Indica que el elemento debe aparecer visible. Es el valor por defecto.

## Ejemplo:

```
div { visibility: hidden; }
```

## PROPIEDAD WIDTH

Especifica la anchura del elemento. Entre sus posibles valores podemos encontrar:

- **AUTO**: Indica que el navegador es quién debe calcular y asignar el ancho. Es el valor por defecto.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

## Ejemplos:

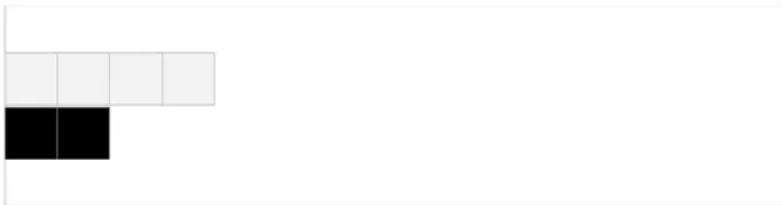
```
aside { width: auto; }  
div { width: 100vw; }
```

## Diseño de cajas flexibles

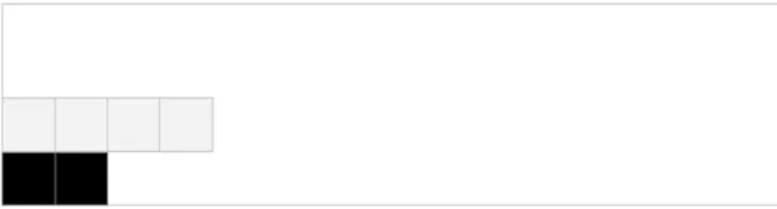
### PROPIEDAD ALIGN-CONTENT

Especifica cómo se deben distribuir los elementos verticalmente. Es una propiedad similar a ALIGN-ITEMS, pero en lugar de alinear elementos flexibles, alinea líneas flexibles. Entre sus posibles valores podemos encontrar:

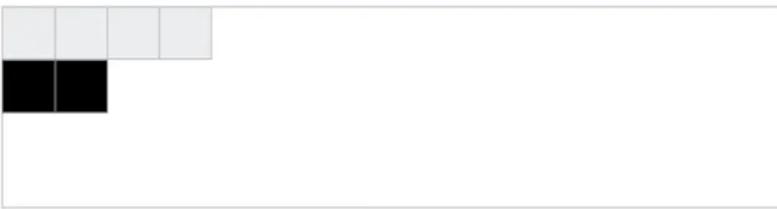
- **CENTER**: Indica que las líneas de elementos deben distribuirse verticalmente por la zona media del contenedor flexible.



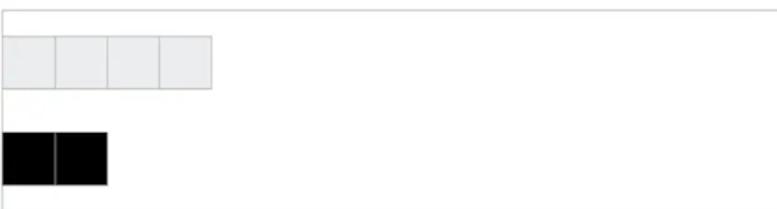
- **FLEX-END**: Indica que las líneas de elementos deben distribuirse verticalmente por la zona final del contenedor flexible.



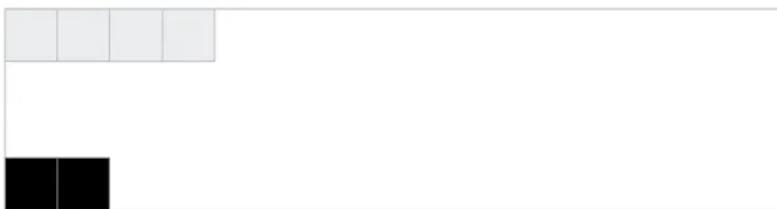
- **FLEX-START:** Indica que las líneas de elementos deben distribuirse verticalmente por la zona inicial del contenedor flexible.



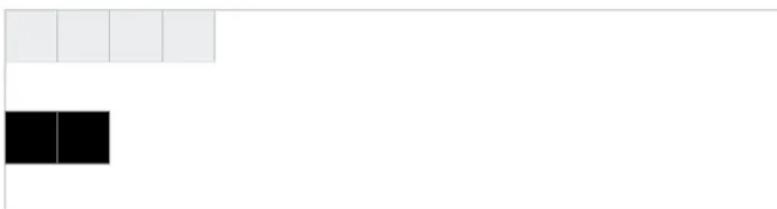
- **SPACE-AROUND:** Indica que las líneas de elementos deben distribuirse verticalmente de forma uniforme por el contenedor flexible con espacios perceptibles en cada extremo.



- **SPACE-BETWEEN:** Indica que las líneas de elementos deben distribuirse verticalmente de forma uniforme por los extremos del contenedor flexible.



- **STRECH:** Indica que las líneas de elementos deben ajustarse verticalmente para ocupar o llenar el espacio restante. Es el valor por defecto.



## NOTA

La propiedad ALIGN-CONTENT sólo tendrá algún efecto cuando el modo de visualización (DISPLAY) sea FLEX y la propiedad FLEX-WRAP esté establecida a WRAP o a WRAP-REVERSE.

## Ejemplos:

```
aside { display: flex; flex-wrap: wrap; align-content:space-around; }  
div { display: flex; flex-wrap: wrap; align-content:center; }  
p { display: flex; flex-wrap: wrap; align-content:flex-end; }
```

## PROPIEDAD ALIGN-ITEMS

Especifica la alineación predeterminada para los elementos que están dentro de un contenedor flexible. Entre sus posibles valores podemos encontrar:

- **BASELINE**: Indica que los elementos deben estar posicionados en la línea base del contenedor flexible.



- **CENTER**: Indica que los elementos deben estar posicionados en la parte central del contenedor flexible.



- **FLEX-END**: Indica que los elementos deben estar posicionados al final del contenedor flexible.



- **FLEX-START**: Indica que los elementos deben estar posicionados al principio del contenedor flexible.



- **STRETCH**: Indica que los elementos deben ajustarse al alto del contenedor para llenarlo. Es el valor por defecto.



## NOTA

La propiedad ALIGN- ITEMS sólo tendrá algún efecto cuando el modo de visualización (DISPLAY) sea FLEX y puede anularse a través de la propiedad ALIGN-SELF.

### Ejemplos:

```
aside { display: flex; flex: 1; align-items: center; }
div { display: flex; flex: 1; align-items: flex-start; }
```

## PROPIEDAD ALIGN-SELF

Especifica la alineación determinada para un elemento que está dentro de un contenedor flexible. Entre sus posibles valores podemos encontrar:

- **AUTO**: Indica que la alineación es inherente y que debe heredarse de la propiedad ALIGN-ITEMS definida en su contenedor. Es el valor por defecto.



- **BASELINE**: Indica que el elemento debe estar posicionado en la línea base del contenedor flexible.



- **CENTER**: Indica que el elemento debe estar posicionado en la parte central del contenedor flexible.



- **FLEX-END**: Indica que el elemento debe estar posicionado al final del contenedor flexible.



- **FLEX-START:** Indica que el elemento debe estar posicionado al principio del contenedor flexible.



- **STRECH:** Indica que el elemento se debe ajustar al alto del contenedor para llenarlo.



## NOTA

La propiedad ALIGN- ITEMS sólo tendrá algún efecto cuando el modo de visualización (DISPLAY) sea FLEX.

## Ejemplos:

```
aside { display: flex; flex: 1; align-self: center; }
div { display: flex; flex: 1; align-self: flex-start; }
p { display: flex; flex: 1; align-self: stretch; }
```

## PROPIEDAD FLEX

Es una propiedad compuesta que especifica, de forma conjunta, las propiedades de crecimiento flexible, decrecimiento flexible y el ancho del elemento.

El crecimiento viene determinado por la propiedad FLEX-GROW y se establece a través de un número que indica cómo irá creciendo el elemento con respecto al resto de elementos flexibles.

El decrecimiento viene determinado por la propiedad FLEX-SHRINK y se establece a través de un número que indica cómo irá decreciendo el elemento con respecto al resto de elementos flexibles.

El ancho viene determinado por la propiedad FLEX-BASIS y se establece a través de un en alguna de las unidades de medida estándar de CSS.

Además, entre sus posibles valores podemos encontrar algunos valores como:

- **AUTO:** Indica que el número de elementos es 1. Es equivalente a 1 1 AUTO.
- **INITIAL:** Indica que el número de elementos es 0. Es equivalente a 0 1 AUTO.
- **NONE:** Indica que el número de elementos es 0. Es equivalente a 0 0 AUTO.

Si se asignan los tres valores, se aplicarán en el orden anteriormente indicado, es decir, es como si se estableciese de forma independiente las variables FLEX-GROW, FLEX-SHRINK y FLEX-BASIS, en este orden.

```
li { flex: 1 1 auto; } /* FLEX-GROW FLEX-SHRINK FLEX BASIS */
```

Si se asignan dos valores, se podrán establecer o el crecimiento y el ancho, o el crecimiento y el decrecimiento. Es decir, es como si se estableciese de forma independiente las variables FLEX-GROW y FLEX-BASIS o FLEX-GROW y FLEX-SHRINK, en este orden.

```
p { flex: 1 100%; } /* FLEX-GROW FLEX-BASIS */  
p { flex: 1 1; } /* FLEX-GROW FLEX-SHRINK */
```

Si se asigna un único valor, podrá aplicarse o un crecimiento o un ancho, es decir, es como si se estableciese de forma independiente la variable FLEX-GROW o la variable FLEX-BASIS.

```
p { flex: 1; } /* FLEX-GROW */  
p { flex: 100%; } /* FLEX-BASIS */
```

## PROPIEDAD FLEX-BASIS

Especifica el ancho inicial de un elemento flexible. Entre sus posibles valores podemos encontrar:

- **AUTO**: Indica que el ancho es igual a la anchura predefinida del elemento flexible. Si, por casualidad, no se estableciese valor o no tuviese un valor especificado, la anchura será establecida en función de su contenido.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

Por ejemplo, imaginemos que tenemos un contenedor flexible con un ancho de 100 píxeles con tres elementos, en donde cada uno de ellos, tiene establecidas las propiedades FLEX-GROW y FLEX-SHRINK a 0 y la propiedad FLEX-BASIS a 33px. Esto debería producir un resultado similar al siguiente:



Ahora, si establecemos la propiedad FLEX-BASIS a 0 al segundo elemento, el resultado debería ser similar al siguiente:



Pero, si estableciésemos la propiedad FLEX-BASIS a 50px para el segundo elemento el resultado debería ser similar al siguiente:



Como se puede apreciar en la ilustración, el elemento 3 no entra en el contenedor de forma completa y se ve desbordado.

## Ejemplos:

```
div { flex-basis: 33px; }
div { flex-basis: 0; }
div { flex-basis: 50px; }
```

## PROPIEDAD FLEX-DIRECTION

Especifica la dirección de los elementos flexibles. Entre sus posibles valores podemos encontrar:

- **COLUMN**: Indica que los elementos deben mostrarse verticalmente empezando por arriba. Un ejemplo podría ser que todos los elementos se sitúen, unos debajo de otros, desde arriba del contenedor en formación de A-B-C-D.
- **COLUMN-REVERSE**: Indica que los elementos deben mostrarse verticalmente, empezando por abajo y con los elementos invertidos de orden. Un ejemplo podría ser que todos los elementos se sitúen, unos encima de otros, desde abajo del contenedor en formación de D-C-B-A.
- **ROW**: Indica que los elementos deben mostrarse horizontalmente, empezando por la izquierda. Un ejemplo podría ser que los elementos se situasen todos seguidos y alineados a la izquierda en la parte superior del contenedor en formación de A-B-C-D. Es el valor por defecto.
- **ROW-REVERSE**: Indica que los elementos deben mostrarse horizontalmente, empezando por la derecha y con los elementos invertidos de orden. Un ejemplo podría ser que los elementos se situasen todos seguidos y alineados a la derecha en la parte superior del contenedor en formación de D-C-B-A.

### Ejemplos:

```
aside { flex-direction: row-reverse; }
div { flex-direction: col; }
p { flex-direction: col-reverse; }
```

## PROPIEDAD FLEX-FLOW

Es una propiedad compuesta que especifica la dirección de los elementos flexibles y si se deben ajustar o no al ancho del contenedor.

El ajuste de los elementos viene determinado por la propiedad FLEX-WRAP, mientras que la dirección viene determinada por la propiedad FLEX-DIRECTION. El orden de asignación es arbitrario, es decir, se puede realizar la asignación de la propiedad a través de la dirección y el ajuste, o a la inversa.

En general, se recomienda utilizar esta, y las demás formas abreviadas, debido a que su interpretación y renderizado se realiza algo más rápido.

### Ejemplos:

```
aside div { flex-flow: row-reverse wrap; }
div { flex-flow: nowrap row; }
```

## PROPIEDAD FLEX-GROW

Especifica la relación de crecimiento del elemento con respecto a los demás. Entre sus posibles valores podemos encontrar:

- **[NÚMERO]**: Es un valor entero que indica, por decirlo así, el factor de multiplicación con respecto a los demás. Esto es, si todos los elementos de un contenedor flexible tienen un valor asignado de 1, menos uno que tiene un valor de 3, eso querrá decir que ese elemento será tres veces mayor que el resto.

## Ejemplos:

```
:nth-of-type(1) {flex-grow: 1;}  
:nth-of-type(2) {flex-grow: 3;}  
:nth-of-type(3) {flex-grow: 1;}
```

## PROPIEDAD FLEX-SHRINK

Especifica la relación de decrecimiento del elemento con respecto a los demás. Entre sus posibles valores podemos encontrar:

- **[NÚMERO]**: Es un valor entero que indica, por decirlo así, el factor de división con respecto a los demás. Esto es, si todos los elementos de un contenedor flexible tienen un valor asignado de 1, menos uno que tiene un valor de 3, eso querrá decir que ese elemento será tres veces menor que el resto.

## Ejemplos:

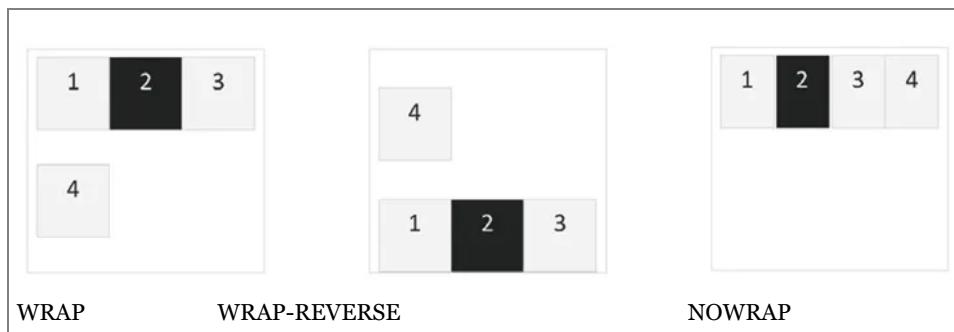
```
:nth-of-type(1) {flex-shrink: 1;}  
:nth-of-type(2) {flex-shrink: 3;}  
:nth-of-type(3) {flex-shrink: 1;}
```

## PROPIEDAD FLEX-WRAP

Especifica si el elemento debe ajustarse o no al ancho del contenedor. Entre sus posibles valores podemos encontrar:

- **NOWRAP**: Indica que el elemento no debe ajustarse. Es el valor por defecto.
- **WRAP**: Indica que el elemento debe ajustarse si fuese necesario.
- **WRAP-REVERSE**: Indica que el elemento debe ajustarse si fuese necesario, pero en orden inverso.

Por ejemplo, imaginemos que tenemos un contenedor flexible que tiene un ancho de 150 píxeles y, dentro, tiene definidos cuatro elementos de 40 por 40 píxeles cada uno. Dependiendo de cómo se establezca la propiedad FLEX-WRAP, debería producirse algo similar a uno de los siguientes resultados:



## Ejemplos:

```
aside { flex-wrap: wrap; }  
div { flex-wrap: wrap-reverse; }  
p { flex-wrap: nowrap; }
```

## PROPIEDAD JUSTIFY-CONTENT

Especifica la alineación horizontal para los elementos flexibles cuando éstos no utilizan, o no cubren, todo el espacio disponible. Entre sus posibles valores podemos encontrar:

- **CENTER**: Indica que los elementos deben estar posicionados en la parte central del contenedor flexible.



- **FLEX-END:** Indica que los elementos deben estar posicionados a la derecha del contenedor flexible.



- **FLEX-START:** Indica que los elementos deben estar posicionados a la izquierda del contenedor flexible.



- **SPACE-BETWEEN:** Indica que los elementos deben ajustarse de forma que los espacios adyacentes sean iguales.



- **SPACE-AROUND:** Indica que los elementos deben ajustarse de forma que los espacios entre ellos sean iguales, a excepción del primer y último elemento, en donde los espacios, anterior al primer elemento, y posterior al último elemento, deben ser la mitad que el espacio que hay entre el resto de los elementos.



### Ejemplos:

```
aside { justify-content: flex-start; }
div { justify-content: space-around; }
p { justify-content: center; }
```

## PROPIEDAD ORDER

Especifica el orden de un elemento flexible con respecto al resto de elementos que tiene a su mismo nivel. Entre sus posibles valores podemos encontrar:

- **[NÚMERO]**: Es un valor entero que indica el orden de aparición en la horizontal de izquierda a derecha. Por defecto, su valor es 0.

Por ejemplo, si tuviésemos un contenedor flexible con tres elementos y no estableciésemos la propiedad ORDER, los elementos aparecerían colocados como en la siguiente ilustración:



Ahora, si quisiéramos que se visualizasen en un orden determinado, por ejemplo, 2, 1, 3, en vez de 1, 2, 3, que es como estaban antes, podríamos hacer lo siguiente:

```
div :nth-child(1) {order: 2;}  
div :nth-child(2) {order: 1;}  
div :nth-child(3) {order: 3;}
```

Lo que generaría una salida como la siguiente ilustración:



## Interfaz de usuario

### PROPIEDAD CURSOR

Especifica el ícono que se debe mostrar en el puntero del ratón o dispositivo señalador cuando se acceda por encima de un elemento, se realice una acción de enfoque, o cualquier otro evento que relacione al dispositivo con el elemento. La propiedad tiene una gran variedad de elementos, sin embargo, los valores más frecuentes que podemos encontrar son:

- **AUTO**: Indica que sea el navegador el que establezca el ícono del puntero.
- **CELL**: Indica que el ícono de puntero debe ser una cruz simulando a un signo de suma.
- **DEFAULT**: Indica que el ícono de puntero debe ser el ícono por defecto.
- **E-RESIZE**: Indica que el ícono de puntero debe ser una flecha de doble dirección horizontal, o de este a oeste. Es el mismo que EW-RESIZE.
- **HELP**: Indica que el ícono de puntero debe llevar una interrogación.
- **MOVE**: Indica que el ícono de puntero debe ser dos flechas de doble dirección cruzadas, como indicando los cuatro puntos cardinales.

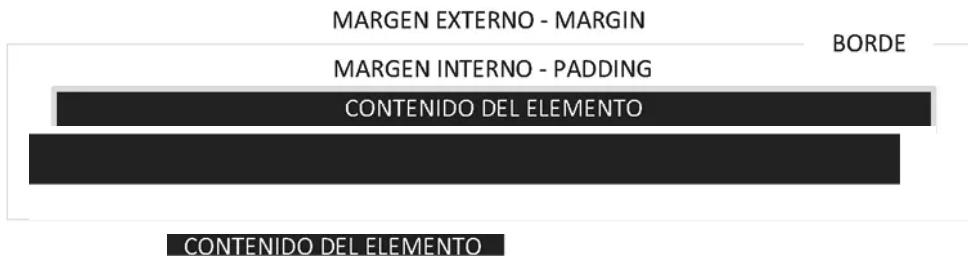
- **N-RESIZE**: Indica que el icono de puntero debe ser una flecha de doble dirección vertical, o de norte a sur. Es el mismo que NS-RESIZE.
- **NOT-ALLOWED**: Indica que el icono de puntero debe ser un signo de prohibido.
- **POINTER**: Indica que el icono de puntero debe ser una mano con el dedo índice extendido.
- **TEXT**: Indica que el icono de puntero debe ser el icono de selección de texto.

### Ejemplo:

```
aside a { cursor: pointer; }
```

## PROPIEDAD OUTLINE

Es una propiedad compuesta que especifica un borde externo para los elementos. Si recordamos la ilustración del apartado de márgenes, era como la siguiente ilustración:



Pues bien, el borde externo OUTLINE está justo entre la línea de borde y el margen externo.

Su sintaxis es idéntica a la propiedad BORDER. Esto es, se compone de un valor de ancho, un estilo y un color de borde externo y, al igual que sucede con BORDER, todas estas opciones pueden asignarse a través de sus propiedades individuales.

- **OUTLINE-COLOR**: Permite definir el color del borde externo.
- **OUTLINE-STYLE**: Permite definir el estilo del borde externo.
- **OUTLINE-WIDTH**: Permite definir el ancho del borde externo.

Dado que se van a comentar todas y cada una de las propiedades individuales que pueden asignarse a esta propiedad compuesta, sólo expondremos un ejemplo.

### Ejemplo:

```
div { outline: 1px dashed #000; }
```

## PROPIEDAD OUTLINE-COLOR

Especifica el color de los bordes externos del elemento. A diferencia de la propiedad BORDER-COLOR, OUTLINE-COLOR sólo se puede asignar un único valor para todos los lados del borde externo. Entre sus posibles valores podemos encontrar:

- **INVERT**: Indica que el color del borde externo debe ser una inversión del color proporcionado por la propiedad BORDER-COLOR. Es el valor por defecto.

- **[COLOR]**: Es un valor textual, como pueda ser BLACK o WHITE o, un código de color en hexadecimal, RGBA o HSLA.

### Ejemplo:

```
div { outline-color: black; }
```

## PROPIEDAD OUTLINE-OFFSET

Especifica el espacio en blanco que debe existir entre el borde interno proporcionado por la propiedad BORDER y el borde externo proporcionado por la propiedad OUTLINE. Entre sus posibles valores podemos encontrar:

- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

### Ejemplo:

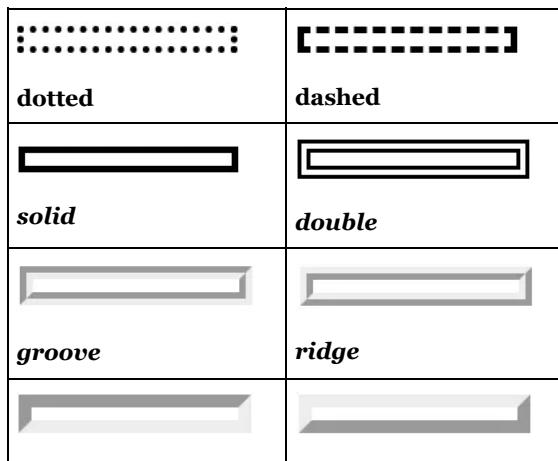
```
div { outline-offset: 0.4rem; }
```

## PROPIEDAD OUTLINE-STYLE

Especifica el estilo de los bordes del elemento. A diferencia de la propiedad BORDER-STYLE, OUTLINE-STYLE sólo se puede asignar un único valor para todos los lados del borde externo. Entre sus posibles valores podemos encontrar:

- **NONE**: Indica que NO se desean bordes en ninguno de los lados.
- **HIDDEN**: Indica que los bordes deben estar ocultos.
- **DOTTED**: Indica que los bordes deben ser únicos y punteados.
- **DASHED**: Indica que los bordes deben ser únicos y rayados.
- **SOLID**: Indica que los bordes deben ser únicos y continuos.
- **DOUBLE**: Indica que los bordes deben ser dobles y continuos.
- **GROOVE**: Indica que los bordes deben tener un efecto 3D acanalado.
- **RIDGE**: Indica que los bordes deben tener un efecto 3D ondulado.
- **INSET**: Indica que los bordes deben tener un efecto 3D presionado.
- **OUTSET**: Indica que los bordes deben tener un efecto 3D en relieve.

A continuación, se muestran ejemplos de todos y cada uno de los estilos de borde:



## Ejemplos:

```
div { outline-style: solid; }
div { outline-style: groove; }
div { outline-style: double; }
```

## PROPIEDAD OUTLINE-WIDTH

Especifica el tamaño de los bordes del elemento. A diferencia de la propiedad BORDER-WIDTH, OUTLINE-WIDTH sólo se puede asignar un único valor para todos los lados del borde externo. Entre sus posibles valores podemos encontrar:

- **THIN**: Indica que los bordes deben tener un tamaño fino o delgado, equivalente a 1 píxel.
- **MEDIUM**: Indica que los bordes deben tener un tamaño medio, equivalente a 3 píxeles. Es el valor por defecto.
- **THICK**: Indica que los bordes deben tener un tamaño grueso, equivalente a 5 píxeles.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

## Ejemplos:

```
div { outline-width: 2px; }
div { outline-width: 0.2vw; }
```

## PROPIEDAD USER-SELECT

Especifica si el contenido del elemento es seleccionable o no. Entre sus posibles valores podemos encontrar:

- **ALL**: Indica que la selección de todo texto se realiza a través de un clic, en vez de un doble click.
- **AUTO**: Indica que el texto es seleccionable si el navegador lo permite. Es el valor por defecto.
- **NONE**: Indica que NO es posible seleccionar el texto.
- **TEXT**: Indica que es posible seleccionar el texto.

## Ejemplos:

```
div { user-select: all; }
div { user-select: none; }
```

## Animaciones y transiciones

Dado que las animaciones y transiciones se ha decidido extraerlas de este capítulo por su extensión y dificultad y dado que todos los valores de esta propiedad pueden asignarse de forma independiente a través de sus propiedades específicas, ahora sólo comentaremos un par de ejemplos y, en el capítulo de animaciones, transiciones y efectos, se comentarán en detalle todas sus posibles variaciones.

## PROPIEDAD ANIMATION

Es una propiedad compuesta que especifica los detalles de cómo debe implementar una animación en el elemento. Se compone de un nombre de animación, una duración, una función de sincronización de tiempo, un retraso de empiece, un contador de iteraciones, una dirección, un modo de relleno y un estado de reproducción.

Todas estas características pueden asignarse a través de sus propiedades individuales y serán descritas en el capítulo de animaciones, transiciones y efectos.

- **ANIMATION-NAME:** Permite definir el nombre de la animación que se desea vincular al elemento, la cual hace referencia a la definición de una regla arroba KEYFRAMES.
- **ANIMATION-DURATION:** Permite definir la duración de la animación.
- **ANIMATION-TIMING-FUNCTION:** Permite definir la función de sincronización de tiempo.
- **ANIMATION-DELAY:** Permite definir el retraso de empiece.
- **ANIMATION-ITERATION-COUNT:** Permite definir el número o contador de iteraciones.
- **ANIMATION-DIRECTION:** Permite definir la dirección.
- **ANIMATION-FILL-MODE:** Permite definir el modo de relleno.
- **ANIMATION-PLAY-STATE:** Permite definir el estado de reproducción.

### Ejemplos:

```
div { animation: ejemplo1 5s infinite; }
li { animation: ejemplo2 1.5s ease-in-out forwards; }
```

## PROPIEDAD TRANSITION

Es una propiedad compuesta que especifica cómo deben suceder los cambios en el elemento. Se compone de una propiedad de transición, una duración, una función de sincronización de tiempo y un retraso de empiece.

Todas estas características pueden asignarse a través de sus propiedades individuales y serán descritas en el capítulo de animaciones, transiciones y efectos.

- **TRANSITION-NAME:** Permite definir el nombre de la transición que se desea vincular al elemento.
- **TRANSITION-DURATION:** Permite definir la duración de la transición.
- **TRANSITION-TIMING-FUNCTION:** Permite definir la función de sincronización de tiempo.
- **TRANSITION-DELAY:** Permite definir el retraso de empiece.

Dado que se van a comentar todas y cada una de las propiedades independientes que pueden asignarse a esta propiedad compuesta, sólo expondremos un ejemplo.

### Ejemplos:

```
/* Efecto de animación de 2 seg. cuando la propiedad WIDTH cambie */
div { transition: width 2s; }
/* Efecto de animación de 300ms cuando cambie cualquier propiedad de LI */
li { transition: all 0.3s ease; }
```

## Reglas arroba

### REGLA CHARSET

Especifica la codificación de caracteres que se debe utilizar en la hoja de estilos.

Aunque generalmente el único valor que se utiliza es UTF-8 por temas de estandarización y compatibilidad, existen otros muchos valores que puede admitir, disponibles en la web de IANA. Para consultar todos los posibles valores

puede visitarse la URL <https://www.iana.org/assignments/character-sets/character-sets.xhtml>.

No obstante, para que esta regla funcione, o se aplique, se deben tener en cuenta unas romas:

- La regla arroba CHARSET debe ser la primera declaración de la hoja de estilos.
- Si por cualquier razón, hubiese declaradas o definidas varias reglas CHARSET, la única que tendrá efecto será la primera, el resto serán ignoradas.
- La regla arroba CHARSET no puede estar declarada dentro de ningún atributo o estructura STYLE de HTML.

## NOTA

Esta regla no es compatible con Internet Explorer 11, aunque sí con todas las versiones de Microsoft Edge.

### Ejemplo:

```
@charset "utf-8"
```

## REGLA FONT-FACE

Especifica la una nueva fuente de texto, o nuevo tipo de fuente, para poder ser utilizada como tipografía en el documento o página web.

Cabe destacar que no todos los formatos de fuente están disponibles para todos los navegadores. Así, por ejemplo, los formatos de fuente de texto TTF/OTF y WOFF pueden ser interpretados por la mayoría de los agentes de usuario, pero el formato WOFF2 no está soportado por Internet Explorer ni Safari. En cuanto al formato SVG, es compatible con Safari, pero no con Firefox.

Por esta razón, lo normal es que se utilicen fuentes de texto vectoriales desde algún directorio interactivo de uso público, los cuales cargan todos los posibles formatos para que cada agente de usuario utilice el que más le convenga.

Para que la regla FONT-FACE pueda ser aplicada, se deben definir serie de parámetros u opciones:

- **FONT-FAMILY:** Define el nombre de la fuente que será usado después, en las reglas CSS para vincular y aplicar el estilo. Es un parámetro requerido.
- **SRC:** Permite definir la dirección o ubicación del archivo. Es un parámetro requerido.
- **FONT-STRETCH:** Permite expandir o condensar el texto. Es opcional y puede contener cualquiera de los posibles valores de FONT-STRETCH.
- **FONT-STYLE:** Permite establecer el estilo del texto. Es opcional y puede contener cualquiera de los posibles valores de FONT-STYLE.
- **FONT-WEIGHT:** Permite establecer el grosor del texto. Es opcional y puede contener cualquiera de los posibles valores de FONT-WEIGHT.
- **UNICODE-RANGE:** Permite definir el rango de caracteres Unicode que admite la fuente de texto. El valor por defecto es “U + 0-10FFFF”.

### Ejemplos:

```
/* Declaración de fuente básica */
@font-face {
  font-family: textFont;
  src: url(new_arial_100.ttf);
  font-weight: 100;
```

```

}

/* Declaración extraída de Google Fonts */
@font-face{
font-family:'Roboto';
font-style:normal;
font-weight:400;
src:local('Roboto Regular'),
local('Roboto-Regular'),
url(//fonts.gstatic.com/s/roboto/v18/KFOmCnqEu92Fr1Mu7mxKOzY.woff2)
format('woff2');unicode-range:U+1F00-1FFF;
}

```

## REGLA IMPORT

Permite incluir otras hojas de estilo dentro de la hoja de estilo actual. Debe estar declarada justo después de la declaración de la regla CHARSET.

### Ejemplo:

```
@import "custom-styles.css";
```

Aunque, en general, se suelen importar las hojas de estilos sin ningún control de medios o resoluciones, es posible hacerlo en función de estos parámetros.

### Ejemplo:

```
@import "mobile.css" screen and (max-width: 768px);
```

Para poder ver todos los posibles valores con los que se puede configurar la regla IMPORT se puede consultar la regla MEDIA, explicada un poco más adelante.

## REGLA KEYFRAMES

Permite especificar una animación a partir de las reglas CSS que estén contenidas en su declaración. Esta regla, que sólo es aplicable para el contexto de las animaciones CSS, permite cambiar las propiedades y valores durante todo el ciclo de la animación.

Para poder definir el ciclo de vida de la animación se pueden utilizar las palabras clave FROM y TO o recurrir a valores de porcentaje, aunque, siempre es mejor utilizar esta segunda opción por temas de compatibilidad.

## NOTA

La característica !IMPORTANT es ignorada dentro de esta regla.

### Ejemplo:

```

@keyframes ejemplo1 {
from {top: 0px;}
to {top: 200px;}
}
@keyframes ejemplo2 {
0% {top: 0px; left: 0; }
25% {top: 0px; left: 50px; }
50% {top: 50px; left: 50px}
75% {top: 50px; left: 0}
100% {top: 0; left: 0; }
}

```

Si se desea más información y ejemplos sobre las animaciones se puede ir directamente al capítulo de animaciones, transiciones y efectos.

## REGLA MEDIA

La regla MEDIA, referida habitualmente como “media query” o consulta de medios, se utiliza para aplicar diferentes reglas de estilo dependiendo de la resolución del dispositivo y/o medio.

Este tipo de consultas puede aplicarse para controlar varias casuísticas como son el medio, ancho y alto de la ventana gráfica (VIEWPORT), ancho y alto del dispositivo, la orientación y/o la resolución.

Seguramente, muchos de los lectores ya se habrán dado cuenta, o ya sabrán, que estas reglas MEDIA son muy utilizadas en diseños adaptativos y/o receptivos tanto en configuraciones de escritorio, como en tables o móviles. No obstante, también son utilizadas, como se ha podido observar en la propiedad IMPORT, para controlar los medios como son la pantalla, la impresora o la voz.

La regla MEDIA funciona de forma similar a como lo hacen otras propiedades de CSS ya que permite una declaración combinada o individual de un tipo de medio, con o sin la especificación de sus características e, incluso, una especificación sin tipo de medio.

### Ejemplo:

```
@media screen and (min-width: 768px) and (max-width: 1024px){  
    /* Reglas CSS aplicables para estas circunstancias */  
}
```

Los tipos de medios disponibles son:

- **ALL:** Indica que las reglas contenidas en la consulta de medios son válidas para cualquier tipo de medio. Es el valor por defecto.
- **PRINT:** Indica que las reglas contenidas en la consulta de medios sólo son válidas sólo para el tipo de medio definido como impresora.
- **SCREEN:** Indica que las reglas contenidas en la consulta de medios sólo son válidas para el medio definido como pantalla, independientemente de si pertenece a un dispositivo de escritorio, tablet o móvil.
- **SPEECH:** Indica que las reglas contenidas en la consulta de medios sólo son válidas para agentes de usuario que permiten la lectura mediante control de voz, como los lectores de pantalla utilizados por las personas con discapacidad.

Las características de medios, también llamadas funciones de medios son muchas y, algunas de ellas, no son compatibles con todos los agentes de usuario, sin embargo, aquí se comentarán la mayoría:

- **ANY-HOVER:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo de entrada disponible puede pasar por los elementos. Sus posibles valores son HOVER, para indicar que todos los dispositivos y mecanismos que tengan disponible esta opción deben aplicar esta consulta de medios cuando esté disponible el desplazamiento por los elementos o, NONE, para cuando NO esté disponible el desplazamiento por los elementos.

```
@media (any-hover: hover){  
    /* Reglas CSS aplicables cuando puede pasar por los elementos */  
}  
@media (any-hover: none){  
    /* Reglas CSS aplicables cuando NO puede pasar por los elementos */  
}
```

- **ANY-POINTER:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo disponga de un dispositivo señalador (o de tipo puntero), sea o no primario, y tenga una precisión determinada. Sus posibles valores son FINE, para indicar que el dispositivo señalador es de alta precisión, COARSE, que indica que el dispositivo señalador es de precisión limitada o, NONE, que indica que no hay dispositivo señalador.

```
@media (any-pointer: coarse){
/* Reglas CSS aplicables cuando el dispositivo es de baja precisión */
}
@media (any-pointer: fine){
/* Reglas CSS aplicables cuando el dispositivo es de alta precisión */
}
@media (any-pointer: none){
/* Reglas CSS aplicables cuando no hay dispositivo señalador */
}
```

- **ASPECT-RATIO:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando la relación de altura y anchura de la ventana gráfica o VIEWPORT se corresponda con el valor indicado. Además, existen dos variaciones que permiten controlar los valores máximo y mínimo a través de los prefijos MAX- y MIN-.

```
@media (aspect-ratio: 16/9){
/* Reglas CSS aplicables cuando el dispositivo es 16 a 9 */
}
@media (min-aspect-ratio: 1/1){
/* Reglas CSS aplicables cuando el dispositivo es, como máximo, 1 a 1 */
}
@media (max-aspect-ratio: 11/16){
/* Reglas CSS aplicables cuando el dispositivo es, como máximo, 11 a 16 */
}
```

- **COLOR:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando la profundidad de color (número de bits para representar un color) solicitada se corresponda con la profundidad de color del dispositivo. Sus posibles valores son 1, 2, 4, 8, 16, 24 y 32 y, por defecto, su valor es 8. Además, existen dos variaciones que permiten controlar los valores máximo y mínimo a través de los prefijos MAX- y MIN-.

```
@media (color){
/* Reglas CSS aplicables cuando el dispositivo es color */
}
@media (min-color: 8){
/* Reglas CSS aplicables cuando el dispositivo admite 8 bits */
}
@media (max-color: 16){
/* Reglas CSS aplicables cuando el dispositivo admite 16 bits */
}
```

- **COLOR-GAMUT:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el rango aproximado de colores admitido por el agente de usuario y dispositivo de salida se corresponda con el valor indicado. Sus posibles valores son SRGB, que admite la gama SRGB y son la inmensa mayoría de las pantallas de color, P3, que admite gama especificada por el espacio de color DCI P3 o una superior como SRGB y, REC2020, que admite la gama especificada por la Recomendación UIT-R BT.2020 Color Space o superior.

```
@media (color-gamut: srgb){
/* Reglas CSS aplicables para estas circunstancias */
}
@media (color-gamut: p3){
```

```
/* Reglas CSS aplicables para estas circunstancias */  
}  
@media (color-gamut: srgb){  
/* Reglas CSS aplicables para estas circunstancias */  
}
```

- **COLOR-INDEX:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo permita mostrar un cierto número de colores. Sus posibles valores van desde 0, que equivale a decir todos, hasta el permitido por cada dispositivo. Su valor por defecto es 0. Además, existen dos variaciones que permiten controlar los valores máximo y mínimo a través de los prefijos MAX- y MIN-.

```
@media (color-index: 0){  
/* Reglas CSS aplicables para estas circunstancias */  
}  
@media (min-color-index: 10){  
/* Reglas CSS aplicables cuando permite, como mínimo, 10 colores */  
}  
@media (max-color-index: 256){  
/* Reglas CSS aplicables cuando permite, como máximo, 256 colores */  
}
```

- **GRID:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo de salida utiliza un sistema basado en rejilla o es un mapa de bits. Sus posibles valores son 1 o 0 que indican si el dispositivo de salida está o no basado en rejilla, respectivamente.

```
@media (grid: 0){  
/* Reglas CSS aplicables cuando NO está basado en rejilla */  
}
```

- **HEIGHT:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando la altura del dispositivo de salida se corresponda con el valor indicado. Como sus análogas, posee dos variaciones que permiten controlar los valores máximo y mínimo a través de los prefijos MAX- y MIN-.

```
@media (height: 610px){  
/* Reglas CSS aplicables para estas circunstancias */  
}  
@media (min-height: 40vh){  
/* Reglas CSS aplicables para estas circunstancias */  
}  
@media (max-height: 80vh){  
/* Reglas CSS aplicables para estas circunstancias */  
}
```

- **HOVER:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo de entrada permita al usuario el desplazamiento por los elementos. Sus posibles valores son HOVER, para indicar que todos los dispositivos y mecanismos que tengan disponible esta opción deben aplicar esta consulta de medios cuando esté disponible el desplazamiento por los elementos o, NONE, para cuando NO esté disponible el desplazamiento por los elementos.

```
@media (hover: hover){  
/* Reglas CSS aplicables cuando puede desplazarse */  
}  
@media (hover: none){  
/* Reglas CSS aplicables cuando NO puede desplazarse */  
}
```

- **INVERTED-COLORS:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el agente de usuario, o el sistema operativo, tenga invertidos los colores. Sus posibles valores son INVERTED,

para indicar que se aplique cuando los colores estén invertidos y, NONE, para indicar que se aplique cuando los colores se muestran normalmente.

```
@media (inverted-colors: inverted){  
    /* Reglas CSS aplicables cuando los colores están invertidos */  
}  
  
@media (inverted-colors: none){  
    /* Reglas CSS aplicables cuando los colores NO están invertidos */  
}
```

- **MONOCHROME:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo sea monocromo. Sus posibles valores son o, para indicar que se aplique cuando NO es monocromo, o “vacío”, para indicar que se aplique cuando sí lo es.

```
@media (monochrome: o){  
    /* Reglas CSS aplicables cuando no es monocromo */  
}  
  
@media (monochrome){  
    /* Reglas CSS aplicables cuando es monocromo */  
}
```

- **ORIENTATION:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo tenga establecida una orientación de pantalla determinada. Sus posibles valores son **PORTRAIT**, para indicar que se aplique cuando el dispositivo está en posición vertical, lo que equivale a decir, cuando la anchura es menor que la altura y, **LANDSCAPE** para indicar que se aplique cuando el dispositivo está en horizontal, es decir, el caso contrario.

```
@media (orientation: landscape){  
    /* Reglas CSS aplicables cuando el dispositivo está en horizontal */  
}  
  
@media (orientation: portrait){  
    /* Reglas CSS aplicables cuando el dispositivo está en vertical */  
}
```

- **POINTER:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo disponga de un dispositivo señalador (o de tipo puntero) considerado primario y tenga una precisión determinada. Sus posibles valores son **FINE**, para indicar que el dispositivo señalador es de alta precisión, **COARSE**, que indica que el dispositivo señalador es de precisión limitada o, **NONE**, que indica que no hay dispositivo señalador.

```
@media (pointer: coarse){  
    /* Reglas CSS aplicables cuando el dispositivo es de baja precisión */  
}  
  
@media (pointer: fine){  
    /* Reglas CSS aplicables cuando el dispositivo es de alta precisión */  
}  
  
@media (pointer: none){  
    /* Reglas CSS aplicables cuando no hay dispositivo señalador */  
}
```

- **RESOLUTION:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando la densidad en píxeles del dispositivo se corresponda con el valor indicado. Como sus análogas, posee dos variaciones que permiten controlar los valores máximo y mínimo a través de los prefijos **MAX-** y **MIN-**.

```
@media (resolution: 100dppx){  
    /* Reglas CSS aplicables cuando la densidad es 100 píxeles */  
}
```

```

@media (min-resolution: 72dpi){
/* Reglas CSS aplicables cuando la densidad es, como mínimo, 72 píxeles */
}
@media (max-resolution: 300dpi){
/* Reglas CSS aplicables cuando la densidad es, como mínimo, 300 píxeles */
}

```

- **WIDTH:** Indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando la anchura del dispositivo de salida se corresponda con el valor indicado. Como sus análogas, posee dos variaciones que permiten controlar los valores máximo y mínimo a través de los prefijos MAX- y MIN-.

```

@media (width: 32vw){
/* Reglas CSS aplicables para estas circunstancias */
}
@media (min-width: 64vw){
/* Reglas CSS aplicables para estas circunstancias */
}
@media (max-width: 96vw){
/* Reglas CSS aplicables para estas circunstancias */
}

```

## FUNCIONES

A continuación, se muestra una descripción, más o menos detallada, de las funciones CSS más frecuentemente utilizadas en páginas web, a excepción de las funciones de filtro y transformaciones que se verán en un capítulo dedicado más adelante.

### Funciones de pseudo-elementos

#### FUNCIÓN ATTR

El término ATTR es la abreviatura de ATTRIBUTE y tiene como objetivo devolver el valor descrito por el atributo indicado. Es muy frecuente utilizarlo para mostrar datos guardados en atributos personalizados DATA.

#### Ejemplo:

```
div::before { content: attr(data-value); }
```

Cabe destacar que, aunque esta función resulta ser experimental y sólo es posible utilizarla en la propiedad CONTENT de los pseudo-elementos ::BEFORE y ::AFTER, puede llegar a ofrecer muchas funcionalidades y/o posibilidades.

#### FUNCIÓN COUNTER

La función COUNTER permite recuperar el valor actual de un contador CSS. Aunque los contadores CSS pueden considerarse variables CSS, su manipulación se realiza de forma muy distinta. Esta manipulación se realiza a través de las propiedades COUNTER-RESET y COUNTER-INCREMENT, las cuales permiten reiniciar una variable y e incrementar su valor, respectivamente.

```
ul.falso-ol { counter-reset: indice; }
ul.falso-ol li { counter-increment: indice; }
```

Los contadores CSS pueden ser muy útiles cuando se desea representar una lista de objetos que utilizan una estructura que no está basada en listas de HTML. Por ejemplo, podría darse el caso de que se quisiera presentar un listado construido a partir de elementos gramáticos ARTICLE en donde, cada uno de ellos tiene definido una imagen, un título, un texto introductorio y un enlace.

La forma de utilizar la función COUNTER es la siguiente:

### Ejemplo:

```
ul.falso-ol li::before { content: counter(indice); }
```

## Funciones de cálculo

### FUNCIÓN CALC

El término CALC es la abreviatura de CALCULATE y tiene como objetivo realizar operaciones matemáticas como sumas, restas, multiplicaciones y divisiones. Puede resultar muy interesante en aquellas ocasiones en donde el control del espacio disponible es muy complejo o cuando se desea que tenga una proporción determinada. No obstante, el segundo operando siempre debe ser en píxeles, em o rem.

```
div { width: calc(100% - 20px); }
section { height: calc(50% - 16px); }
article { top: calc(50% - 16px); }
header { left: calc(2% - 2px); }
aside { right: calc(5% - 0.5em); }
span { bottom: calc(100% - 2em); }
nav { border-width: calc(100% - 2em); }
ol { border-radius: calc(50% - 10px); }
ul { padding-top: calc(50% - 32px); }
dt { margin-left: calc(25% - 1px); }
```

Entre las peculiaridades que presenta esta función, cabe destacar que, todos sus operandos deben llevar asociada una unidad de medida junto al valor y que deben estar separados del operando mediante un espacio.

## Funciones gráficas

### FUNCIÓN LINEAR-GRADIENT

La función LINEAR-GRADIENT tiene como objetivo crear transiciones suaves y progresivas a lo largo de una línea que viene definida a través de una dirección o ángulo entre dos o más colores separados por coma. Las transiciones lineales, también conocidas como degradados o gradientes lineales, pueden ser utilizadas en las propiedades BACKGROUND, BACKGROUND-IMAGE, BORDER-IMAGE y LIST-STYLE-IMAGE.

Para establecer los posibles valores de ángulo o dirección, CSS dispone de varias posibilidades. Una de ellas es especificar el lado o esquina desde donde empezará la transición. Esto es posible realizarlo a través de la palabra clave TO, seguida de una de las palabras clave LEFT, TOP, RIGHT o BOTTOM.

```
div { background-image: linear-gradient(to right, black, transparent); }
```

Otra de las formas se definir la dirección o ángulo es a través de una de las unidades de medida de ángulos que maneja CSS. Esto es:

- **DEG:** Unidad de medida que representa un valor en grados. Sus posibles valores van de 0 a 360 con cualquier valor decimal.
- **GRAD:** Unidad de medida que representa un valor en grados centesimales. Sus posibles valores van de 0 a 400 con cualquier valor decimal.
- **RAD:** Unidad de medida que representa un valor en radianes. Sus posibles valores van desde 0 a  $2\pi$  (aproximadamente 6.283184), o múltiplos del mismo.

- **TURN:** Unidad de medida que representa un valor en número de vueltas. Sus posibles valores van de 0 a 1 con cualquier valor decimal.

EQUIVALENCIA	DEG	GRAD	TURN	RAD
	0	0	0	0
	90	100	0.25	1.5708
	180	200	0.50	3.1416
	270	300	0.75	4.7124

Si nos fijamos en la tabla anterior, podremos ver que todas las unidades de medida avanzan en el sentido de las agujas de un reloj. Por tanto, si lo que se desea es avanzar en sentido contrario, los valores deberán ser negativos.

```
div { border-image: linear-gradient(45deg, black, transparent); }
div { border-image: linear-gradient(50grad, black, transparent); }
div { border-image: linear-gradient(0.125turn, black, transparent); }
div { border-image: linear-gradient(0.7854rad, black, transparent); }
```

Por último, sólo hay que destacar que el parámetro de dirección o ángulo es opcional y que, de omitirse, el degradado será vertical de arriba hacia abajo.

## FUNCTION REPEATING-LINEAR-GRADIENT

La función REPEATING-LINEAR-GRADIENT tiene como objetivo crear transiciones repetitivas de tipo lineal entre dos o más colores. Al igual que la función LINEAR-GRADIENT, este tipo de transiciones o degradados es posible utilizarlos en las propiedades BACKGROUND, BACKGROUND-IMAGE, BORDER-IMAGE y LIST-STYLE-IMAGE.

La única diferencia que añade en sus sintaxis es que usa un parámetro adicional que indica el espacio o tamaño al que se debe aplicar. Estos valores pueden ser establecidos en cualquiera de las unidades de medida de longitud permitidas por CSS.

```
div {
background-image: repeating-linear-gradient(
45deg,
black 45px, transparent 47px, gray 60px);
}
/* El resultado debería ser algo como: */
```



## FUNCTION RADIAL-GRADIENT

La función RADIAL-GRADIENT tiene como objetivo crear transiciones suaves y progresivas circulares o elípticas, a partir de un centro, una posición y un contorno, entre dos o más colores. Este tipo de transiciones o degradados es posible utilizarlos en las propiedades BACKGROUND, BACKGROUND-IMAGE, BORDER-IMAGE y LIST-STYLE-IMAGE.

Los degradados radiales crean, por tanto, una transición que empieza en un punto central situado en una posición concreta y a lo largo y ancho del elemento contenedor. Al igual que sucede con su variante LINEAR-GRADIENT, los valores se separan a través de comas.

Las transiciones radiales se alimentan de tres parámetros, aunque no dispone de límite. La forma y posición, color y tamaño para el punto de inicio y color y tamaño para el punto de parada. Los siguientes parámetros serán los siguientes puntos de parada.

Si, en el parámetro primer parámetro, se omite la forma y posición, la transición que se asignará al elemento será en base a las proporciones del elemento contenedor, es decir, si el elemento es cuadrado, el degradado será circular, pero si el elemento tiene forma rectangular, el degradado será elíptico.

```
div { background: radial-gradient(  
#FFFFFF 5px, #CCCCCC 50px,  
#AAAAAA 100px, #oooooo 200px); }  
/* El resultado debería ser algo como: */
```



Si, en el parámetro primer parámetro, se omite sólo la posición, la transición que se asignará al elemento será la que se defina por parámetro, es decir, CIRCLE, ELLIPSE, CLOSEST-CORNER, FARTHEST-CORNER, CLOSEST-SIDE o FARTHEST-SIDE, pero la transición o degradado comenzará en el punto central del elemento contenedor.

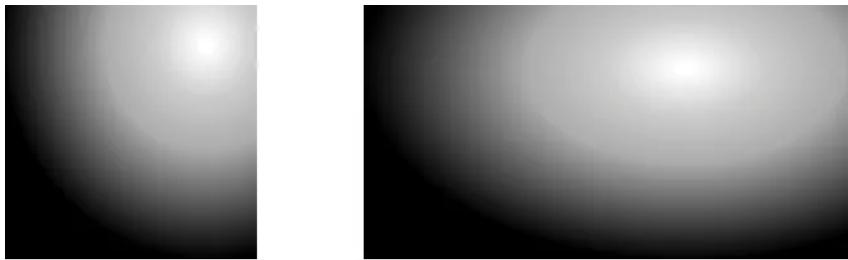
Si, en el parámetro primer parámetro, se especifican tanto la forma, como la posición, la transición que se asignará al elemento será la que se defina por su parámetro forma, es decir, CIRCLE, ELLIPSE, CLOSEST-CORNER, FARTHEST-CORNER, CLOSEST-SIDE o FARTHEST-SIDE, y empezando en el punto indicado por la palabra clave AT.

```
div { background: radial-gradient(  
ellipse at 0 0,  
#FFFFFF 5px,  
#CCCCCC 50px,  
#AAAAAA 100px,  
#oooooo 200px); }  
/* El resultado debería ser algo como: */
```



```
div { background: radial-gradient(  
ellipse farthest-corner at 200px 40px,  
#FFFFFF 5px,  
#CCCCCC 50px,  
#AAAAAA 100px,  
#oooooo 200px); }
```

```
/* El resultado debería ser algo como: */
```

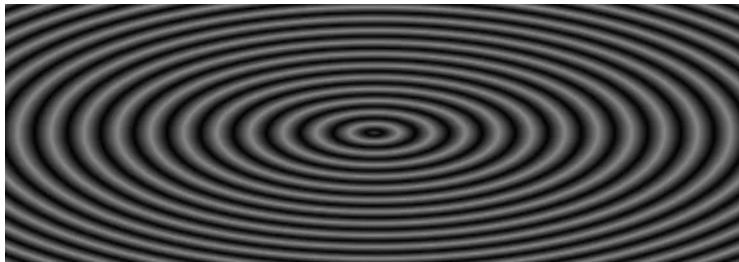


## FUNCIÓN REPEATING-RADIAL-GRADIENT

La función REPEATING-LINEAR-GRADIENT tiene como objetivo crear transiciones repetitivas de tipo radial entre dos o más colores. Al igual que la función RADIAL-GRADIENT, este tipo de transiciones o degradados es posible utilizarlos en las propiedades BACKGROUND, BACKGROUND-IMAGE, BORDER-IMAGE y LIST-STYLE-IMAGE.

La única diferencia en sus sintaxis es que los degradados se van acumulando unos encima de otros en función de sus valores de tamaño, los cuales pueden ser establecidos en cualquiera de las unidades de medida de longitud permitidas por CSS.

```
div {  
background-image: repeating-radial-gradient(  
closest-side,  
#000000 5px, #888888 15px, #888000 25px);  
}  
/* El resultado debería ser algo como: */
```

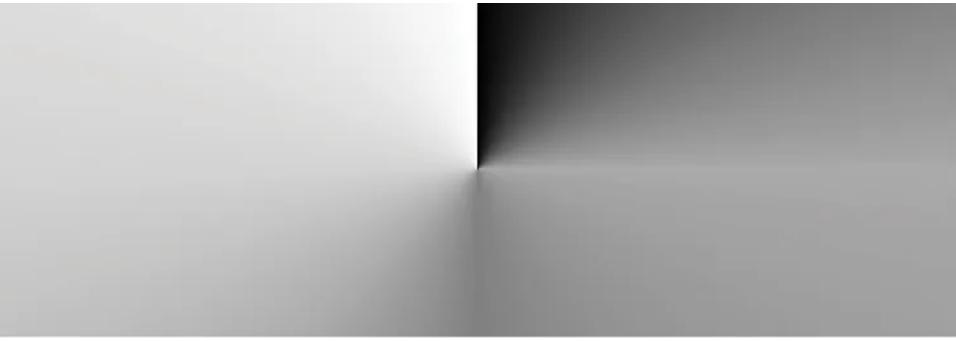


## FUNCIÓN CONIC-GRADIENT

La función CONIC-GRADIENT tiene como objetivo crear transiciones de tipo cónico entre dos o más colores. Al igual que sus análogas, este tipo de transiciones es posible utilizarla en las propiedades BACKGROUND, BACKGROUND-IMAGE, BORDER-IMAGE y LIST-STYLE-IMAGE.

Su sintaxis es idéntica a LINEAR-GRADIENT y su aplicación idéntica a RADIAL-GRADIENT, con la diferencia de que la transición se aplica con respecto a un cono.

```
div {  
background: conic-gradient(black, darkgray, gray, lightgray, white);  
}  
/* El resultado debería ser algo como: */
```



```
div { background: conic-gradient( black 0deg, #333 0deg 10deg, #666 10deg  
20deg, #999 40deg 120deg, #ccc 120deg 200deg, white 400deg); }  
/* El resultado debería ser algo como: */
```



## HACKS

El término hack significa “corte” o “hachazo” y en CSS se usa para discernir el agente de usuario que ejecuta la página o aplicación web. La distinción del agente de usuario se realiza a partir de un prefijo de codificación que se suele corresponder con el motor que utiliza y/o su versión.

El objetivo de esta distinción es aplicar un marcado CSS específico para conseguir que los documentos o páginas web se vean exactamente igual, independientemente del navegador o herramienta de asistencia que se utilice.

### NOTA

Que una página web pueda funcionar correctamente, independientemente del agente de usuario que utilice el usuario, es a lo que se refieren los diseñadores y maquetadores cuando suelen hacer referencia al concepto de Cross Browsing.

Aunque existen multitud de hacks o prefijos, en el mundo web únicamente se suelen utilizar cuatro.

- **-MS-**: Es el prefijo que aplica a Microsoft Internet Explorer.
- **-MOZ-**: Es el prefijo que aplica a Firefox y navegadores basados en Gecko.
- **-O-**: Es el prefijo que aplica a Opera.
- **-WEBKIT-**: Es el prefijo que aplica a Chrome Safari y navegadores basados en Webkit y/o Blink.

Cabe destacar que sólo existen unas pocas propiedades que admitan estos prefijos y que conocerlas y recordarlas puede ser algo complicado si no se usan con cierta frecuencia, pero, por suerte, todas ellas están descritas en la página de W3SCHOOLS - CSS3 SOPORTE A NAVEGADORES:

[https://www.w3schools.com/cssref/css3\\_browsersupport.asp](https://www.w3schools.com/cssref/css3_browsersupport.asp).

## CSS Reference With Browser Support

The table below lists all CSS properties and how each property is supported in the different browsers:

The number to the right of the browser icon indicates in which browser version the property was first supported.

Property	Edge	Firefox	Chrome	Safari	Opera
<b>A</b>					
align-content	11	28	21	9	12.1
align-items	11	20	21	9	12.1
align-self	11	20	21	9	12.1
all		27	37		24
animation	10	16	43	9	30

Captura de W3Schools.

También es importante aclarar que, aunque los hacks suelen hacer referencia a declaraciones específicas basadas en estos prefijos, también pueden hacer referencia a consultas de medios que sólo interpreta un agente de usuario determinado. A continuación, se muestran las más utilizadas, sin embargo, existen gran cantidad de variaciones y combinaciones.

Todas ellas pueden ser consultadas en <http://browserhacks.com/>.

The screenshot shows the homepage of BrowserHacks. At the top, there's a navigation bar with links to contribute on GitHub and follow them on Twitter (@browserhacks). Below the header, there are four main sections: "What's this?", "How to?", "Reminder!", and "Powered by:". The "What's this?" section explains what browser hacks are and how to use them. The "How to?" section provides a step-by-step guide. The "Reminder!" section cautions against using hacks as they might not always be perfect solutions. The "Powered by:" section lists partners like Slack and BrowserStack. At the bottom, there are checkboxes for "Show legacy hacks" and "Enable tests", and a search bar for "Browser search (e.g. IE 6)".

Captura de BrowserHacks.

```
/* Firefox CSS hacks */
@media screen and (min--moz-device-pixel-ratio:0) {} {
/* Reglas únicamente para Firefox 4 o superior */
}

/* Chrome CSS hacks */
@media screen and (-webkit-min-device-pixel-ratio:0) {
/* Reglas únicamente para Chrome 29 o superior */
}

/* Opera CSS hacks */
@media (min-resolution: .001dpcm) {
/* Reglas únicamente para Opera 12 o superior */
}

/* Safari CSS hacks */
@media \\\o screen {
/* Reglas únicamente para Safari 7 o superior */
}

/* Internet Explorer y Edge CSS hacks */
@media screen and (-ms-high-contrast: active), (-ms-high-contrast: none) {
/* Reglas únicamente para IE 10 o superior y Edge */
}
```

## **SELECTORES**

A continuación, se muestra una descripción, más o menos detallada, con los selectores CSS más frecuentemente utilizados en documentos y páginas web.

### **Simples y combinados**

#### **SELECTOR UNIVERSAL**

El carácter “asterisco” es un selector universal que selecciona todos los elementos del documento.

**Ejemplo:** Aplicar a todos los elementos un color negro.

```
* { color: black; }
```

#### **SELECTOR DE TIPO**

Permite seleccionar todos los elementos que se correspondan con un nombre de etiqueta HTML.

**Ejemplo:** Aplicar a todos los elementos DIV del documento un color de borde verde.

```
div { border-color: #00FF00; }
```

#### **SELECTOR DE CLASE O PUNTO**

El carácter “punto” selecciona todos los elementos que tengan como clase el identificador indicado detrás del punto.

**Ejemplo:** Aplicar a todos los elementos que contengan la clase “items-group” un color negro.

```
.items-group { color: #0000; }
```

#### **SELECTOR DE ID O ALMOHADILLA**

El símbolo “almohadilla” permite seleccionar el elemento que tenga como ID el identificador indicado detrás de la almohadilla. Aunque no provoca error que haya dos elementos con el mismo ID, es importante controlarlo para no obtener resultados imprevisibles.

**Ejemplo:** Aplicar al elemento “banner” un color de fondo negro y un color textual en blanco.

```
#banner { background: black; color: white; }
```

#### **SELECTOR ESPACIO**

El carácter “espacio” permite seleccionar todos los elementos que estén contenidos en los elementos que se correspondan con el selector previamente declarado.

**Ejemplo:** Aplicar a todos los elementos P que estén dentro de elementos DIV un tamaño de fuente de 14 píxeles.

```
div p { font-size: 14px; }
```

#### **SELECTOR COMA**

El carácter “coma” permite seleccionar todos los elementos que resulten de combinar los selectores de forma independiente.

**Ejemplo:** Aplicar a todos los elementos DIV y SPAN del documento un color rojo.

```
div, span { color: red; }
```

## **SELECTOR MAYOR QUE**

El símbolo “mayor que” permite seleccionar todos los elementos que estén contenidos en los elementos que se correspondan con el selector previamente declarado y que sean hijos directos.

### **Ejemplo:**

Aplicar a los hijos P directos de elementos DIV un ancho del 100%.

```
div > p { width: 100%; }
```

## **SELECTOR SUMA**

El símbolo “suma” permite seleccionar todos los elementos que resulten del selector de la derecha y que estén declarados justo después del de los elementos seleccionados por el selector predecesor.

**Ejemplo:** Aplicar a los elementos BUTTON que estén definidos justo después de un elemento INPUT un borde de 2 píxeles con estilo sólido y color gris.

```
input + button { border: 2px solid gray; }
```

## **SELECTOR VIRGULILLA**

Permite seleccionar todos los elementos que sean hermanos del elemento previamente declarado.

**Ejemplo:** Aplicar a todos los elementos LABEL que tengan como hermano a un elemento DIV un color de borde verde.

```
div ~ label { border-color: #00FF00; }
```

## **SELECTOR [ATRIBUTO]**

Permite seleccionar los elementos que tengan definido el atributo establecido entre corchetes.

**Ejemplo:** Aplicar a todos los elementos SPAN que tengan establecido el atributo ROLE a un ancho del 100%.

```
span[role] { width: 100%; }
```

## **SELECTOR [ATRIBUTO=VALOR]**

Permite seleccionar los elementos que tengan definido el atributo establecido entre corchetes y cuyo valor sea el “valor” indicado.

**Ejemplo:** Aplicar a todos los elementos INPUT de tipo imagen del documento una altura de 32 píxeles.

```
input[type="image"] { height: 32px; }
```

## **SELECTOR [ATRIBUTO^=VALOR]**

Permite seleccionar los elementos que tengan definido el atributo establecido entre corchetes y cuyo valor empiece por el “valor” indicado.

**Ejemplo:** Aplicar a todos los elementos INPUT que empiecen la definición del atributo CLASS con el patrón FORM una altura de 32 píxeles.

```
input[class^="form"] { height: 32px; }
```

Esto seleccionaría, por ejemplo, todos los elementos que tengan el atributo CLASS establecido a FORM-GROUP, FORM-CONTROL, pero no ACTION-FORM.

## **SELECTOR [ATRIBUTO\*=VALOR]**

Permite seleccionar los elementos que tengan definido el atributo establecido entre corchetes y cuyo valor contengan el “valor” indicado.

**Ejemplo:** Aplicar a todos los elementos INPUT que contengan en el atributo CLASS el patrón FORM una altura de 32 píxeles.

```
input[class*="form"] { height: 32px; }
```

Esto seleccionaría, por ejemplo, todos los elementos que tengan el atributo CLASS establecido a FORM-GROUP, FORM-CONTROL y ACTION-FORM.

## **SELECTOR [ATRIBUTO\$=VALOR]**

Permite seleccionar los elementos que tengan definido el atributo establecido entre corchetes y cuyo valor termine por el “valor” indicado.

**Ejemplo:** Aplicar a todos los elementos INPUT que terminen la definición del atributo CLASS con el patrón FORM una altura de 32 píxeles.

```
input[class$="form"] { height: 32px; }
```

Esto seleccionaría, por ejemplo, todos los elementos que tengan el atributo CLASS establecido a ACTION-FORM, pero no FORM-CONTROL y FORM-GROUP.

## **Pseudo-clases**

Las pseudo-clases son un tipo de selector que tiene CSS y que sirven para dar funcionalidad a los elementos en función de su estado. A continuación, se explican la mayoría de ellas.

### **SELECTOR :ACTIVE**

Permite seleccionar todos los elementos de tipo enlace que estén activos.

**Ejemplo:** Aplicar a todos los elementos A activos un color de texto azul.

```
a:active { color: #003366; }
```

### **SELECTOR :CHECKED**

Permite seleccionar todos los elementos de tipo RADIO, CHECKBOX u OPTION que estén chequeados o conmutados, es decir, que los elementos de tipo RADIO, CHECKBOX tengan establecida la propiedad CHECKED o que los elementos OPTION de un elemento SELECT tengan establecida la propiedad SELECTED.

**Ejemplo:** Aplicar a todos los elementos RADIO chequeados un color de fondo negro.

```
input[type="radio"]:checked { background: #000; }
```

RADIO RESULTADO

(•) Sí

( ) No

CHECKBOX

[■] Acepta los Términos de uso

[■] Acepta la Política de Privacidad

[ ] Quiero recibir ofertas y novedades en mi correo personal

SELECT

OPTION

OPTION SELECTED

Cabe destacar que, aunque puede no afectar, cuando se desea crear un sitio totalmente accesible no es recomendable cambiar el aspecto de los RADIO, CHECKBOX u OPTION.

### **SELECTOR :DISABLE**

Permite seleccionar todos los elementos que estén deshabilitados, es decir, que tengan establecida la propiedad DISABLED.

**Ejemplo:** Aplicar a los elementos INPUT deshabilitados un color de fondo gris claro con texto gris oscuro.

```
input:disabled{ background: #eee; color: #999; }
```

### **SELECTOR :EMPTY**

Permite seleccionar todos los elementos que estén totalmente vacíos, es decir, que no tengan ni espacios, ni texto, ni hijos.

**Ejemplo:** Aplicar a los elementos SPAN que estén vacíos un color de fondo negro con borde blanco.

```
span:empty{ background: #000; border: 1px solid #fff; }
```

#### **UL RESULTADO**

```
<LI></LI>  
<LI>  
<input ... />  
</LI>
```

### **SELECTOR :ENABLE**

Permite seleccionar todos los elementos que estén habilitados, es decir, que no tengan establecida la propiedad DISABLED.

**Ejemplo:** Aplicar a los elementos INPUT no deshabilitados un color de fondo negro con borde blanco.

```
input:enabled{ background: #000; border: 1px solid #fff; }
```

### **SELECTOR :FIRST-CHILD**

Permite seleccionar los elementos que sean el primer descendiente del elemento especificado. Si no indica elemento, se seleccionan todos los primeros descendientes.

**Ejemplo:** Supongamos una lista desordenada UL con cinco elementos LI y que lo que se desea es aplicar un estilo de negrita al primer elemento de cada elemento UL. Lo que se podría hacer es:

```
ul li:first-child { font-weight: bold; }
```

#### **UL RESULTADO**

**Elemento 1**  
Elemento 2  
Elemento 3  
Elemento 4  
Elemento 5

### **SELECTOR :FIRST-OF-TYPE**

Permite seleccionar los elementos que sean el primer descendiente del elemento y tipo (nombre de etiqueta) especificado. Si no indica elemento, se seleccionan todos los primeros descendientes.

La diferencia entre :FIRST-CHILD y :FIRST-OF-TYPE es que :FIRST-OF-TYPE seleccionará aquellos elementos que puede que no sean el primer elemento del elemento padre, es decir, puede que el elemento P sea el número dos o más del elemento padre, pero es el primero de ese tipo.

**Ejemplo:** Supongamos un contenedor DIV con cinco elementos dispuestos como [P, SPAN, P, P, SPAN] y que lo que se desea es aplicar un estilo de negrita al primer elemento de tipo SPAN. Lo que se podría hacer es:

```
div span:first-of-type { font-weight: bold; }
```

#### DIV RESULTADO

Elemento P

#### Elemento SPAN

Elemento P

Elemento P

Elemento SPAN

## SELECTOR :FOCUS

Permite seleccionar el elemento que posea el foco de teclado.

**Ejemplo:** Aplicar a todos los elementos INPUT, BUTTON y A que posean el foco un color de texto blanco con fondo azul.

```
a:focus,  
input:focus,  
button:focus { background: #003366; color: #FFFFFF; }
```

## SELECTOR :HOVER

Permite seleccionar todos los elementos que estén por dentro del elemento por el que se pasa el puntero del ratón.

**Ejemplo:** Aplicar a todos los elementos INPUT, BUTTON y A un color de texto blanco con fondo azul cuando se pasa el ratón por encima.

```
a:hover, input:hover, button:hover { background: #003366; color: #FFFFFF; }
```

## SELECTOR :INVALID

Permite seleccionar todos los elementos que no pasen la validación requerida por HTML5.

El selector :INVALID sólo funciona cuando los elementos tienen alguna limitación como que sea requerido, que tenga un mínimo o máximo establecido o que sea de un campo de tipo especial como el email o number. Por tanto, si un elemento INPUT tiene el atributo REQUIRED establecido y se define una regla que contempla esta pseudo-clase, se aplicará de manera automática.

**Ejemplo:** Aplicar a los elementos INPUT que no sean válidos un color de fondo rojo puro.

```
input:invalid { background: red; }
```

## SELECTOR :LAST-CHILD

Permite seleccionar los elementos que sean el último descendiente del elemento especificado. Si no indica elemento, se seleccionan todos los últimos descendientes.

**Ejemplo:** Supongamos una lista desordenada UL con cinco elementos LI y que lo que se desea es aplicar un estilo de negrita al último elemento de cada elemento UL. Lo que se podría hacer es:

```
ul li:last-child { font-weight: bold; }
```

#### Elemento 1 RESULTADO

Elemento 2

Elemento 3

Elemento 4

Elemento 5

## **SELECTOR :LAST-OF-TYPE**

Permite seleccionar los elementos que sean el último descendiente del elemento y tipo (nombre de etiqueta) especificado. Si no indica elemento, se seleccionan todos los últimos descendientes.

La diferencia entre :LAST-CHILD y :LAST-OF-TYPE es que :LAST-OF-TYPE seleccionará aquellos elementos que puede que no sean el último elemento del elemento padre, es decir, puede que el elemento P sea el penúltimo o uno anterior del elemento padre, pero es el último de ese tipo.

**Ejemplo:** Supongamos un contenedor DIV con cinco elementos dispuestos como [P, SPAN, P, P, SPAN] y que lo que se desea es aplicar un estilo de negrita al último elemento de tipo P. Lo que se podría hacer es:

```
div p:last-of-type { font-weight: bold; }
```

### **DIV RESULTADO**

Elemento P  
Elemento SPAN  
Elemento P  
**Elemento P**  
Elemento SPAN

## **SELECTOR :LINK**

Permite seleccionar todos los elementos de tipo enlace no visitados.

**Ejemplo:** Aplicar a todos los elementos A no visitados un color de texto rojo y sin subrayado.

```
a:link {  
color: red;  
text-decoration: none;  
}
```

Cabe destacar que, por temas de accesibilidad, no es recomendable cambiar el aspecto de los enlaces, aunque esta acción no se realice casi nunca.

## **SELECTOR :OPTIONAL**

Permite seleccionar todos los elementos de tipo INPUT, SELECT y TEXTAREA que sean opcionales, es decir, que no tengan establecido la propiedad REQUIRED.

**Ejemplo:** Aplicar a todos los elementos INPUT de tipo texto opcionales un color fondo dorado y texto negro.

```
input:optional { background: gold; color: #000; }
```

## **SELECTOR :ONLY-CHILD**

Permite seleccionar los elementos que sean el único descendiente entre los elementos del mismo nivel.

**Ejemplo:** Supongamos dos listas desordenada UL, una con un único elemento LI y otra con dos elementos LI. Lo que se desea es aplicar un estilo de negrita a las lista que sólo tienen un único hijo. Lo que se podría hacer es:

```
ul li:only-child { font-weight: bold; }
```

### **Lista 1 RESULTADO**

**Elemento 1**  
Lista 2  
Elemento 1  
Elemento 2

## **SELECTOR :ONLY-OF-TYPE**

Permite seleccionar los elementos que sean el único descendiente de ese tipo o nombre de etiqueta entre los elementos del mismo nivel.

**Ejemplo:** Supongamos dos contenedores DIV, ambos con único elemento, pero en uno el hijo es un elemento SPAN y, en el otro, es un elemento P. Lo que se desea es aplicar un estilo de negrita al elemento del contenedor que sólo tenga un único hijo y que sea de tipo SPAN. Lo que se podría hacer es:

```
div span:only-of-type { font-weight: bold; }
```

DIV RESULTADO

SPAN

DIV

P

### SELECTOR :NOT(SELECTOR)

Permite seleccionar todos los elementos que NO coincidan con el selector indicado entre paréntesis.

**Ejemplo:** Aplicar a todos los elementos que sean A un color de texto azul y, para aquellos que no sean A un color de texto negro.

```
a { color: blue; }  
*:not(a) { color: black; }
```

Cabe destacar que el valor dentro de los paréntesis puede ser cualquier selector de los anteriores, es decir, puede ser una etiqueta, una clase, un identificador, una pseudo-clase, un pseudo-elemento o una combinación de ellos.

### SELECTOR :NTH-CHILD(N)

Permite seleccionar los elementos que sean el número de descendiente indicado por el valor de N, empezando por 1.

Cuando se trabaja con este selector, se pueden utilizar valores de palabra clave (también llamados valores predefinidos) y de notación funcional (también llamados valores calculados). Los valores de palabra clave son las palabras reservadas EVEN y ODD, que representan a los valores pares e impares respectivamente.

Los valores de notación funcional utilizan el formato de expresión AN + B, que representa la serie de valores que resultan de multiplicar un factor N por A y sumarle el valor de B. Dicho de otro modo, el valor del patrón AN + B representa a los elementos que resulten de realizar la división entera de A más el offset entero B. Por tanto, el patrón 2N simboliza todos los elementos pares y, 3N+1, simboliza los elementos que estén definidos en las posiciones 1, 4, 7, ...

**Ejemplo:** Supongamos una lista desordenada UL con cinco elementos LI y que lo que se desea es aplicar un estilo de negrita al tercer elemento de cada elemento UL. Lo que se podría hacer es:

```
li:nth-child(3) { font-weight: bold; }  
Elemento 1 RESULTADO  
Elemento 2  
Elemento 3  
Elemento 4  
Elemento 5
```

### SELECTOR :NTH-LAST-CHILD(N)

Permite seleccionar los elementos que sean el número de ascendiente indicado por el valor de N empezando desde el final y con el valor 1.

A modo aclaratorio, el valor de N igual a 1 se corresponderá con el último descendiente del elemento padre, el valor de N igual a 2, se corresponderá con el penúltimo descendiente, y así sucesivamente.

Cuando se trabaja con este selector, se pueden utilizar valores de palabra clave (también llamados valores predefinidos) y de notación funcional (también llamados valores calculados). Los valores de palabra clave son las

palabras reservadas EVEN y ODD, que representan a los valores pares e impares respectivamente.

Los valores de notación funcional utilizan el formato de expresión AN + B, que representa la serie de valores que resultan de multiplicar un factor N por A y sumarle el valor de B. Dicho de otro modo, el valor del patrón AN + B representa a los elementos que resulten de realizar la división entera de A más el offset entero B. Por tanto, el patrón 2N simboliza todos los elementos pares y, 3N+1, simboliza los elementos que estén definidos en las posiciones 1, 4, 7, ...

**Ejemplo:** Supongamos una lista desordenada UL con cinco elementos LI y que lo que se desea es aplicar un estilo de negrita al penúltimo elemento de cada elemento UL. Lo que se podría hacer es:

```
li:nth-last-child(2) { font-weight: bold; }  
Elemento 1 RESULTADO  
Elemento 2  
Elemento 3  
Elemento 4  
Elemento 5
```

### SELECTOR :NTH-LAST-OF-TYPE(N)

Selecciona los elementos que sean del tipo especificado y cuya posición sea el valor indicado por N empezando desde el final, con el valor 1.

A modo aclaratorio, el valor de N igual a 1 se corresponderá con el último descendiente del elemento padre que tenga ese tipo (nombre de etiqueta), que no tiene por qué ser el último descendiente.

Cuando se trabaja con este selector, se pueden utilizar valores de palabra clave (también llamados valores predefinidos) y de notación funcional (también llamados valores calculados). Los valores de palabra clave son las palabras reservadas EVEN y ODD, que representan a los valores pares e impares respectivamente.

Los valores de notación funcional utilizan el formato de expresión AN + B, que representa la serie de valores que resultan de multiplicar un factor N por A y sumarle el valor de B. Dicho de otro modo, el valor del patrón AN + B representa a los elementos que resulten de realizar la división entera de A más el offset entero B. Por tanto, el patrón 2N simboliza todos los elementos pares y, 3N+1, simboliza los elementos que estén definidos en las posiciones 1, 4, 7, ...

**Ejemplo:** Supongamos un contenedor DIV con cinco elementos dispuestos como [P, SPAN, P, P, SPAN] y que lo que se desea es aplicar un estilo de negrita al penúltimo elemento de tipo P. Lo que se podría hacer es:

```
div p:nth-last-of-type(2) { font-weight: bold; }  
DIV RESULTADO  
Elemento P  
Elemento SPAN  
Elemento P  
Elemento P  
Elemento SPAN
```

### SELECTOR :READ-ONLY

Permite seleccionar todos los elementos que tengan el atributo READONLY establecido entre sus atributos HTML, es decir, que sean de sólo lectura.

**Ejemplo:** Aplicar a todos los elementos INPUT que sean de sólo lectura un color de fondo lima.

```
span:read-only { background: lime; }
```

## **SELECTOR :REQUIRED**

Permite seleccionar todos los elementos que tengan el atributo REQUIRED establecido entre sus atributos HTML, es decir, que tengan un carácter obligatorio.

**Ejemplo:** Aplicar a todos los elementos INPUT que sean requeridos u obligatorios y sean inválidos un color de fondo naranja.

```
span:invalid:required { background: orange; }
```

## **SELECTOR :ROOT**

Permite seleccionar el elemento raíz del documento, al igual que el elemento HTML, pero con la diferencia de que, la pseudo-clase :ROOT tiene mucho más peso y relevancia. Por ejemplo, mientras que el selector HTML no se suele utilizar para definir variables CSS, la pseudo-clase :ROOT sí.

**Ejemplo:** Definir una variable bg-color que contenga el color rojo para que pueda ser utilizada por todas las reglas declaradas en el documento.

```
:root { --bg-color: #Foo; }
```

## **SELECTOR :TARGET**

Las direcciones o URL que comienzan con una almohadilla representan un identificador de anclaje a un determinado elemento dentro de un documento. Cuando el usuario pulsa en un elemento A que posee un HREF con una almohadilla, se lanza un evento que pone el foco en el elemento que tiene ese ID. Cuando eso sucede se activa la pseudo-clase :TARGET.

Por tanto, la pseudo-clase :TARGET selecciona el elemento destino cuando el usuario pulsa en un elemento que vincula dicho elemento con el punto de anclaje.

**Ejemplo:** Imaginemos un elemento A que apunta a un DIV que posee el mismo ID que el referenciado por el atributo HREF y que está oculto por previa declaración de CSS.

```
<style>
div { display: none }
</style>
<a href="#enlace">Enlace</a>
<div id="enlace">
<h2>Enlace</h2>
<p>Un contenido cualquiera</p>
</div>
```

Si ejecutásemos el código anterior en un navegador, lo que se vería es sólo el enlace y si pulsásemos en dicho enlace, no mostraría nada, únicamente provocaría una navegación hacia el punto de anclaje. Para hacerlo visible cuando se pulse el enlace podríamos recurrir a una funcionalidad en JavaScript, pero no, es más sencillo a través de la pseudo-clase :TARGET.

```
div:target { display: block; }
```

Ahora sí, si añadimos esta regla al código anterior, y pulsamos en el enlace, veremos que, además de provocar una navegación, se muestra el DIV con todo su contenido.

## **SELECTOR :VALID**

Permite seleccionar todos los elementos que pasen la validación requerida por HTML5 y JavaScript.

El selector :VALID sólo funciona cuando los elementos tienen alguna limitación como que sea requerido, que tenga un mínimo o máximo establecido, que sea de un campo de tipo especial como el email o sólo numérico. Por tanto, si un elemento INPUT no tiene el atributo REQUIRED establecido, no realiza ningún otro tipo de validación, y se define una regla que contempla esta pseudo-clase, se aplicará de manera automática.

**Ejemplo:** Aplicar a los elementos INPUT que sean válidos un color de fondo verde claro.

```
input:valid { background: lightgreen; }
```

## SELECTOR :VISITED

Permite seleccionar todos los elementos de tipo enlace visitados.

**Ejemplo:** Aplicar a todos los elementos A visitados un color de texto verde.

```
a:visited { color: green; }
```

Cabe destacar que, por temas de accesibilidad, no es recomendable cambiar el aspecto de los enlaces, aunque esta acción no se realice casi nunca.

## Pseudo-elementos

Los pseudo-elementos son un tipo de selector que tiene CSS y que sirven para añadir estilos adicionales en una parte concreta del elemento. Es por ello que se suele afirmar que, mientras que las pseudo-clases están vinculadas a los estados del elemento, los pseudo-elementos están vinculados al aspecto visual.

### SELECTORES ::BEFORE Y ::AFTER

Prácticamente todos los elementos de CSS tienen tres capas de visualización superpuestas, la anterior, la seleccionada y la posterior. Mientras que la capa seleccionada es manipulable a través del propio elemento, la anterior y posterior son manipulables a través de los pseudo-elementos BEFORE y AFTER.

Se dice que los pseudo-elementos BEFORE y AFTER son el primer y último hijo del elemento, respectivamente, sin embargo, aunque sean hijos del elemento, no pertenecen al HTML. Por esta razón, cuando se desea insertar contenido en estas capas se debe recurrir a la propiedad CONTENT, para asignar contenido sin la intervención de HTML.

Aunque pueda parecer que no, el contenido de la propiedad CONTENT puede ser muy diverso. Permite la asignación de una cadena de texto (vacía o no), un código hexadecimal o Unicode, una palabra clave de CSS como OPEN-QUOTE o CLOSE-QUOTE, una función CSS o, incluso, una combinación de todos ellos. Por ejemplo, un uso muy frecuente para los pseudo-selectores BEFORE y AFTER es la asignación de una URL para mostrar un gráfico o ícono.

A continuación, se muestran unos cuantos ejemplos:

```
.elemento::before { content: ""; }
.card::before { content: "\1F340"; }
button i::after { content: url("./icono.png"); }
.list::after { content: counter(index); }
h2 span::before { content: attr(data-value); }
div:lang(es)::before { content: open-quote; }
/* Combinación de funciones, palabras clave y strings */
div::after { content: open-quote " " attr(data-title) " " close-quote; }
```

## SELECTOR ::FIRST-LETTER

Permite seleccionar la primera letra del contenido de cada elemento.

**Ejemplo:** Aplicar a la primera letra de todos los elementos P un tamaño de fuente de 24 píxeles.

```
p:first-letter { font-size: 24px; }
```

## NOTA

Este pseudo-elemento sólo puede utilizar las propiedades referidas a la fuente, color de texto y fondo, márgenes, borde, y propiedades específicas TEXT-DECORATION, VERTICAL-ALIGN, TEXT-TRANSFORM, LINE-HEIGHT, FLOAT y CLEAR.

## SELECTOR ::FIRST-LINE

Permite seleccionar la primera línea del contenido de cada elemento.

**Ejemplo:** Aplicar a la primera línea de todos los elementos P un color textual negro con subrayado.

```
p:first-line { color: black; text-decoration: underline; }
```

## NOTA

Este pseudo-elemento sólo puede utilizar las propiedades referidas a la fuente, color de texto y fondo, márgenes, borde, y propiedades específicas TEXT-DECORATION, VERTICAL-ALIGN, TEXT-TRANSFORM, LINE-HEIGHT, FLOAT y CLEAR.

## SELECTOR ::SELECTION

Permite definir un estilo específico para marcar el texto seleccionado.

**Ejemplo:** Definir que el color de la selección de texto sea en color amarillo con fondo negro.

```
::selection { background: black; color: yellow; }
```

## NOTA

Este pseudo-elemento sólo puede utilizar las BACKGROUND, COLOR, CURSOR y OUTLINE. Además, para que se vuelva totalmente compatible con todos los navegadores se debe recurrir al uso de prefijos CSS o CSS Hacks para que este selector sea aplicado. En general, los tres más utilizados son -WEBKIT-, -MOZ- o -MS-.

## SELECTOR ::PLACEHOLDER

Permite definir un estilo específico para los elementos que admiten el establecimiento del atributo PLACEHOLDER de HTML.

**Ejemplo:** Definir que el color de la selección de texto sea en color amarillo con fondo negro.

```
::placeholder { background: black; color: yellow; }
```

## NOTA

Este pseudo-elemento sólo puede utilizar las BACKGROUND, COLOR, CURSOR y OUTLINE. Además, para que se vuelva totalmente compatible con todos los navegadores se debe recurrir al uso de prefijos CSS o CSS Hacks para que este selector sea aplicado. En general, los tres más utilizados son -WEBKIT-, -MOZ- o -MS-.

## Especificidad de los selectores

La especificidad es la forma mediante la cual los agentes de usuario deciden que tiene más o menos relevancia o peso. Esto es una de las características que hará que una regla se aplique y sobrescriba a otras, aunque esté declarada antes.

Selector	Peso
<b>Tipo</b>	0
<b>Pseudo-elementos</b>	0
<b>Clase</b>	1

<b>Atributo</b>	1
<b>ID</b>	2
<b>Atributo style</b>	3
<b>Palabra clave !important</b>	4
<b>Attributo style con !important dentro</b>	5

Tanto el selector universal, como los combinadores (+, >, ~ o el espacio) y la pseudo-clase :NOT(), no tienen efectividad en lo que respecta a esto pesos, pero sí dentro de la definición de la pseudo-clase :NOT().

Cualquier declaración en línea sobre los elementos HTML sobrescribe los estilos definidos en las hojas de estilo internas o externas, a menos que, las reglas CSS tengan contenidas la palabra clave IMPORTANT.

Aunque, en principio, esta palabra clave no tiene nada que ver con la especificidad, sí que influye en ella anulando prácticamente toda posible herencia. Por esta razón, se ha puesto con el mayor peso en nuestra tabla.

También cabe destacar que, el uso de la palabra clave IMPORTANT no es una buena práctica a nivel de desarrollo profesional, primero porque rompe la estandarización de las reglas en cascada, segundo, porque dificulta la lectura y depuración del código y, tercero, pero no menos importante, porque puede provocar comportamientos no contemplados o deseados en el documento.

## VARIABLES

Hasta no hace tanto, la utilización de variables era una de las limitaciones que presentaba CSS. Sin embargo, en el año 2015, eso empezó a cambiar.

La necesidad de poder definir variables viene ya desde las primeras incorporaciones web a nivel profesional. Si se piensa un poco, es muy frecuente encontrarse con valores repetidos en las hojas de estilo, pero no sólo a nivel de colores, también a otros niveles como puedan ser los márgenes, animaciones o transiciones.

Las variables CSS, más correctamente denominadas **propiedades personalizadas**, pueden ser establecidas o manipuladas por varios métodos, pero su alcance lo decide la característica de herencia.

Para definir una propiedad personalizada se debe utilizar el prefijo -- (doble guion), que le indica al agente de usuario que es una variable. La forma más frecuente de declarar ésta y otras variables es recurriendo a la pseudo-clase :ROOT:

```
:root { --background: whitesmoke; }
```

Si se define una propiedad personalizada dentro de la pseudo-clase :ROOT, podrá ser utilizado por todas las reglas CSS declaradas en el documento. Sin embargo, si se define una propiedad personalizada dentro de, por ejemplo, un selector DIV, podrá ser utilizado tanto por él y todos sus pseudo-elementos, pero no podrá ser utilizado por un LABEL.

```
div {
background: var(--backcolor);
color: var(--forecolor);
padding: 5px;
--forecolor: yellow;
--backcolor: black;
}
div::before{
content:"Texto en amarillo";
position:relative;
width: 100%;
```

```
height: 100%;  
color: var(--backcolor);  
background: var(--forecolor);  
}
```

Como se puede apreciar en la ilustración anterior, para utilizar dichas variables o propiedades personalizadas, se debe utilizar la función VAR().

Sin embargo, como hemos dicho anteriormente, no sólo es posible realizar la declaración de variables CSS a través de CSS. Esto se debe a que, esencialmente, las variables CSS se introducen el DOM del documento pudiendo ser accedidas y manipuladas a través de JavaScript. Esto es posible hacerlo mediante el uso del método setProperty perteneciente a la interfaz STYLE.

```
document.documentElement.style.setProperty('--background', 'whitesmoke');
```

Esto puede ser útil cuando se requiere manipular en tiempo real las variables para realizar, por ejemplo, estilos personalizados para cada usuario. Sólo por entender mejor esta casuística, imaginemos que tenemos una aplicación web en la que se desea que cada usuario pueda definir su tamaño de letra, colores y tipo de fuente, entre otros valores.

En un principio, declararíamos unos estilos por defecto, por si el usuario no tiene definida ninguna personalización.

En concreto algo como:

```
:root {  
--backcolor: #fofofo;  
--forecolor: #003366;  
--fontFamily: Arial, sans-serif;  
--fontSize: 15px;  
}
```

Luego, haríamos una llamada en JavaScript para recuperar la configuración personalizada del usuario y sobrescribiríamos la configuración por defecto.

```
var http = new XMLHttpRequest()  
http.open("GET", './getCustomPersonalization.php')  
http.onreadystatechange = function(){  
if(this.readyState == 4 && this.status == 200){  
var resultado = JSON.parse(this.responseText)  
override(resultado);  
}  
}  
http.send();  
function override(customJSON){  
var style = document.documentElement.style;  
for(key in customJSON){  
style.setProperty('--' + key, customJSON[key]);  
}  
}
```

## PRÁCTICA 3: PRIMEROS ESTILOS DE “UNIVESES”

### Código del ejemplo

<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-03>



## Objetivo de la práctica

Dar formato y una imagen adecuada a la estructura del documento que se creó en la práctica 2, la página de inicio de la web de UniversES.

## Resultado

The screenshot displays the UniversES homepage with a dark header and footer. The main content area is styled with columns and rounded corners. The 'INICIO' section contains two cards with placeholder text and images. The 'COMENTARIOS' section shows a list of comments with user icons and names. The 'ARCHIVOS' section lists months from Enero to Abril. The footer includes social media links and copyright information.

**UNIVERSES**

UniversES  
Todo el conocimiento del universo a tu alcance

**INICIO**

**Dolor sit amet**

**COMENTARIOS**

**ARCHIVOS**

**Sed do eiusmod**

**Resolucion**

(c) Copyright 2020 Aviso Legal Política de Privacidad Contactar

## Resolución

En la práctica anterior hemos visto cómo crear la estructura básica de un par de páginas HTML5, incluyendo todas las secciones importantes. Sin embargo, cuando ejecutábamos esos documentos, lo que se apreciaba era una página en blanco con mucho texto, enlaces y sin formato alguno. Ahora es el momento de cambiar eso y conseguir que obtengan una imagen adecuada.

Antes de empezar a definir las reglas de estilo, se debe tener en cuenta que el código de esta práctica es todo CSS y que debe estar contenido en el archivo STYLES.CSS que declaramos en la sección de cabecera de definición del

documento de la práctica 2.

Dicho esto, lo primero que haremos es proporcionar unas reglas básicas que establezcan las fuentes de texto a utilizar y las bases para algunos elementos.

Si nos fijamos en el código de la práctica, veremos que, el elemento IMG, tiene declarado en el pseudo-elemento AFTER, un posicionamiento absoluto. Recordemos que cuando se establece un posicionamiento absoluto, el elemento se posiciona con respecto al primer ancestro que tenga un posicionamiento relativo o estático, por lo que el punto (0, 0), asociado al pseudo-elemento AFTER será el punto (0,0) del elemento IMG.

Aclarado esto, ahora, lo que haremos es definir la parte superior de la página, es decir, la zona de cabecera que contiene el logo y el menú de navegación.

Si estamos en una resolución de escritorio, la cabecera tendrá color de fondo transparente con color de texto blanco. Además, la dejaremos fija para que, aunque se avance hacia abajo con la barra de desplazamiento, siempre tengamos mano el menú principal.

Si estamos en resolución de tableta o móvil, es decir, una resolución menor o igual a 768 píxeles, se añadirá una clase denominada FIXED que hará que aparezca el botón de hamburguesa, que inicialmente estaba oculto, y se oculten las opciones de menú. Cuando se pulse este botón, se producirá una animación como si se abriese una capa (que en realidad es el elemento UL con las opciones de menú) y, el ícono del botón se transforme en una cruz emulando un ícono de cerrar.

Todavía no vemos grandes cambios, no obstante, eso va a cambiar en unos momentos, en cuanto establezcamos las reglas de estilo del banner.

Este banner, se caracterizará por tener un degradado de fondo de gris oscuro a negro que cubrirá el ancho de la pantalla y una con una altura de 250 píxeles. Sé que esto puede parecer un poco tosco, pero más adelante, esto cambiará cuando obtenga un carácter más personal e impactante al insertarle una imagen de fondo.

En esta casuística también nos encontraremos con una situación particular. Al tener el banner un posicionamiento relativo y su antecesor un posicionamiento fijo, se situará justo en la parte superior del documento o página, lo que hará que el logo y las opciones de menú se vuelvan visibles debido al contraste entre el color de fondo del banner (negro) y el color del texto del menú (blanco).

El cuerpo del documento constará de un elemento principal de información MAIN que ocupará el cien por cien del ancho de la pantalla menos 250 píxeles. Este espacio de 250 píxeles será utilizado por un elemento ASIDE, el cual contendrá los comentarios de los usuarios y acceso a los artículos o posts del sitio agrupados por mes.

Para establecer este ancho tan específico en el elemento MAIN, veremos que se recurre a la función CALC de CSS, la cual nos permitirá calcular el espacio disponible en base al tamaño fijo del elemento ASIDE. No obstante, esta función de CSS no se utilizará sólo aquí, también se utilizará en otras reglas, incluyendo la regla anterior en la parte de la cabecera BODY > HEADER NAV.

A cada artículo o elemento ARTICLE de la sección MAIN, se le establecerá un ancho del cincuenta por ciento menos veinte píxeles, que son los píxeles que ocupa el margen establecido a cada artículo. A la imagen de cabecera de cada uno de ellos, se le establecerá un ancho del cien por cien del artículo y una altura de 160 píxeles, en disposición de bloque.

A los textos introductorios de cada artículo, se le establecerá una altura mínima para asegurar la alineación correcta entre artículos y, en los enlaces de acceso al contenido completo del artículo, se establecerá un ancho idéntico al

ancho del texto, con un aspecto similar a un botón y un efecto animado (un intento de hacer un cometa en movimiento) que se mostrará cuando el usuario se sitúe encima de él.

Para proporcionar un comportamiento responsive o receptivo a esta parte, cuando entremos en una resolución de 768 píxeles, el elemento MAIN, que es quien tiene los artículos, se maximizará hasta el cien por cien, dejando debajo de él el elemento ASIDE. Aquí seguiremos teniendo dos artículos por fila, pero cuando entremos en una resolución de 480 píxeles, esto cambiará un artículo por fila para que se vuelva más legible.

La capa lateral ASIDE se colocará a la derecha a través de un posicionamiento flotante con un ancho fijo del 25% menos el margen que son 10 píxeles. Para darle un comportamiento responsive o receptivo, cuando entremos en una resolución de 768 píxeles o menor, lo que pasará es que los bloques de comentarios y archivos ocuparán el cien por cien del ancho de la pantalla y sus subelementos se redistribuirán en línea según vayan entrando en el espacio disponible.

Pues nada, sólo nos resta darle estilo a pie de página para terminar esta práctica. En esta parte de la página, lo que haremos es situar el logo a la izquierda, los enlaces de las redes sociales a la derecha y, tras una línea inferior separadora, el resto de información en una distribución en línea.

Y con esto, ya hemos terminado de realizar todas las reglas necesarias para poder ver la página como se muestra a continuación.

# 4

## IMÁGENES Y MULTIMEDIA

Una imagen se podría definir como una representación gráfica que muestra cómo es una entidad, figura o cosa. El problema de las imágenes viene, como casi siempre, cuando se deben presentar en resoluciones diferentes y hay que buscar soluciones adaptativas que no provoquen pérdida de información ni relación de aspecto.

En este capítulo vamos a ver cómo definir contenidos multimedia de forma adecuada, cómo hacer las imágenes adaptativas y receptivas y cómo hacer que cualquier contenido multimedia sea usable y accesible.

### TIPOS DE IMÁGENES

Existen múltiples tipos de imágenes, pero los más extendidos son las imágenes de mapas de bits o rasterizadas y las imágenes vectoriales.

Las **imágenes rasterizadas** están formadas por píxeles y son, básicamente, la mayoría de las imágenes que se pueden encontrar en Internet. En general, ocupan más espacio en disco que las vectoriales y, dependiendo del formato de compresión que se utilice, pueden variar su tamaño, sin embargo, su gran inconveniente es que, cuando se amplían o reducen, se produce una pérdida de calidad debido, fundamentalmente a los procesos extrapolación o interpolación que intervienen en dichas ampliaciones y/o reducciones.

Mientras que la interpolación puede provocar un efecto de imagen borrosa y algo desenfocada, la extrapolación puede, y de hecho lo hace la mayoría de las veces, provocar un efecto de pixelado que, básicamente, se produce por mostrarlas a un tamaño en el que los píxeles individuales se vuelven visibles a simple vista.

Entre los diferentes formatos de compresión que existen o se aplican para las imágenes rasterizadas se encuentran el formato WEBP, PNG, JPEG o GIF y entre los diferentes recursos que se pueden utilizar para manipularlas se encuentran Adobe Photoshop y GIMP.

Las **imágenes vectoriales** no están formadas por píxeles, sino que están compuestas por unas líneas (rectas o curvas) que se generan entre dos puntos de control y que se calculan a partir de una fórmula matemática. Esta es la principal razón por la que las imágenes vectoriales suelen ocupar mucho menos espacio, en comparación con las rasterizadas, y de por qué al ampliarlas o reducirlas no se produce pérdida de calidad ni efecto de pixelado.

Entre los diferentes formatos de compresión que existen o se aplican para las imágenes vectoriales se encuentran el formato SVG, EPS o AI y, entre los diferentes recursos que se pueden utilizar para manipularlas se encuentran Adobe Illustrator y CorelDraw.

## ELEMENTOS DISPONIBLES EN HTML5

HTML5 es un lenguaje versátil que permite manejar audio, video e imágenes adaptativas de múltiples formas, no obstante, los elementos más recurrentes siguen siendo AUDIO, VIDEO e IMG.

### Elemento audio

El elemento AUDIO especifica que el contenido que se va a representar es un sonido o música. Aunque actualmente existen varios formatos de sonido, entre los que podemos encontrar el MP3, WAV o OGG, el único compatible con todos los navegadores es el MP3.

```
<audio controls>
<source src="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=e672fcf48aecca494667969381aobdd4"
type="audio/ogg">
<source src="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=1346bb36942b4d980fe6711fb04f77de"
type="audio/mpeg">
<source src="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=842b32ddb7c6a349f788c6b1ec5e4ocd"
type="audio/wav">
<track src="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=bceb43629d14a77be1b185d5da92d296"
kind="descriptions"
srclang="en"
label="Inglés" />
<track src="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=a992a0d767ac4d82d2633da43478d5e3"
kind="descriptions"
srclang="es"
label="Español" />
El navegador no soporta la etiqueta audio.
</audio>
```

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>autoplay</b>	Especifica que el audio debe reproducirse tan pronto como esté listo, es decir, en cuanto esté cargado e integrado con el DOM.
<b>controls</b>	Especifica que se deben mostrar los controles de interacción, es decir, los botones de parar, reanudar, siguiente, etcétera.
<b>loop</b>	Especifica que el audio se debe repetir de manera continua cuando termine su reproducción.
<b>muted</b>	Especifica que el volumen del medio de reproducción debe estar mudo, es decir, que la salida de audio debe estar silenciada.
<b>preload</b>	Especifica cómo se debe precargar el archivo de audio. Si su valor se establece a AUTO, se realizará de forma automática. Si su valor es METADATA, se precargarán sólo los metadatos y, si se establece a NONE, no se hará nada.
<b>src</b>	Especifica la URL del archivo de audio a cargar.

Si nos fijamos en el código de ejemplo anterior, veremos que se han establecido varios elementos SOURCE. Esto se suele hacer así porque, cuando se vaya a renderizar el elemento AUDIO, el navegador seleccionará entre todos los elementos SOURCE disponibles, el que más se ajuste a las necesidades o, habitualmente, el que soporte. La forma de seleccionar la fuente de audio será a través del tipo MIME descrito en el atributo TYPE.

Si el elemento AUDIO no puede reproducir ninguno de los audios propuestos o el agente de usuario indica que no es posible la reproducción de sonido, el elemento TRACK podrá adquirir un papel importante. Esto es así porque, el elemento TRACK especifica una pista adicional que servirá como descripción textual para el elemento AUDIO.

## Elementos figure y figcaption

El elemento FIGURE especifica que el contenido que se va a representar es una ilustración, diagrama, fotografía, listado de códigos o algo similar. Para describir el contenido del FIGURE puede ser descrito a través del elemento FIGCAPTION.

```
<figure>

<figcaption>Ejemplo de diagrama de Gantt</figcaption>
</figure>
```

Cabe destacar que el elemento FIGURE es un elemento de sección que está excluido del esquema principal del documento por considerarse que su propósito es introducir contenidos externos y que no tiene por qué estar formado por un único elemento de contenido. De hecho, es frecuente verlo para definir un conjunto de elementos multimedia que representan una única entidad que se desea, se interprete, como una única figura.

## Elemento img

El elemento IMG especifica que el contenido que se va a representar es una imagen. Este elemento imagen no es incrustado en el documento, aunque sí que se reserva un espacio de retención para la imagen.

```

```

El elemento IMG no se debería utilizar si el contenido que muestra no está relacionado directamente con el contenido del documento, es decir, sólo se debe utilizar cuando su representación sea significativa para el contenido del documento.

También es importante saber que la inserción de imágenes en un documento puede afectar al rendimiento global y a la accesibilidad, por lo que se deben definir de forma precisa utilizando todos los atributos necesarios.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>alt</b>	Especifica el texto descriptivo que se debe mostrar cuando la imagen no esté disponible, sea cual sea la razón. Este valor es importante para la usabilidad web y la accesibilidad web.

<b>decoding</b>	Es un valor que ayuda a los agentes de usuario cuál es el método por el que se debe decodificar la imagen. Si su valor es SYNC, la imagen deberá ser decodificada de forma síncrona, si es ASYNC, de manera asíncrona y, si es AUTO, la decisión será tomada por el agente de usuario.
<b>height</b>	Especifica la altura en píxeles de la imagen dentro del documento.
<b>src</b>	Especifica la URL del archivo a cargar.
<b>srcset</b>	Es una lista de valores separados por comas que indican las posibles fuentes disponibles. Cada valor de esta lista se compone de un SRC, el ancho del recurso seguido del carácter "w" (por ejemplo, 360w o 480w) y la densidad en píxeles del recurso seguido del carácter "x".
<b>sizes</b>	Especifica una lista de elementos, separados por comas, con las preferencias de tamaño en función una condición de medios. Cada elemento de esta lista se compone de una condición de medios y una preferencia de tamaño en píxeles para esa condición. Si el último valor de la lista no tiene especificada una condición de medios, se tomará como valor por defecto.
<b>usemap</b>	Especifica que la imagen u objeto es un mapa que tiene asociadas áreas en las que se puede hacer clic. Cabe destacar que, debe corresponderse con el atributo NAME del elemento MAP y que, si el elemento IMG está dentro de un elemento A, o elemento BUTTON, no funcionará.
<b>width</b>	Especifica la anchura en píxeles de la imagen dentro del documento.

## Elementos map y area

El elemento MAP se suele utilizar para definir mapas de imagen a través del elemento AREA. Este elemento AREA posee la capacidad de ser vinculable y, por tanto, dispone de la capacidad de asociar una acción a una zona concreta de dicho elemento.

```
<map name="socialmedia">
<area href="https://twitter.com/home"
title="Pulsa aquí para ir a Twitter"
shape="rect"
coords="0, 0 48, 48"
target="_blank" alt="twitter" />
</map>

```

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>alt</b>	Especifica el texto descriptivo que está asociado al atributo HREF del elemento AREA. Este valor es importante para la usabilidad web y la accesibilidad web.
<b>coords</b>	Especifica las coordenadas de la forma. Si el atributo SHAPE está establecido a RECT, COORDS contendrá las coordenadas de la esquina superior izquierda y de la esquina inferior derecha. Si el atributo SHAPE está establecido a CIRCLE, COORDS contendrá las coordenadas la coordenada (X, Y) de posicionamiento y el radio del círculo. Si el atributo SHAPE está establecido a POLY, COORDS contendrá una lista de coordenadas que se corresponderán con los bordes del polígono. Si el atributo SHAPE está establecido a DEFAULT, COORDS será ignorado ya que se tomará la región entera como área de interacción.

<b>download</b>	Especifica el nombre del archivo que se asociará al documento vinculado. Sólo se debe especificar cuando el atributo HREF esté presente.
<b>href</b>	Especifica el destino hacia dónde se irá cuando se pulse en el enlace. Si este valor empieza por el símbolo almohadilla, indicará que se desea ir a otra sección del documento actual. De no ser así, indicará la dirección hacia otro documento diferente.
<b>hreflang</b>	Especifica el idioma del documento vinculado.
<b>media</b>	Especifica el medio para el que está optimizado o pensado el objeto destino. Normalmente, se suele utilizar para indicar que el documento vinculado es para un dispositivo especial, de impresión, o de manipulación por voz. La forma de realizar esta condición es similar a las media-queries de CSS, se establece una propiedad MIN-WIDTH o MAX-WIDTH y un valor en píxeles.
<b>rel</b>	Especifica la relación que existe entre el documento actual y el documento vinculado. Sus posibles valores son ALTERNATE, AUTHOR, BOOKMARK, HELP, LICENSE, NEXT, NOFOLLOW, NOREFERRER, PREFETCH, PREV, SEARCH y TAG.
<b>shape</b>	Especifica la forma del área. Los posibles valores de RECT, CIRCLE, POLY y DEFAULT, que definen un rectángulo, un círculo, un polígono o una región entera, respectivamente.
<b>target</b>	Especifica dónde a qué pestaña o ventana se enviará la información. Entre los posibles valores que puede tomar, los más frecuentes son _BLANK, para indicar que se abra en una nueva pestaña, _SELF, para indicar que se abra en la misma pestaña y _TOP, para que se abra en el primer elemento BODY de la ventana.
<b>type</b>	Especifica el tipo MIME del objeto o imagen referenciado por el elemento AREA.

Cabe destacar que, el elemento MAP necesita de un atributo denominado NAME que hace posible que sea referenciado por una imagen. Dicha imagen se asocia a través del atributo USEMAP.

## Elemento picture

El elemento PICTURE fue diseñado con la idea de proporcionar soporte nativo a imágenes responsive o adaptativas. En general, se utiliza de forma conjunta con el elemento SOURCE y IMG para ofrecer las diferentes alternativas de la imagen en distintos escenarios o resoluciones.

```
<picture>
<source srcset="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=a9a24fee6cf05c811695fdf20dae2850" media="(min-
width: 1680px)" />
<source srcset="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=c4923e2b20aff6fdd11543caeae667e1" media="(min-
width: 1366px)" />
<source srcset="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=c5d109f7d940d610cece4c846eaeb5eb" media="(min-
width: 640px)" />
<source srcset="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=815c3c68d24582ea3fca2403137e8e27" media="(min-
width: 360px)" />

</picture>
```

Cuando se definen todos los elementos, el agente de usuario seleccionará, entre todos los elementos secundarios SOURCE, el que mejor coincida con el escenario actual. Si no encuentra una coincidencia que se

ajuste lo suficientemente, o no soporta el elemento PICTURE, lo que se representará será la imagen asociada al elemento IMG.

Cabe destacar que, el elemento PICTURE no dispone de atributos particulares y que, descartando Internet Explorer 11 y Microsoft Edge versión 12, prácticamente todos los navegadores proporcionan soporte nativo.

## Elemento source

El elemento SOURCE permite especificar los recursos alternativos de medios que están disponibles para ser gestionados por los elementos AUDIO, PICTURE y VIDEO.

Estos recursos serán seleccionados de forma automática por el agente de usuario en función del tipo de medio, códec o consulta de medios.

```
<picture>
<source srcset="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=c4923e2b20aff6fddf11543caeae667e1" media="(min-
width: 1366px)" />
<source srcset="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=c5d109f7d940d610cece4c846eaeb5eb" media="(min-
width: 900px)" />
<source srcset="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=815c3c68d24582ea3fc2403137e8e27" media="(min-
width: 768px)" />
</picture>
```

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>src</b>	Especifica la dirección o ubicación del recurso multimedia a cargar. Es un elemento obligatorio cuando está definido dentro de una estructura AUDIO o VIDEO.
<b>srcset</b>	Especifica una lista de imágenes, separadas por coma, a seleccionar según sea el medio, resolución, ... Cada elemento de esta lista se compone de una URL, un descriptor de ancho seguido de la letra W minúscula (por ejemplo, 360w o 480w) y un descriptor de densidad seguido de la letra X minúscula (por ejemplo, 2x). Aunque las opciones de descriptor de ancho y descriptor de densidad son opcionales, al menos, una de ellas siempre debe estar presente. Es un elemento obligatorio cuando está definido dentro de una estructura PICTURE.
<b>media</b>	Especifica la consulta de medios que se debería cumplir para poder ser aplicado el recurso. Sigue las mismas normas y validaciones que las consultas de medios definidas por la regla @MEDIA.
<b>sizes</b>	Especifica una lista de elementos, separados por comas, con las preferencias de tamaño en función una condición de medios. Cada elemento de esta lista se compone de una condición de medios y una preferencia de tamaño en píxeles para esa condición. Este atributo es ignorado cuando el elemento SOURCE es descendiente directo del elemento PICTURE. En su lugar, se debe utilizar el atributo MEDIA. Si el último valor de la lista no tiene especificada una condición de medios, se tomará como valor por defecto.
<b>type</b>	Especifica el tipo MIME del recurso. Todos los posibles valores que puede tomar este atributo están disponibles en la dirección <a href="http://www.iana.org/assignments/media-types/">http://www.iana.org/assignments/media-types/</a> .

## Elemento track

El elemento TRACK especifica una pista de texto que transcribe lo que se está emitiendo por sonido.

```
<track src="https://api.perlego.com/assets/asset?  
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=f2f3af5c7dcc47ecf2e47255e58a0b44" kind="subtitles"  
srclang="en" label="Inglés" />
```

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>default</b>	Especifica que la pista se asignará por defecto, siempre que las preferencias del usuario no digan lo contrario.
<b>kind</b>	Especifica el objetivo de la pista de texto. Entre los posibles valores se puede seleccionar LAPTOP, CAPTIONS, CHAPTERS, DESCRIPTIONS, METADATA y SUBTITLES.
<b>label</b>	Especifica el título a asignar a la pista de texto.
<b>src</b>	Especifica la URL del archivo a cargar.
<b>srclang</b>	Especifica el idioma de la pista de texto. El valor de este campo debe ser representado bajo el estándar ISO-639-1, el cual identifica el lenguaje a partir de dos caracteres.

## Elemento video

El elemento VIDEO especifica que el contenido que se va a representar es una película, cortometraje o cualquier otro contenido de vídeo. Aunque actualmente existen varios formatos de video, entre los que podemos encontrar el MP4, AVI, WEBM o OGG, el más compatible es el formato en MP4.

```
<video controls>  
<source src="https://api.perlego.com/assets/asset?  
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=8dc7807de704ba7325c90a8bdbece4b4"  
type="audio/ogg">  
<source src=" https://api.perlego.com/assets/asset?  
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=edodd3129ba236acf9735155d62fe10"  
type="audio/mp4">  
<source src=" https://api.perlego.com/assets/asset?  
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=bec15945326b5e0fe11dd008f5527d22"  
type="audio/webm">  
<track src="https://api.perlego.com/assets/asset?  
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=f2f3af5c7dec47ecf2e47255e58a0b44" kind="subtitles"  
srclang="en" label="Inglés" />  
<track src="https://api.perlego.com/assets/asset?  
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=ad94b07db5e8827af292ead0968977c7"  
kind="subtitles" srclang="es" label="Español" />  
El navegador no soporta la etiqueta video.  
</video>
```

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>autoplay</b>	Especifica que el vídeo debe reproducirse tan pronto como esté listo, es decir, en cuanto esté cargado e integrado con el DOM.
<b>controls</b>	Especifica que se deben mostrar los controles de interacción, es decir, los botones parar, reanudar, siguiente, etcétera.
<b>loop</b>	Especifica que el audio se debe repetir de manera continua cuando termine su reproducción.

<b>height</b>	Especifica la altura del objeto para reproducir el video. Habitualmente, se corresponde con un tamaño en proporción con el formato de la película. Por ejemplo, si el video está en formato 4:3, el alto suele ser 288.
<b>muted</b>	Especifica que el volumen del medio de reproducción debe estar mudo, es decir, que la salida de audio debe estar silenciada.
<b>preload</b>	Especifica cómo se debe precargar el archivo de vídeo. Si su valor se establece a AUTO, se realizará de forma automática. Si su valor es METADATA, se precargará sólo los metadatos y, si se establece a NONE, no se hará nada.
<b>src</b>	Especifica la URL del archivo a cargar.
<b>width</b>	Especifica la anchura del objeto para reproducir el video. Habitualmente, se corresponde con un tamaño en proporción con el formato de la película. Por ejemplo, si el video está en formato 4:3, las proporciones podrían ser 320x240.

Si el elemento VIDEO no puede reproducir ninguno de los vídeos propuestos con sonido o el agente de usuario indica que no es posible la reproducción de sonido, el elemento TRACK podrá adquirir un papel importante. Esto es así porque, el elemento TRACK especifica una pista adicional que servirá como descripción textual para el elemento VIDEO.

## PROPIEDADES DISPONIBLES EN CSS

CSS es un lenguaje que posee una gran variedad de propiedades para el manejo y manipulación de imágenes. A continuación, se muestran la mayor parte de ellas, si no todas.

### Propiedad background-attachment

Especifica si la imagen establecida por BACKGROUND-IMAGE debe desplazarse con el resto del documento o debe quedarse fija. Entre sus posibles valores podemos encontrar:

- **SCROLL**: Especifica que la imagen se desplazará con el documento. Es el valor por defecto.
- **FIXED**: Especifica que la imagen debe mantenerse fija, es decir, no se desplazará con la página o documento.
- **LOCAL**: Especifica que la imagen debe desplazarse con el contenido del elemento al que está asociada.

#### Ejemplos:

```
div { background-attachment: fixed; }
div { background-attachment: scroll; }
div { background-attachment: inherit; }
```

### Propiedad background-clip

Especifica cómo debe extenderse el fondo, gradiente o imagen dentro del elemento actual. Entre sus posibles valores podemos encontrar:

- **BORDER-BOX**: Indica que el fondo debe extenderse incluyendo el borde del elemento. Es el valor por defecto.

El fondo se extiende incluyendo el borde y el margen interno.

- **PADDING-BOX:** Indica que el fondo debe extenderse sin incluir el borde del elemento.

El fondo se extiende incluyendo el borde y el margen interno.

- **CONTENT-BOX:** Indica que el fondo debe extenderse hasta dónde empieza el espacio útil para el contenido del elemento sin incluir el margen interno.

El fondo se extiende incluyendo el borde y el margen interno.

### Ejemplo:

```
div { background-clip: border-box; }
```

## Propiedad background-image

Especifica una o varias imágenes o gradientes para un elemento. Entre sus posibles valores podemos encontrar:

### NOTA

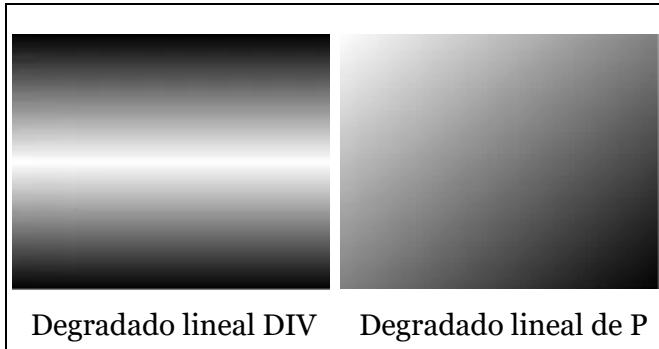
Para más información sobre cómo utilizar cada una de las siguientes funciones, consultar el apartado de “Interfaz de usuario” del capítulo anterior.

- **[URL]:** Indica las imágenes o gradientes que se utilizarán como fondo, todas ellas separadas con coma.
- **NONE:** Indica que no se desea fondo alguno. Es el valor por defecto.
- **LINEAR-GRADIENT:** Indica que se desea un degradado lineal como imagen de fondo a través de, como mínimo, dos colores separados por coma.
- **RADIAL-GRADIENT:** Indica que se desea un degradado radial, del centro hacia afuera, como imagen de fondo a través de, como mínimo, dos colores separados por coma.
- **REPEATING-LINEAR-GRADIENT:** Indica que se desea un degradado lineal repetitivo.
- **REPEATING-RADIAL-GRADIENT:** Indica que se desea un degradado radial repetitivo.

### Ejemplos de degradado lineal:

```
div { background-image: linear-gradient(black, white, black); }  
p { background-image: linear-gradient(to bottom right, white, black); }
```

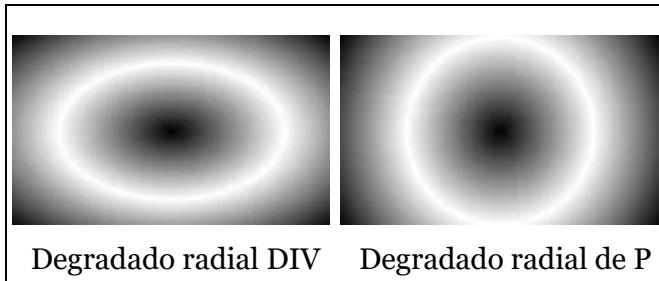
El resultado debería ser algo como:



### Ejemplos de degradado radial:

```
div { background-image: radial-gradient(black, white, black); }
p { background-image: radial-gradient(circle, black, white, black); }
```

El resultado debería ser algo como como:



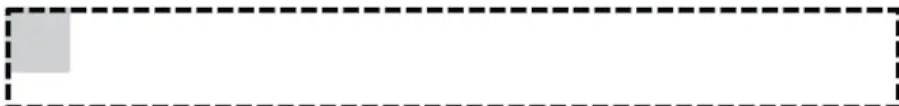
### Ejemplos con URL:

```
div { background-image: url("https://pixabay.com/es/photos/vía-láctea-
cielo-estrellado-2695569/"); }
```

### Propiedad background-origin

La propiedad BACKGROUND-ORIGIN funciona de forma similar a la propiedad BACKGROUND-CLIP y especifica la posición de origen de una imagen de fondo. Entre sus posibles valores podemos encontrar:

- **BORDER-BOX:** Indica que la imagen empezará en la esquina superior izquierda del borde.



- **PADDING-BOX:** Indica que la imagen empezará en la esquina superior izquierda del límite del margen interno. Es el valor por defecto.



- **CONTENT-BOX:** Indica que la imagen empezará en la esquina superior izquierda del límite del contenido.



## Ejemplos:

```
div { background-clip: border-box; }
span { background-clip: padding-box; }
i { background-clip: content-box; }
```

## Propiedad background-position

Especifica la posición inicial del fondo. Entre sus posibles valores podemos encontrar:

- **[POSICIÓN]**: Indica un valor de posicionamiento. Puede ser una combinación de palabras clave como son LEFT TOP, LEFT CENTER, LEFT BOTTOM, CENTER TOP, CENTER CENTER, CENTER BOTTOM, RIGHT TOP, RIGHT CENTER, RIGHT BOTTOM, o un valor establecido en una de las medidas permitidas de CSS.

A continuación, se muestra una ilustración con cada uno de los significados:

LEFT TOP	CENTER TOP	RIGHT TOP
LEFT CENTER	CENTER CENTER	RIGHT CENTER
LEFT BOTTOM	CENTER BOTTOM	RIGHT BOTTOM

## Ejemplos:

```
div { background-position: right top; }
span { background-position: 50% 50%; }
i { background-position: 0px 150px; }
```

## Propiedad background-repeat

Especifica si la imagen establecida como fondo debe repetirse y cómo debe hacerlo. Entre sus posibles valores podemos encontrar:

- **REPEAT**: Indica que la imagen debe repetirse tanto horizontal, como verticalmente. Es el valor por defecto.
- **REPEAT-X**: Indica que la imagen debe repetirse sólo horizontalmente.
- **REPEAT-Y**: Indica que la imagen debe repetirse sólo verticalmente.
- **NO-REPEAT**: Indica que la imagen NO debe repetirse. Sólo se mostrará una única vez.
- **SPACE**: Indica que la imagen debe repetirse tanto como sea posible, siempre y cuando, no se deformé ni se corte. Esto podrá causar que se generen espacios entre repetición y repetición, precisamente, porque no puede ser deformada ni recortada.
- **ROUND**: Indica que la imagen debe repetirse para llenar el espacio del elemento, aunque eso implique que sea deformada.

## Ejemplos:

```
div { background-repeat: no-repeat; }  
p { background-repeat: space; }
```

## Propiedad background-size

Especifica el tamaño del fondo. Entre sus posibles valores podemos encontrar:

- **AUTO**: Indica que el tamaño de la imagen debe ser igual al tamaño original. Es el valor por defecto.
- **COVER**: Indica que el tamaño de la imagen debe ajustarse para cubrir todo el contenedor o elemento, aunque eso implique que la imagen se corte por los extremos.
- **CONTAIN**: Indica que el tamaño de la imagen debe ajustarse para asegurarse de que sea totalmente visible. Cuando este valor se utiliza, lo normal es que se generen espacios en blanco en alguno de los extremos del elemento.
- **[VALOR]**: Indica un valor establecido en una de las medidas permitidas de CSS.

## Ejemplo:

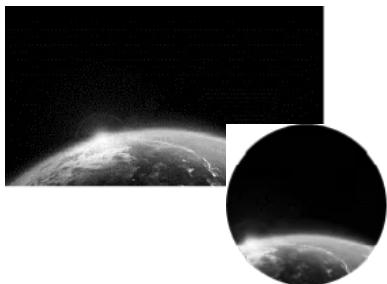
```
div { background-size: cover; }
```

## Propiedad clip-path

Especifica una forma de recorte básica sobre el elemento. Entre sus posibles valores podemos encontrar:

- **CIRCLE**: Indica que la forma de recorte debe ser circular. Para especificar el valor del radio o las coordenadas del centro se puede usar las palabras clave (CLOSEST-SIDE o FARTHEST-SIDE o una longitud en cualquiera de las unidades de medida estándar de CSS. Su sintaxis es:

```
clip-path: circle();  
clip-path: circle(radius at posX posY);
```

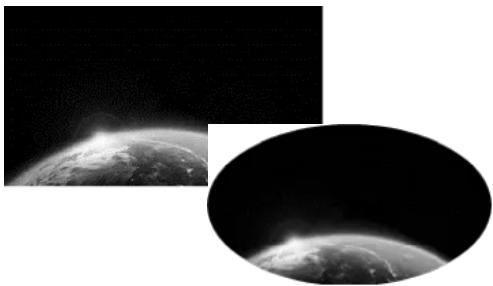


Original clip-path: circle();

- **ELLIPSE**: Indica que la forma de recorte debe ser elíptica. Para especificar el valor del radio o las coordenadas del centro se puede usar las palabras clave (CLOSEST-SIDE o FARTHEST-SIDE o una longitud en cualquiera de las unidades de medida de longitud estándar de CSS. Su sintaxis es:

```
clip-path: ellipse();
```

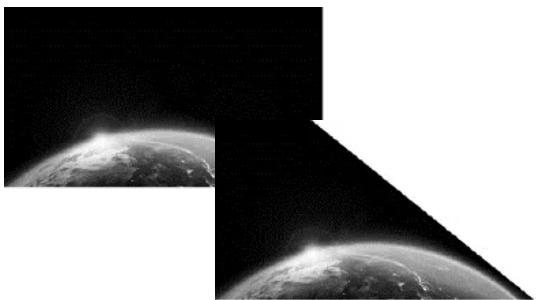
clip-path: ellipse(radio en X Y at posX posY);



Original clip-path: ellipse();

- **POLYGON:** Indica que la forma de recorte debe ser poligonal. Para especificar el valor de coordenada (X, Y) de cada punto de corte se puede utilizar cualquiera de las unidades de medida de longitud estándar de CSS, separados por el carácter coma. Su sintaxis es:

clip-path: polygon(X Y, X Y, ..., X Y);

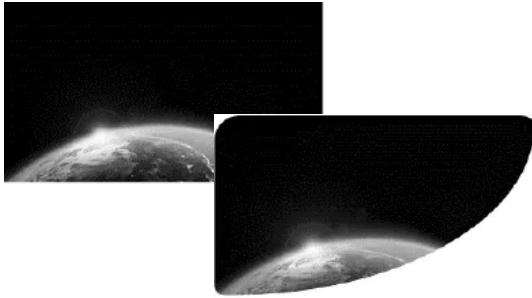


Original polygon(0 0,30% 0,100% 100%,0 100%)

- **INSET:** Indica que la forma de recorte debe ser rectangular. Si se especifica la palabra clave ROUND se pueden establecer valores de redondeo para cada esquina como si de la propiedad BORDER-RADIUS se tratase. Para especificar el valor de coordenada (X, Y) de cada punto de corte se puede utilizar cualquiera de las unidades de medida de longitud estándar de CSS, separados por el carácter coma. Su sintaxis es:

clip-path: inset(top right bottom left);

clip-path: inset(top right bottom left round r1 r2);



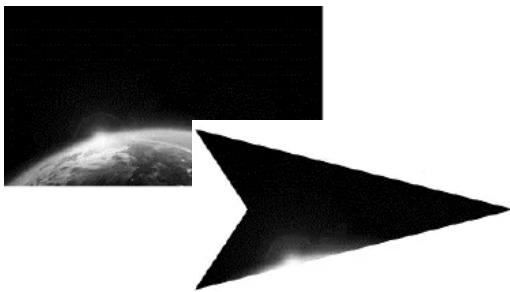
Original inset(0 0 0 0 round 25px 5% 100% 10px)

- **URL:** Indica que la forma de recorte debe seguir a partir de un patrón proporcionado mediante un SVG. Su sintaxis es:

```
clip-path: url([ID_SVG]);
```

Supongamos el siguiente SVG y veamos cómo afectaría al aplicarlo:

```
<svg width="0" height="0">
<defs>
<clipPath id="shape">
<path d="M 0 0 L 300 75 L 0 150 L 50 75 L 0 0 " ></path>
</clipPath>
</defs>
</svg>
```



Original url(#shape);

#### Otros ejemplos:

```
div { clip-path: url('..../images/shape1.png'); }
div { clip-path: url('data:image/png;base64,iVBORw0K...'); }
```

## I Propiedad object-fit

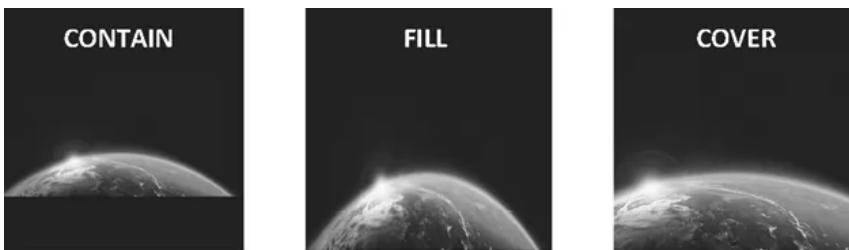
Especifica cómo se debe ajustar el elemento con respecto a su elemento padre o contenedor. Entre sus posibles valores podemos encontrar:

- **FILL:** Indica que el elemento será ajustado al tamaño del contenedor, aunque este deba ser deformado, si así se requiere. Es el valor por defecto.

- **CONTAIN**: Indica que el elemento será ajustado con respecto al tamaño del contenedor, pero guardando la relación de aspecto para que entre todo su contenido en el espacio disponible.
- **COVER**: Indica que el elemento será ajustado con respecto al tamaño del contenedor, pero guardando la relación de aspecto para llenar el espacio disponible. Este valor puede hacer que se corte información por los extremos.
- **NONE**: Indica que el elemento NO será ajustado ni deformado.
- **SCALE-DOWN**: Indica que el elemento será ajustado con respecto al tamaño del contenedor y tomando como referencia el menor valor entre el alto y ancho para entre todo su contenido en el espacio disponible.

#### NOTA

Esta propiedad sólo es aplicable a los elementos IMG y VIDEO.



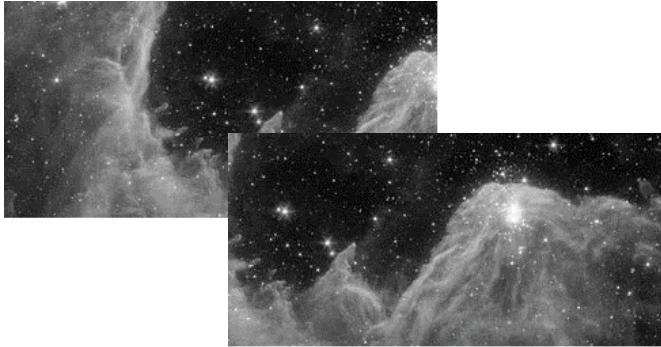
#### Ejemplo:

```
img { width: 320px; height: 240px; object-fit: cover; }
video { width: 480px; height: 360px; object-fit: contain; }
```

### Propiedad object-position

Especifica dónde se debe colocar el elemento con respecto a su elemento padre o contenedor. Entre sus posibles valores podemos encontrar:

- **FILL**: Indica que el elemento será ajustado al tamaño del contenedor, aunque este deba ser deformado, si así se requiere. Es el valor por defecto
- **CONTAIN**: Indica que el elemento será ajustado con respecto al tamaño del contenedor, pero guardando la relación de aspecto para que entre todo su contenido en el espacio disponible.
- **COVER**: Indica que el elemento será ajustado con respecto al tamaño del contenedor, pero guardando la relación de aspecto para llenar el espacio disponible. Este valor puede hacer que se corte información por los extremos.
- **NONE**: Indica que el elemento NO será ajustado ni deformado.
- **[POSICIÓN]**: Indica un valor de posicionamiento. Puede ser una combinación de palabras clave como son LEFT TOP, LEFT CENTER, LEFT BOTTOM, CENTER TOP, CENTER CENTER, CENTER BOTTOM, RIGHT TOP, RIGHT CENTER, RIGHT BOTTOM, o un valor establecido en una de las medidas permitidas de CSS.



object-position: center center object-position: left center

A continuación, se muestra una ilustración con cada uno de los significados:

LEFT TOP	CENTER TOP	RIGHT TOP
LEFT CENTER	CENTER CENTER	RIGHT CENTER
LEFT BOTTOM	CENTER BOTTOM	RIGHT BOTTOM

#### NOTA

Esta propiedad sólo es aplicable a los elementos IMG y VIDEO.

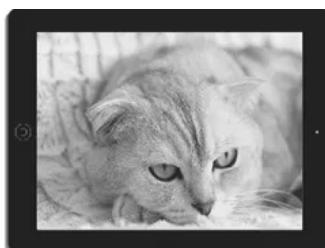
#### Ejemplos:

```
img { object-position: left top; }  
video { object-position: 50% 2vh; }
```

## IMÁGENES RECEPTIVAS Y ADAPTATIVAS

Cada vez más, accedemos a los contenidos web desde muy diferentes dispositivos con distintos tamaños y resoluciones. Esto provoca que los diseñadores y desarrolladores tengan que ingeníárselas para mostrar los contenidos de forma que no pierdan información, calidad o relación de aspecto.

Hasta no hace tanto, era habitual ver las imágenes deformadas o con espacios en “blanco” alrededor, lo que provocaba sensación de mala calidad, mal gusto o una imagen corporativa descuidada.



Pero entonces, apareció el concepto de diseño receptivo o adaptativo y, con él, varias técnicas de adaptación de contenidos que trataban de conseguir que las imágenes se viesen de forma adecuada.



Aunque, a primera vista, no es la misma la información dependiendo del dispositivo en el que se muestra la imagen, toda la información está disponible. Esto es posible gracias a la implementación de funcionalidades adicionales que permiten, entre otras cosas, agrandar o empequeñecer la imagen o captar cualquier punto de la misma a través de un desplazamiento.

Dicho esto, las imágenes receptivas o adaptativas pueden conseguirse, fundamentalmente, a través de varios métodos o técnicas, que suelen implementarse de forma combinada.

## Adaptación de imágenes mediante propiedades CSS

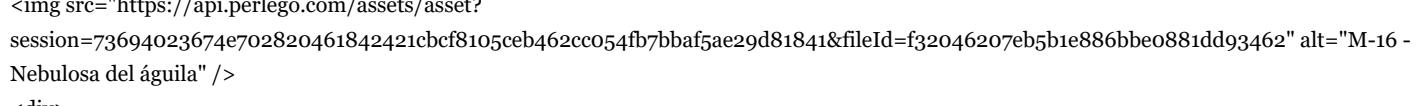
Si lo que se desea es hacer que un elemento IMG se muestre de forma adaptable, la manera más sencilla de conseguir esto es utilizar las propiedades WIDTH, HEIGHT y OBJECT-FIT de forma combinada.

En general, podríamos decir que todas las imágenes deben estar establecidas al cien por cien del ancho del contenedor y con asignación de altura automática. Esto provocará que las imágenes se muestren correctamente, independientemente de su relación de aspecto y tamaño. Sin embargo, si el contenedor no se corresponde, en proporción, con el tamaño de la imagen, lo más probable es que la imagen se corte o se salga fuera de los límites marcados por el contenedor.



Supongamos una situación en la que se desea presentar la imagen de la derecha en un contenedor de 1920x250 píxeles únicamente utilizando un elemento IMG y CSS. Una posibilidad podría ser escribir este código:

```

<style>
div { display: block; height: 250px; position: relative; width: 100%; }
img { height: auto; object-fit: none; width: 100%; }
</style>
<div>

<div>

```

El resultado de este código sería algo como:



Si ahora, en el elemento IMG, en vez de establecer la propiedad HEIGHT a AUTO, la establecemos al 100%, el resultado sería:



Como se puede observar, la imagen se ha deformado para ajustarse a las proporciones del contenedor dónde se encuentra incrustado. Sin embargo, si ahora, en el elemento IMG, establecemos la propiedad OBJECT-FIT a CONTAIN, el resultado será:



Si con la propiedad OBJECT-FIT establecida a CONTAIN, aumentamos el alto del contenedor, la imagen se irá agrandando a la nueva altura. No obstante, si la altura fuese mayor que la requerida por la imagen, lo que se

producirá es un espacio en blanco entre la imagen y el siguiente elemento.

Ahora, si en el elemento IMG, establecemos la propiedad OBJECT-FIT a COVER, el resultado será muy diferente:



Si observamos la imagen anterior con detenimiento, lo que veremos es que la imagen se ha ajustado al 100% del ancho disponible y se ha centrado con respecto al alto del contenedor mostrándonos sólo 250 píxeles desde el punto central de la imagen.

Al final, todo esto se podría reducir en que los contenedores deben estar ajustados, en proporción, a las imágenes que se van a mostrar y bajo la orientación del dispositivo donde se vayan a reproducir.

## Adaptación de imágenes mediante consultas de medios

Si lo que se desea es hacer que se cargue una u otra imagen a través de CSS y en función de la resolución, la manera más sencilla de conseguir esto es utilizar las variaciones de la propiedad BACKGROUND, en combinación con la regla @MEDIA.

Supongamos una situación en la que tenemos cuatro versiones de una misma imagen y, lo que se desea hacer es presentar la imagen como fondo de un elemento DIV, pero con la condición de que se cargue una u otra versión en función de la resolución.

Una posibilidad podría ser el siguiente código:

```
<style>
.banner {
background-image: url(https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=ad483702507f5f5e5bf8e9287dfa3f3f);
border-bottom: 1px solid rgba(0,0,0,0.1);
color: #fff;
display: block;
height: 100vh;
position: relative;
text-align: center;
width: 100%;
}

@media (min-width: 800px) {
.banner { background-image: url(https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=15cb1c5b1d6410df2477a608a6b418d3); }
}

@media (min-width: 1400px) {
.banner { background-image: url(https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=40e869d6ec0988b88dd639db472b251e); }
}

@media (min-width: 2000px) {
```

```
.banner { background-image: url(https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=8ce914b8731af11d1a24e16da09609b8); }
}
</style>
<div class="banner"></div>
```

Basándonos en disciplina de Mobile First, lo que se conseguirá con esta solución es definir la imagen de menor resolución que se desea cargar y, según se vaya detectando que el dispositivo admite una mayor resolución, se irá sobrescribiendo la propiedad BACKGROUND-IMAGE para seleccionar la imagen que más se ajusta al escenario actual.

## Selección de imágenes mediante IMG y SRCSET

El elemento IMG, como hemos visto anteriormente, dispone de un atributo denominado SRCSET que permite seleccionar el origen a presentar, mediante una sintaxis similar a las consultas de medios.

Supongamos, como antes, una imagen de la que disponíamos hasta cuatro versiones. Lo primero que nos podría venir a la cabeza es definir un elemento IMG que fuese actualizando el atributo SRC a través de un lenguaje como JavaScript. Como solución podría estar bien, no obstante, definir un elemento IMG con varios atributos SRCSET es una solución mucho más simple y de menor coste en rendimiento.

```

```

En el código anterior seleccionamos diferentes imágenes en función del ancho disponible, sin embargo, si, por la razón que fuese, necesitásemos también mostrar la imagen en función de su densidad, lo único que haría falta es añadir el indicador de densidad seguido de la letra X.

```

```

Pero, ¿qué pasa si lo que se desea es también manipular el tamaño de la imagen? Pues para ello, disponemos del atributo SIZES.

```


```

Al insertar en un documento HTML esta declaración, lo que se está indicando al sistema es que se dispone de una imagen con tres posibles tamaños, los cuales, serán seleccionados si se cumplen las condiciones marcadas por SRCSET. En otras palabras, si el tamaño de la ventana gráfica (el VIEWPORT) tiene, como máximo el tamaño establecido por el parámetro W, su imagen se seleccionará.

Otra cosa es el parámetro SIZES, el cual, indica exactamente qué tamaño se desea que tenga la imagen en relación con el tamaño de la ventana gráfica expuesta en la condición de medios. En nuestro ejemplo, la expresión entre paréntesis es la condición de medios y, el valor a su derecha, el tamaño que se desea que tenga la imagen en esas circunstancias. Por ejemplo, si el tamaño de la ventana es menor de 800 píxeles, la imagen deberá tener un ancho de 100% del tamaño de la ventana.

En lo referente al atributo SRC, debe establecerse siempre porque, aunque las imágenes se seleccionen por SRCSET, puede que el agente de usuario no soporte esta característica y, si no está definido, no se mostrará imagen alguna.

## NOTA

El atributo SIZES, no funcionará si el elemento IMG tiene asignada una anchura o altura, independientemente de si se establece por estilos en línea o por reglas CSS. Es decir, si se desea utilizar esta técnica, las propiedades HEIGHT y WIDTH del elemento IMG deben estar establecidas a INITIAL o INHERIT.

## Selección de imágenes mediante PICTURE y SOURCE

Como comentamos anteriormente, el elemento PICTURE fue diseñado con la idea de proporcionar soporte nativo a imágenes responsive o adaptativas. Para conseguir este objetivo, lo habitual es recurrir al elemento SOURCE, el cual permite las mismas capacidades que el atributo SRCSET del elemento IMG.

Supongamos, como antes, una imagen de la que disponíamos hasta cuatro versiones. Para conseguir nuestro objetivo mediante el uso de PICTURE y SOURCE, lo que podría hacer es algo como lo siguiente:

```

<picture>
<source srcset="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=8ce914b8731af11d1a24e16da09609b8" media="(min-
width: 2000px" />
<source srcset="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=40e869d6ec0988b88dd639db472b251e" media="(min-
width: 1360px" />
<source srcset="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=15cb1c5b1d6410df2477a608a6b418d3" media="(min-
width: 768px" />

</picture>
```

Al insertar en un documento HTML esta declaración, lo que se está indicando al sistema es que se dispone de una imagen con cuatro posibles tamaños, los cuales, serán seleccionados si se cumplen las condiciones marcadas por el atributo MEDIA. En otras palabras, si el tamaño de la ventana gráfica (el VIEWPORT) tiene, como máximo el tamaño establecido por el atributo MEDIA, la imagen definida en el atributo SRCSET se seleccionará.

Es importante destacar que, las declaraciones del elemento SOURCE deben estar en un orden determinado en función de su condición de medios. Si la condición de medios, el atributo MEDIA, está aplicando a través de MIN-WIDTH, como es este último ejemplo, deben declararse de mayor a menor resolución. Pero, si la condición de medios que se está aplicando va definida a través de MAX-WIDTH, deben declararse de menor a mayor.

Para poder comparar y verlo mejor, vamos a ver cómo sería la declaración del elemento PICTURE anterior, pero utilizando la condición de MAX-WIDTH, en vez de MIN-WIDTH.

```
<picture>
<source srcset="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=ad483702507f5f5e5bf8e9287dfa3f3f" media="(max-width: 768px" />
<source srcset="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=15cb1c5b1d6410df2477a608a6b418d3" media="(max-width: 1360px" />
<source srcset="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=40e869d6ec0988b88dd639db472b251e" media="(max-width: 2000px" />

</picture>
```

Aunque, el elemento SOURCE admite el atributo SIZES, éste, no se aplicará porque el elemento PICTURE lo ignora cuando SOURCE es un descendiente directo. Sin embargo, si, por la razón que fuese, necesitásemos también mostrar la imagen en función de su densidad, lo único que haría falta es añadir el indicador de densidad seguido de la letra X.

```

```

En lo referente al atributo SRC, debe establecerse siempre porque, aunque las imágenes se seleccionen por SRCSET, puede que el agente de usuario no soporte esta característica y, si no está definido, no se mostrará ninguna imagen.

## VIDEOS RECEPTIVOS

Aunque es posible hacer una manipulación de igual forma que el elemento PICTURE, es decir, estableciendo varios elementos SOURCE con sus respectivas resoluciones dentro de una estructura VIDEO, lo habitual es que sólo se establezcan unos pocos estilos y sea el usuario el que escoja la resolución del video que desea ver.

De hecho, muchos sistemas multimedia suelen precargar el vídeo de menor tamaño y añadir un desplegable con las diferentes resoluciones disponibles. Con esta posibilidad, el usuario puede escoger la resolución que más se ajusta a sus deseos y se cubren algo más los requerimientos de usabilidad y accesibilidad web.

Dicho esto, una posible forma de hacer que este tipo de recursos se vuelvan adaptables a cualquier dispositivo es hacer lo siguiente:

```
video {  
width: 100%;  
display: block;  
object-fit: cover;  
height: 100vh;  
}  
  
<video src="https://api.perlego.com/assets/asset?  
session=73694023674e702820461842421bcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=ef3388d6915435a170c7acedfb443ab8"  
type="video/mp4"  
loop=""  
muted=""  
controls=""  
autoplay="">  
Tu navegador no admite el elemento <code>video</code>. Puedes descargar el video desde la URLhttps://www.pexels.com/video/beautiful-  
timelapse-of-the-night-sky-with-reflections-in-a-lake-857251/  
</video>
```

Si nos fijamos en el código anterior, sólo hace falta manejar las propiedades del objeto, como si de una imagen se tratase, es decir, establecer el objeto al tamaño máximo deseado (en este caso al 100% del ancho y alto de la ventana gráfica) y habilitar el ajuste del elemento con respecto a su contendor con la propiedad OBJECT-FIT.

Por desgracia, no siempre es posible agregar contenidos multimedia a través de esta estructura y debemos recurrir a otros métodos.

Este es el caso de YouTube o Vimeo, los cuales requieren de un elemento OBJECT o IFRAME para poder incrustarse en nuestra página. Ambos casos, se verán con detalle, cuando lleguemos al capítulo de iframes y objetos.

## USABILIDAD Y ACCESIBILIDAD EN LOS ELEMENTOS MULTIMEDIA

### En imágenes

Uno de los factores que más ralentiza la carga de los documentos o páginas web son las imágenes. Sin embargo, son necesarias porque, entre otras cosas, tienen la cualidad de atraer y proporcionar una sensación

de claridad, modernidad y simplicidad, siempre que se gestionen bien.

Para que una imagen se vuelva usable y accesible, lo primero que hay que hacer es establecer un texto alternativo a través de su propiedad ALT. Esto proporcionará información a los usuarios si la imagen no puede ser cargada o si han deshabilitado la carga de imágenes en el agente de usuario.

El tamaño de las mismas debe optimizarse. Esto es, la compresión de las imágenes con la mínima pérdida posible de datos relevantes para disminuir el tiempo de latencia. Los métodos más frecuentes para reducir el tamaño de las imágenes son disminuir el número de colores utilizado, aumentando su compresión o reducir la resolución.

Si el uso de la imagen tiene un objetivo decorativo, se debe intentar eliminarla porque este tipo de recursos aumentan el peso de las páginas y, por consiguiente, el tiempo de latencia. Además, perjudican a la accesibilidad web.

Si el objetivo de la imagen es definir un banner o capa publicitaria, debe situarse, sobre todo, al principio de la página de inicio o home ya que el ratio de CTR aumenta en este tipo de páginas. No obstante, debe tener el tamaño adecuado para que llame la atención, pero sin restarle importancia al resto de contenidos. Además, no debe contener colores muy brillantes puesto que pueden perjudicar seriamente la accesibilidad web.

Si el objetivo de la imagen es definir un carrusel de imágenes o slider, es importante que se traten como si un banner se tratase y hacer que todas las animaciones sean realizadas a través de estilos CSS o a través de gráficos basados en vectores escalables (SVG). Además, debe realizar las transiciones entre imágenes de forma suave, disponer de botones de parada y reanudación y, en general, no debe contener más de 5 imágenes.

Si el objetivo de la imagen es definir algo como una infografía, debe ofrecerse, al menos, dos versiones. Una con un tamaño adecuado de forma que no ralentice la carga y otra, a petición de usuario, con un tamaño lo suficientemente grande como para que se vuelva legible.

## **En audio y vídeo**

El uso de elementos multimedia en un documento web debe estar limitado al justo y necesario, es decir, no se deben incluir elementos que no sean estrictamente necesarios.

Ningún elemento multimedia debe empezar a reproducirse de forma automática, es decir, siempre deben cargarse sin reproducir ninguna imagen ni sonido en movimiento.

Tampoco deben contener cambios lumínicos bruscos, como flashes, para evitar que se puedan producir daños a personas con enfermedades como la epilepsia fotosensible.

Si el objetivo del elemento multimedia es un audio, se debe proporcionar una transcripción del elemento a través de TRACK u otra opción similar.

Si el objetivo del elemento multimedia es un vídeo o presentación multimedia, se debe proporcionar la transcripción del audio a través de TRACK u otra opción similar y una descripción del vídeo que incluya las imágenes y su contenido.

También puede ser necesario incluir una versión adicional que contenga la descripción del audio en lengua de signos, lo cual facilita mucho el buen entendimiento a aquellas personas que poseen una discapacidad o incapacidad para oír.

## **PRÁCTICA 4: ESTRUCTURA, TEXTOS E IMÁGENES DE “UNIVERSES”**

### **Código del ejemplo**

<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-04>



### **Objetivo de la práctica**

Estructurar los diferentes contenidos para que se encuentren organizados y jerarquizados, y añadir los textos, títulos e imágenes adecuadas en cada apartado.

## Resultado

UNIVERSES

UniversES  
Todo el conocimiento del universo a tu alcance

INICIO

Vía Láctea desde la Tierra

Da noche se ve como una borrosa banda de luz blanca alrededor de toda la esfera celeste. El fenómeno visual de la Vía Láctea se debe a estrellas y otros materiales que se hallan sobre el plano de la galaxia, como el gas interestelar. Vía Láctea aparece más brillante en la dirección de la constelación de Sagitario, ya que hacia allí se ubica su núcleo.

Seguir leyendo

Nebulosa del Águila

La nebulosa del Águila es parte del objeto astronómico catalogado como M16, es decir el objeto 16 del catálogo de Messier. M16 está conformado por la nebulosa y un cúmulo estelar abierto asociado con ella, catalogado como NGC 6611, y cuyas estrellas se aprecian en las distintas imágenes de M16. Se encuentra en la constelación Serpens (la serpiente).

Seguir leyendo

COMENTARIOS

- Pablo en Vía Láctea desde la Tierra
- Elena en Galaxia de Andrómeda
- Pedro en Nebulosa del Águila
- Isabel en Galaxia del Sombrero

ARCHIVOS

- Enero 2020
- Febrero 2020
- marzo 2020
- Abril 2020

Galaxia de Andrómeda

La galaxia de Andrómeda, también conocida como Galaxia Espiral M31, Messier 31 o NGC 224, es una galaxia espiral con un diámetro de doscientos veinte mil años luz (en lo que concierne a su halo galáctico) y de unos ciento cincuenta mil años luz entre los extremos de sus brazos.

Seguir leyendo

Galaxia del sombrero

La galaxia del Sombrero (también conocida como objeto Messier 104, Messier 104, o NGC 4594), es una galaxia lenticular de la constelación de Virgo a una distancia de 28 millones de años luz. Fue descubierta por Pierre Méchain en 1781.

Seguir leyendo

UNIVERSES

f t i n

(c) Copyright 2020 Aviso Legal Política de Privacidad Contactar

## Recursos para hacer la práctica

### Recurso

<https://pixabay.com/es/photos/nebulosa-de-orión-nebulosa-de-emisión-11107/>



### *Para qué este recurso*

Imagen de fondo a utilizar en la parte superior de nuestra página.

En esta dirección se nos da la posibilidad de descargar cuatro versiones de la imagen y, dado que vamos a jugar con técnicas receptivas o adaptativas, es importante se descarguen las cuatro.

### **Recurso**

<https://pixabay.com/photos/british-columbia-canada-clear-lake-2382640/>



### *Para qué este recurso*

Imagen para el primer artículo, titulado “Vía Láctea desde la Tierra”. La resolución a descargar será la de 1280x854 y su texto alternativo será el “Vía-Láctea-desde-la-columbia-británica”.

### **Recurso**

<https://pixabay.com/photos/eagle-nebula-ic-4703-fog-11172/>



### *Para qué este recurso*

Imagen para el segundo artículo, titulado “Nebulosa del Águila”. La resolución a descargar será la de 1280x800 y su texto alternativo será el “M16, NGC-6611”.

### **Recurso**

<https://pixabay.com/photos/m31-space-astronomy-astronomical-3613931/>



### *Para qué este recurso*

Imagen para el tercer artículo, titulado “Galaxia de Andrómeda”. La resolución a descargar será la de 1280x853 y su texto alternativo será el “M31, NG-C224”.

## **Recurso**

<https://pixabay.com/es/photos/galaxy-niebla-de-sombrero-67712/>



*Para qué este recurso*

Imagen para el cuarto artículo, titulado “Galaxia del sombrero”. La resolución a descargar será la de 1280x717 y su texto alternativo será el “M104, NGC-4594”.

En todas las imágenes, se añadirá un evento ONERROR que añadirá una clase llamada ERROR. Esto nos permitirá saber a primera vista, y sea por la razón que sea, si se ha producido un error en la carga o no se ha encontrado la imagen.

Y, en lo referente a los textos que necesitamos, los recursos son:

## **Recurso**

[https://es.wikipedia.org/wiki/Vía\\_Láctea](https://es.wikipedia.org/wiki/Vía_Láctea)



*Para qué este recurso*

Texto para el primer artículo, titulado “Vía Láctea desde la Tierra”. La parte a introducir será el primer párrafo de la sección “Vista desde la Tierra”.

## **Recurso**

[https://es.wikipedia.org/wiki/Nebulosa\\_del\\_Aguila](https://es.wikipedia.org/wiki/Nebulosa_del_Aguila)



*Para qué este recurso*

Texto para el segundo artículo, titulado “Nebulosa del Águila”. La parte a introducir será el primer párrafo, hasta “(la serpiente)”.

## **Recurso**

[https://es.wikipedia.org/wiki/Galaxia\\_de\\_Andr%C3%B3meda](https://es.wikipedia.org/wiki/Galaxia_de_Andr%C3%B3meda)



### *Para qué este recurso*

Texto para el primer artículo, titulado “Galaxia de Andrómeda”. La parte a introducir será el primer párrafo de, hasta el primer punto y seguido.

## **Recurso**

[https://es.wikipedia.org/wiki/Galaxia\\_del\\_Sombrero](https://es.wikipedia.org/wiki/Galaxia_del_Sombrero)



### *Para qué este recurso*

Texto para el primer artículo, titulado “Galaxia del Sombrero”. La parte a introducir será el primer párrafo completo.

## **Resolución**

Lo primero que se debe hacer es estructurar los diferentes contenidos estableciendo, o creando si procede, las carpetas IMG, CSS, JS y PAGES. Las imágenes irán en la carpeta llamada IMG, los estilos en la carpeta CSS, los JavaScript en la carpeta JS y, las páginas enlazables desde ésta, en la carpeta PAGES.

Una vez terminado con el proceso de estructuración, en primer lugar, añadiremos, en la regla IMG, la propiedad OBJECT-FIT con el valor COVER, para conseguir que la imagen se ajuste a las proporciones del contenedor.

En segundo lugar, como lo que se desea es que la imagen del banner se ajuste de igual manera que las imágenes de cabecera de los artículos, lo que haremos es modificar la propiedad BACKGROUND de la regla .BANNER y añadir unas cuantas casuísticas a través de consultas de medios al final de las declaraciones referentes a este componente.

Estas consultas de medios, únicamente, modificarán la imagen a cargar dependiendo de si estamos en una resolución de 800 píxeles o menos, si estamos en una resolución de portátil (es decir, 1366 píxeles) o en una resolución de escritorio como pueda ser Full HD.

Aunque parezca que hemos terminado, no hacen falta unos pequeños retoques para hacer que el menú de navegación se muestre correctamente siempre y que la barra lateral que se muestra en el modo receptivo se oculte cuando se pulsa fuera de la misma. Para ello se debe recurrir a JavaScript y los eventos ONSCROLL y ONCLICK del objeto WINDOW.

## PRÁCTICA 5: BANNER DE “UNIVERSES” A PANTALLA COMPLETA

### Código del ejemplo

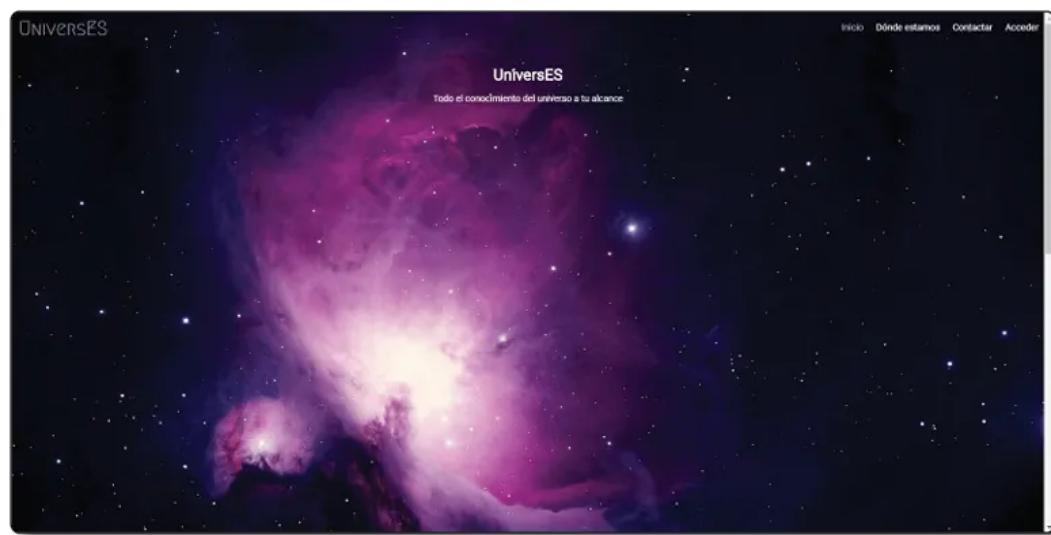
<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-05>



### Objetivo de la práctica

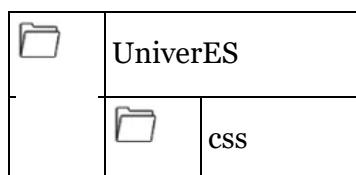
Proporcionar a la página de inicio de UniversES un aspecto más actual estableciendo la imagen del banner a pantalla completa.

### Resultado



### Resolución

Llegados a este punto, ya deberíamos tener una carpeta con la una estructura similar, si no igual, a la siguiente:



		img
		js
		pages
		index.html

Si todo se ha realizado como debe, en la carpeta CSS tendremos ubicado el archivo STYLES.CSS con el código de la tercera y cuarta práctica. En la carpeta IMG tendremos ocho archivos, las cuatro versiones de la imagen del banner y las cuatro imágenes correspondientes a los artículos. En la carpeta JS tendremos un nuevo archivo llamado SCRIPTS.JS que contendrá el código JavaScript que se fue definido en la práctica anterior y, luego, además, tendremos una carpeta denominada PAGES que estará vacía, a espera de las siguientes prácticas.

Dicho esto, para llevar a cabo nuestro objetivo en esta práctica, sólo necesitamos realizar tres pequeños cambios en el CSS.

Como ya tenemos definidas las consultas de medios necesarias para el banner y, queremos que ocupe toda la pantalla, lo único que hay que hacer es eliminar el borde inferior, para que no se vea una línea horizontal en la parte inferior de la pantalla, reposicionar los textos del banner y cambiar la altura del banner a 100VH.

Para la eliminación del borde inferior, simplemente, estableceremos la propiedad BORDER-BOTTOM a 0 NONE y, para reposicionar los textos del componente, lo queharemos es ir a la regla .BANNER P y establecerle un PADDING-TOP de 45VH, en vez de un PADDING-TOP de 10VH como tenía antes.

# 5

## TABLAS

Las tablas no son nada más que una forma de organizar la información a través de filas y columnas. El problema reside cuando estas estructuras contienen mucha información y no pueden representarse de manera correcta en dispositivos con poca resolución o de pequeño tamaño.

En este capítulo vamos a ver cómo definir tablas, cómo hacerlas decorativas, cómo hacerlas adaptativas o Responsive y cómo hacerlas usables y accesibles.

# ELEMENTOS DISPONIBLES EN HTML5

## Elemento caption

El elemento CAPTION especifica que el contenido que se va a representar es el título de una tabla.

Sólo puede definirse un elemento CAPTION por tabla y es importante que el elemento CAPTION sea el primer hijo directo del elemento TABLE.

```
<table>
<caption>Título de la tabla</caption>
<tbody>
...
</tbody>
</table>
```

## Elemento table

El elemento TABLE especifica que el contenido que se va a representar es una estructura de datos tabulados en forma de filas y columnas, es decir, una tabla.

```
<table summary="Lista de usuarios dados de alta en ejemplo.com">
<caption>Usuarios permitidos</caption>
<thead>
<tr>
<th>ID</th>
<th>Nombre</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Pablo</td>
</tr>
<tr>
<td>02</td>
<td>Pedro</td>
</tr>
</tbody>
</table>
```

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>border</b>	Especifica el tamaño en píxeles del borde que se debe añadir a la tabla y a las celdas a través de un valor entero. En general, este valor se establece a 0 y se modifica a través de CSS.
<b>cellpadding</b>	Especifica el espacio que debe haber entre el contenido de la celda y su borde, se muestre o no. Para lograr un código más limpio se puede establecer, mediante CSS, la propiedad BORDER-COLLAPSE al elemento TABLE y un margen interno a los elementos TH y TD.
<b>cellspacing</b>	Especifica el espacio que debe haber entre dos celdas. Este valor se puede establecer de manera porcentual o en píxeles. Para lograr un código más limpio se puede establecer, mediante CSS, la propiedad BORDER-SPACING al elemento TABLE.
<b>summary</b>	Especifica un texto alternativo que resume el contenido de la tabla.
<b>width</b>	Especifica la anchura de la tabla.

Las tablas es uno de los elementos de HTML menos accesibles que, a menudo, encontramos en las páginas. Primero porque los desarrolladores no conocen todas las posibilidades de configuración y, segundo, porque si no se ve toda ella en su conjunto puede ser algo muy difícil de entender o contextualizar. Como ejemplo extremo, piénsese que, si un usuario sólo puede ver un dato en una tabla que, además, no presenta una cabecera por la circunstancia que sea, puede no saber a qué se refiere dicho dato.

Por tanto, si se han de utilizar, se deben especificar las dimensiones en términos de porcentaje y establecer todas sus propiedades para que no se pierda semántica y/o accesibilidad.

La declaración de los elementos de cabecera y pie de tabla (THEAD y TFOOT) deben establecerse antes que el elemento del contenido de la tabla TBODY para que el agente de usuario pueda renderizar la información de contexto antes de recibir el detalle con todas las filas de datos, que pueden ser muchas.

Cabe destacar que, los atributos SUMMARY, ID, HEADERS y SCOPE y, el elemento CAPTION, no tienen ningún efecto visual, sin embargo, son muy útiles para las tecnologías asistivas como los lectores de pantalla puesto que aclaran y fortalecen su significado.

## Elemento colgroup

El elemento COLGROUP especifica que el contenido que se va a representar es un grupo de una o más columnas de una tabla. Suele ser útil para aplicar estilos de forma agrupada en vez de tener que repetirlos de uno en uno.

Es importante que el elemento COLGROUP sea hijo directo del elemento TABLE, que esté declarado justo después del elemento CAPTION y justo antes de los elementos THEAD, TBODY o TFOOT porque, de no ser así, puede afectar a la usabilidad web y a la accesibilidad web.

Para especificar o definir las propiedades de cada columna dentro de cada elemento COLGROUP se debe utilizar el elemento COL. Este elemento sólo permite el atributo SPAN para definir el número de columnas que debe abarcar.

```
<table summary="Lista de usuarios dados de alta en ejemplo.com">
<caption>Usuarios permitidos</caption>
<colgroup>
<col style="background: whitesmoke;"></col>
<col span="2" style="background: lavender;"></col>
</colgroup>
<thead>
<tr>
<th>ID</th>
<th>Nombre</th>
<th>Apellido</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Pablo</td>
<td>García</td>
</tr>
<tr>
<td>02</td>
<td>Pedro</td>
<td>Rodríguez</td>
</tr>
</tbody>
</table>
```

## **Elemento thead**

El elemento THEAD especifica que el contenido que se va a representar es el encabezado de una tabla.

```
<thead>
<tr>
<th>ID</th>
<th>Nombre</th>
<th>Apellido</th>
</tr>
</thead>
```

Cabe destacar que elemento THEAD debe declararse justo después del elemento CAPTION y COLGROUP y justo antes de los elementos TBODY y TFOOT. Además, no se debe omitir puesto que su omisión puede perjudicar de forma notable a la usabilidad web y a la accesibilidad web de la página.

## **Elemento tbody**

El elemento TBODY especifica que el contenido que se va a representar es el cuerpo de una tabla.

```
<tbody>
<tr>
<td>o1</td>
<td>Pablo</td>
<td>García</td>
</tr>
</tbody>
```

Cabe destacar que elemento TBODY debe declararse justo después de los elementos THEAD y TFOOT. Además, no se debe omitir puesto que su omisión puede perjudicar de forma notable a la usabilidad web y a la accesibilidad web de la página.

## **Elemento tfoot**

El elemento TFOOT especifica que el contenido que se va a representar es el pie de página de una tabla.

```
<tfoot>
<tr>
<td colspan="2">Número de usuarios</td>
<td>3</td>
</tr>
</tfoot>
```

Cabe destacar que elemento TFOOT debe declararse justo después del elemento CAPTION y COLGROUP y justo antes del elemento TBODY. Sin embargo, a diferencia del elemento THEAD, el elemento TFOOT puede omitirse, aunque su omisión puede perjudicar a la accesibilidad web de la página.

## **Elemento tr**

El elemento TR especifica que el contenido que se va a representar es una fila perteneciente a un encabezado, cuerpo o pie de página una tabla.

```
<tr>
<td>o2</td>
<td>Pedro</td>
<td>Rodríguez</td>
</tr>
```

## Elemento th

El elemento TH especifica que el contenido que se va a representar es una celda de encabezado.

```
<tr>
<th>ID</th>
<th>Nombre</th>
<th>Apellido</th>
</tr>
```

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>abbr</b>	Especifica una abreviatura que representa el contenido de la celda. Este atributo está obsoleto, pero su función puede ser trasladada al atributo TITLE.
<b>colspan</b>	Especifica el número de columnas que se deben unificar.
<b>headers</b>	Especifica una lista de identificadores únicos que se corresponden con los atributos ID pertenecientes a los elementos TH. Los valores contenidos deben ir separados por espacios en blanco. Aunque es posible definir este atributo en los elementos TH, su uso está más destinado a los TD. No obstante, las tablas más complejas pueden llegar a utilizarlo en ambos elementos para volver más accesibles las tablas.
<b>id</b>	Especifica el identificador de la columna. Este valor es necesario para utilizarlo con el atributo HEADERS del elemento TD.
<b>rowspan</b>	Especifica el número de filas que se deben unificar.
<b>scope</b>	Especifica un único valor que vincula la información entre las celdas de la cabecera y las celdas de datos. Es decir, especifica si una celda de encabezado es un encabezado para una columna, una fila o un grupo de columnas o filas. Sus posibles valores son ROW, para especificar que la celda es un encabezado para una fila, COL, para especificar que la celda es un encabezado para una columna, ROWGROUP, para especificar que la celda es un encabezado para un grupo de filas y COLGROUP, para especificar que la celda es un encabezado para un grupo de columnas.

Cabe destacar que elemento TH debe declararse, únicamente, en la estructura THEAD para evitar problemas semánticos y de accesibilidad web.

## Elemento td

El elemento TD especifica que el contenido que se va a representar es una celda de datos.

```
<tr>
<td>02</td>
<td>Pedro</td>
<td>Rodríguez</td>
</tr>
```

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>colspan</b>	Especifica el número de columnas que se deben unificar.
<b>headers</b>	Especifica una lista de identificadores únicos que se corresponden con los atributos ID pertenecientes a los elementos TH. Los valores contenidos deben ir separados por espacios en blanco. Aunque es posible definir este atributo en los elementos TH, su uso está más destinado a los TD. No obstante, las tablas más complejas pueden llegar a utilizarlo en ambos elementos para volver más accesibles las tablas.
<b>rowspan</b>	Especifica el número de filas que se deben unificar.

## ELEMENTOS DISPONIBLES EN CSS

A continuación, se muestran las propiedades de CSS que están expresamente dedicadas a tablas.

### Propiedad border-collapse

Especifica si se deben fusionar o separar los bordes del elemento. Entre sus posibles valores podemos encontrar:

- **COLLAPSE**: Indica que los bordes deben fusionarse cuando sea posible. Este valor puede no ser efectivo cuando se encuentra en conjunción con las propiedades EMPTY-CELLS y BORDER-SPACING.


- **SEPARATE**: Indica que los bordes deben mostrarse separados e independientes para cada celda o elemento. Es el valor por defecto.


#### Ejemplos:

```
ul { border-collapse: separate; }
ol { border-collapse: collapse; }
```

#### NOTA

La propiedad BORDER-COLLAPSE sólo es válida para los elementos TABLE, TH y TD.

### Propiedad border-spacing

Especifica la distancia entre los bordes de las celdas adyacentes, siempre y cuando la propiedad BORDER-COLLAPSE esté establecida a SEPARATE. Sus posibles se deben asignar a través de un valor establecido en una de las medidas permitidas de CSS.

#### Ejemplos:

```
#table1 { border-spacing: 15px; }
#table2 { border-spacing: 2vh; }
```

#### NOTA

La propiedad BORDER-COLLAPSE sólo es válida para los elementos TABLE, TH y TD.

### Propiedad caption-side

Especifica la posición del título de una tabla. Entre sus posibles valores podemos encontrar:

- **BOTTOM**: Indica que el título debe estar debajo de la tabla.
- **TOP**: Indica que el título debe estar encima de la tabla. Es el valor por defecto.

## Ejemplo:

```
table { caption-side: bottom; }
```

## Propiedad empty-cells

Especifica si se deben mostrar o no los bordes de las celdas vacías. Entre sus posibles valores podemos encontrar:

- **HIDE**: Indica que NO se deben mostrar.
- **SHOW**: Indica que se deben mostrar. Es el valor por defecto.

## Ejemplo:

```
table { empty-cells: hide; }
```

## Propiedad table-layout

Especifica el modo en el que se tienen que diseñar celdas, filas y columnas de la tabla. Entre sus posibles valores podemos encontrar:

- **AUTO**: Indica que el ancho de la columna debe establecerse sin romper el texto o contenido de las celdas.
- **FIXED**: Indica que el ancho de la tabla será gestionado por el usuario y que, las columnas, deberán ser gestionadas por el ancho de las celdas de la primera fila. Si no se estableciesen anchos en la primera fila, los anchos de las columnas se dividirán por partes iguales, independientemente de su contenido.

## Ejemplo:

```
table { table-layout: fixed; }
```

## CREACIÓN DE TABLAS RESPONSIVE

Las tablas son, quizás, el componente menos flexible que ofrece HTML. Sin embargo, gracias a CSS y JavaScript es posible hacer que esta característica se vuelva algo menos rígida. Por ejemplo, supongamos una tabla de datos como la siguiente:

EJEMPLO DE TABLA ADAPTATIVA						
ID	Empresa	F. Movimiento	Tipo	Concepto	Importe	Estado
1	Consultores SA	30-12-2019	Ingreso	Nómina	+1268.00 €	Efectuado
2	Carrefour	01-01-2020	Recibo	Supermercado	-128.56 €	Efectuado
3	El Corte Inglés	03-01-2020	Recibo	Chaqueta hombre L-XL	-99.99 €	Pendiente
4	El Corte Inglés	03-01-2020	Recibo	Pantalón hombre M-L	-48.50 €	Pendiente

Si probásemos esta tabla en un dispositivo móvil, lo más probable es que viésemos una barra de desplazamiento horizontal y, dependiendo de la resolución del dispositivo, de los tamaños de fuente y de los estilos agregados, puede que hasta prácticamente nada de información útil.

Para solucionar este supuesto, a continuación, se muestran algunas de las técnicas para hacer que las tablas de HTML puedan verse en cualquier dispositivo sin perder legibilidad e independientemente de sus dimensiones o densidad.

## Mediante barras de desplazamiento

Esta técnica consiste en definir normalmente la tabla y aplicarle unas consultas de medios cuando se produce una condición determinada, como pueda ser el ancho del dispositivo.

Es la técnica más sencilla de todas, pero no la que mejor se adapta a las condiciones del dispositivo y consiste en ajustar el tamaño de la tabla al 100% del ancho del dispositivo, cambiar el modo de representación de la tabla a BLOCK, en vez de TABLE, y habilitar el desplazamiento horizontal en la misma.

## Código HTML

```
<table id="table01"
summary="Ejemplo de tabla adaptativa">
<caption>EJEMPLO DE TABLA ADAPTATIVA</caption>
<thead>
<tr>
<th scope="col">ID</th>
<th scope="col">Empresa</th>
<th scope="col">F. Movimiento</th>
<th scope="col">Tipo</th>
<th scope="col">Concepto</th>
<th scope="col">Importe</th>
<th scope="col">Estado</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Consultores SA</td>
<td>30-12-2019</td>
<td>Ingreso</td>
<td>Nómina</td>
<td>+1268.00 € </td>
<td>Efectuado</td>
</tr>
<tr>
<td>2</td>
<td>Carrefour</td>
<td>01-01-2020</td>
<td>Recibo</td>
<td>Supermercado</td>
<td>-128.56 €</td>
<td>Efectuado</td>
</tr>
<tr>
<td>3</td>
<td>El Corte Inglés</td>
<td>03-01-2020</td>
<td>Recibo</td>
<td>Chaqueta hombre L-XL</td>
<td>-99.99 € </td>
<td>Pendiente</td>
</tr>
<tr>
<td>4</td>
<td>El Corte Inglés</td>
<td>03-01-2020</td>
<td>Recibo</td>
<td>Pantalón hombre M-L</td>
<td>-48.50 € </td>
<td>Pendiente</td>
</tr>
</tbody>
</table>
```

## Código CSS

```
@import url('https://fonts.googleapis.com/css?family=Roboto&display=swap');
* { box-sizing: border-box; outline: 0 none; }
html,
body { margin: 0; padding: 0; font-family: 'Roboto', sans-serif;
display: block; font-size: 14px;
}
table { border: 1px solid rgba(0,0,0,0.2); border-spacing: 2px;
margin: 10px 0; }
table caption { background: #000000; color: #fff; font-size: 1.0rem;
font-weight: bold; line-height: 1.5; padding: 0; }
table td,
table th { border: 1px solid rgba(0,0,0,0.2); border-spacing: 0;
font-size: 1rem; padding: 5px; text-align: left;
margin: 2px 0; white-space: nowrap; }
table thead th:nth-child(6),
table tbody td:nth-child(6){ text-align: right; }
@media screen and (max-width: 620px) {
table { display: block; overflow-x: auto; width: 100%; }
}
```

En lo referente a este último código de CSS, el cambio de comportamiento se realiza cuando la resolución llega al valor de 620 píxeles. Esto se debe a que como la tabla no tiene demasiados campos, la información encapsulada en ella se muestra sin pérdida hasta llegar a esa resolución. Sin embargo, si la tabla tuviese más campos, o las descripciones fuesen más largas, el valor para visualizar la tabla de manera completa se vería incrementado provocando que el valor de la consulta de medios cambie.

## Resultado

EJEMPLO DE TABLA ADAPTATIVA						
ID	Empresa	F. Movimiento	Tipo	Concepto	Importe	Estado
1	Consultores SA	30-12-2019	Ingreso	Nómina	+1268.00 €	Efectuado
2	Carrefour	01-01-2020	Recibo	Supermercado	-128.56 €	Efectuado
3	El Corte Inglés	03-01-2020	Recibo	Chaqueta hombre L-XL	-99.99 €	Pendiente
4	El Corte Inglés	03-01-2020	Recibo	Pantalón hombre M-L	-48.50 €	Pendiente

## Mediante consultas de medios

Al igual que sucede con la técnica de la barra de desplazamiento horizontal, esta técnica consiste en definir normalmente la tabla y aplicarle unas consultas de medios cuando se produce una condición determinada, como pueda ser el ancho del dispositivo.

Aunque esta técnica resulta ser algo más tediosa y laboriosa, es la que mejor se adapta a las condiciones del dispositivo si no se desea recurrir a JavaScript. Por ello, es una de las más utilizadas en situaciones reales.

La técnica consiste en ajustar el tamaño de la tabla al 100% del ancho del dispositivo, ocultar los campos de cabecera, cambiar el modo de representación de las celdas y, agregar unos atributos personalizados con los nombres de las columnas para utilizarlos como identificadores de campo.

Estos identificadores de campo serán mostrados a través de pseudo-elemento BEFORE en la parte izquierda de cada celda cuando las condiciones de la consulta de medios se cumplan y, para que los valores de estos campos no pierdan legibilidad, el valor de las celdas se alinearán a la derecha, todo ello, además, con la intención de aprovechar, al máximo, el espacio disponible.

## Código HTML

```
<table id="tableo1"
summary="Ejemplo de tabla adaptativa">
<caption>Ejemplo de tabla adaptativa</caption>
<thead>
<tr>
<th scope="col">ID</th>
<th scope="col">Empresa</th>
<th scope="col">F. Movimiento</th>
<th scope="col">Tipo</th>
<th scope="col">Concepto</th>
<th scope="col">Importe</th>
<th scope="col">Estado</th>
</tr>
</thead>
<tbody>
<tr>
<td data-field="ID">1</td>
<td data-field="Empresa">Consultores SA</td>
<td data-field="F. Movimiento">30-12-2019</td>
<td data-field="Tipo">Ingreso</td>
<td data-field="Concepto">Nómina</td>
<td data-field="Importe">+1268.00 € </td>
<td data-field="Pago">Efectuado</td>
</tr>
<tr>
<td data-field="ID">2</td>
<td data-field="Empresa">Carrefour</td>
<td data-field="F. Movimiento">01-01-2020</td>
<td data-field="Tipo">Recibo</td>
<td data-field="Concepto">Supermercado</td>
<td data-field="Importe">-128.56 €</td>
<td data-field="Pago">Efectuado</td>
</tr>
<tr>
<td data-field="ID">3</td>
<td data-field="Empresa">El Corte Inglés</td>
<td data-field="F. Movimiento">03-01-2020</td>
<td data-field="Tipo">Recibo</td>
<td data-field="Concepto">Chaqueta hombre L-XL</td>
<td data-field="Importe">-99.99 € </td>
<td data-field="Pago">Pendiente</td>
</tr>
<tr>
<td data-field="ID">4</td>
<td data-field="Empresa">El Corte Inglés</td>
<td data-field="F. Movimiento">03-01-2020</td>
<td data-field="Tipo">Recibo</td>
<td data-field="Concepto">Pantalón hombre M-L</td>
<td data-field="Importe">-48.50 € </td>
<td data-field="Pago">Pendiente</td>
</tr>
</tbody>
</table>
```

Como se puede apreciar en el código anterior, lo único que llama la atención es el atributo personalizado DATA-FIELD, que es el valor que, como decíamos antes, se mostrará como dato identificativo (a la izquierda) cuando esté en modo receptivo.

## Código CSS

```
@import url('https://fonts.googleapis.com/css?family=Roboto&display=swap');
* { box-sizing: border-box; outline: 0 none; }
html,
body { margin: 0; padding: 0; font-family: 'Roboto', sans-serif;
display: block; font-size: 14px;
}
table { border: 1px solid rgba(0,0,0,0.2); border-spacing: 2px;
margin: 10px 0; }
table caption { background: #000000; color: #fff; font-size: 1.0rem;
font-weight: bold; line-height: 1.5; padding: 0;
text-transform: uppercase; }
table td,
table th { border: 1px solid rgba(0,0,0,0.2); border-spacing: 0;
font-size: 1rem; padding: 5px; text-align: left;
margin: 2px 0; white-space: nowrap; }
table thead th:nth-child(6),
table tbody td:nth-child(6){ text-align: right; }
@media screen and (max-width: 620px) {
table { width: 100%; }
thead { display: none; }
tr td:first-child { background: #fofofo; font-weight: bold; }
tbody td { display: block; text-align: right; }
tbody td:before { content: attr(data-field); display: block;
float: left; font-weight: bold; padding: 0 10px 0 0;
text-align: left; width: auto; }
}
}
```

En lo referente a este último código de CSS, y al igual que pasaba con la técnica anterior, el cambio de comportamiento se realiza cuando la resolución llega al valor de 620 píxeles. Esto se debe a que como la tabla no tiene demasiados campos, la información encapsulada en ella se muestra sin pérdida hasta llegar a esa resolución. Sin embargo, si la tabla tuviese más campos, o las descripciones fuesen más largas, el valor para visualizar la tabla de manera completa se vería incrementado provocando que el valor de la consulta de medios cambie.

## Resultado

EJEMPLO DE TABLA ADAPTATIVA	
ID	1
Empresa	Consultores SA
F. Movimiento	30-12-2019
Tipo	Ingreso
Concepto	Nómina
Importe	+1268.00 €
Pago	Efectuado
ID	2

## Mediante ocultación de columnas a petición de usuario

Otra de las técnicas que se pueden utilizar es la visualización/ocultación de columnas a petición del usuario. Es, quizás, la menos recurrente porque, además de ser bastante más laboriosa que las demás, puede no ser eficiente ya que el espacio disponible es limitado y el usuario puede no llegar a ver toda la información que desea.

Básicamente, se trata de añadir un botón que funcionará como interruptor para mostrar u ocultar una lista con los campos de la tabla. Cada uno de los elementos de esta lista contendrá un elemento INPUT de tipo CHECKBOX, que

será quién muestre u oculte la columna y un LABEL con el nombre del campo o identificador de la columna.

A ese botón se le asignará un evento ONCLICK que hará, que el elemento de lista que tiene a continuación, se oculte o se muestre para poder seleccionar las columnas que se desean ver.

Cada vez que cambie el valor de cada uno de los CHECKBOX se ejecutará una función en JavaScript que recorrerá la tabla y cambiará a la columna seleccionada su estado de visualización, es decir, ocultará o mostrará la columna en función de si está o no chequeada.

Tanto el botón, como la lista, estarán en la primera fila del cuerpo de la tabla que, por defecto, estará oculta y sólo se mostrará cuando la resolución de la pantalla sea la indicada por la consulta de medios de CSS. En nuestro ejemplo, como antes, se establecerá a 620 píxeles, pero, evidentemente, podrá ser la que se considere.

## Código HTML

```
<table id="table01"
summary="Ejemplo de tabla adaptativa">
<caption>Ejemplo de tabla adaptativa</caption>
<thead>
<tr>
<th scope="col">ID</th>
<th scope="col">Empresa</th>
<th scope="col">F. Movimiento</th>
<th scope="col">Tipo</th>
<th scope="col">Concepto</th>
<th scope="col">Importe</th>
<th scope="col">Estado</th>
</tr>
</thead>
<tbody>
<tr>
<td class="fields-list" colspan="7">
<button type="button"
id="fields"
data-table="table01"
onclick="toggle(this)">
Mostrar/Ocultar Lista de columnas
</button>
<ul id="fieldList">
<li>
<input type="checkbox" id="field01" checked />
<label for="field01">ID</label>
</li>
<li>
<input type="checkbox" id="field02" checked />
<label for="field02">Empresa</label>
</li>
<li>
<input type="checkbox" id="field03" checked />
<label for="field03">F. Movimiento</label>
</li>
<li>
<input type="checkbox" id="field04" checked />
<label for="field04">Tipo</label>
</li>
<li>
<input type="checkbox" id="field05" checked />
<label for="field05">Concepto</label>
</li>

```

```

</li>
<li>
<input type="checkbox" id="field06" checked />
<label for="field06">Importe</label>
</li>
<li>
<input type="checkbox" id="field07" checked />
<label for="field07">Estado</label>
</li>
</ul>
</td>
</tr>
<tr>
<td data-field="ID">1</td>
<td data-field="Empresa">Consultores SA</td>
<td data-field="F. Movimiento">30-12-2019</td>
<td data-field="Tipo">Ingreso</td>
<td data-field="Concepto">Nómina</td>
<td data-field="Importe">+1268.00 € </td>
<td data-field="Pago">Efectuado</td>
</tr>
<tr>
<td data-field="ID">2</td>
<td data-field="Empresa">Carrefour</td>
<td data-field="F. Movimiento">01-01-2020</td>
<td data-field="Tipo">Recibo</td>
<td data-field="Concepto">Supermercado</td>
<td data-field="Importe">-128.56 €</td>
<td data-field="Pago">Efectuado</td>
</tr>
<tr>
<td data-field="ID">3</td>
<td data-field="Empresa">El Corte Inglés</td>
<td data-field="F. Movimiento">03-01-2020</td>
<td data-field="Tipo">Recibo</td>
<td data-field="Concepto">Chaqueta hombre L-XL</td>
<td data-field="Importe">-99.99 € </td>
<td data-field="Pago">Pendiente</td>
</tr>
<tr>
<td data-field="ID">4</td>
<td data-field="Empresa">El Corte Inglés</td>
<td data-field="F. Movimiento">03-01-2020</td>
<td data-field="Tipo">Recibo</td>
<td data-field="Concepto">Pantalón hombre M-L</td>
<td data-field="Importe">-48.50 € </td>
<td data-field="Pago">Pendiente</td>
</tr>
</tbody>
</table>

```

## Código JavaScript

```

<script type="text/javascript">
function toggle(el){
el.nextElementSibling.classList.toggle('visible')
}
// Asignamos a los todos los checkbox un evento ONCHANGE
var items = document.querySelectorAll("#fieldList input");
for (var i = 0; i < items.length; ++i) {

```

```

var item = items[i];
item.onchange = function (e) {
// A través del ID recuperamos la columna a tratar
var col = parseInt(this.id.replace("field", ""));
// Recorremos las filas de la tabla
// Empezamos en 1 porque la primera fila es la del botón
var trs = document.querySelectorAll("#tableo1 tbody tr");
for (var i = 1; i < trs.length; ++i) {
var tr = trs[i];
var td = tr.querySelector("td:nth-child(" + col + ")");
// Si el input está chequeado, mostramos la columna
td.style.display = this.checked ? "" : "none";
}
}
}

// Para que la técnica sea más real y utilizable, se habilita un evento
// ONCLICK en la página para que se oculte
document.body.onclick = function(e){
// Recorremos el árbol de elementos hacia arriba. Si el usuario no
// hizo click en algún elemento que esté dentro del elemento con
// clase "fields-list", ocultamos la lista de columnas
var el = e.target;
while (el &&
el.tagName != "TABLE" &&
!el.classList.contains("fields-list")){
el = el.parentElement;
}
// Si el elemento referenciado por "el" no está definido o no
// contiene la clase "fields-list", eliminamos la clase "visible"
if(!el || !el.classList.contains("fields-list")){
document.querySelector("#fieldList").classList.remove("visible");
}
}
}

</script>

```

## Código CSS

```

@import url('https://fonts.googleapis.com/css?family=Roboto&display=swap');
* { box-sizing: border-box; outline: 0 none; }
html,
body { margin: 0; padding: 0; font-family: 'Roboto', sans-serif;
display: block; font-size: 14px;
}
table { border: 1px solid rgba(0, 0, 0, 0.2); border-spacing: 2px;
margin: 10px 0; }
table caption { background: #000000; color: #fff; font-size: 1.0rem;
font-weight: bold; line-height: 1.5; padding: 0;
text-transform: uppercase; }
table td,
table th { border: 1px solid rgba(0, 0, 0, 0.2); border-spacing: 0;
font-size: 1rem; padding: 5px; text-align: left;
margin: 2px 0; }
table thead th:nth-child(6),
table tbody td:nth-child(6) { text-align: right; }
table .fields-list { display: none; }
@media screen and (max-width: 620px) {
table { width: 100%; }
thead { display: none; }
table .fields-list { display: table-cell; position: relative;
}

```

```

text-align: right; }
table .fields-list ul { background: #ffff;
border: 1px solid rgba(0, 0, 0, 0.2);
box-shadow: 1px 1px 3px 0px rgba(0, 0, 0, 0.2);
display: none; list-style: none; margin: 5px;
padding: 0; position: absolute; right: 0;
top: 28px; width: 128px; }
table .fields-list ul li { display: inline-block; padding: 5px;
width: 100%; }
table .fields-list ul li:nth-child(odd) { background-color: #fofofo; }
table .fields-list ul li * { display: block; float: left;
line-height: 1; padding: 0 5px;
width: 16px; }
table .fields-list ul li label { width: calc(100% - 16px);
text-align: left; }
table .fields-list ul.visible { display: block; }
table .fields-list button { background: #fofofo;
border: 1px solid rgba(0, 0, 0, 0.2);
line-height: 24px; padding: 0 5px; }
}

```

## Resultado

EJEMPLO DE TABLA ADAPTATIVA				
				Mostrar/Ocultar Lista de columnas
1	Consultores SA	Nómina	+1268.0	<input checked="" type="checkbox"/> ID
2	Carrefour	Supermercado	-128.5	<input checked="" type="checkbox"/> Empresa
3	El Corte Inglés	Chaqueta hombre L-XL	-99.9	<input type="checkbox"/> F. Movimiento
4	El Corte Inglés	Pantalón hombre M-L	-48.5	<input type="checkbox"/> Tipo

Checklist of columns:

- ID
- Empresa
- F. Movimiento
- Tipo
- Concepto
- Importe
- Estado

## USABILIDAD Y ACCESIBILIDAD EN LAS TABLAS

La utilización de tablas no es una práctica recomendada en usabilidad ya que, en la mayoría de los casos, basta con dividir el texto a través de saltos de línea, capas, tabulaciones o párrafos y, además, el procesado de las tablas ralentiza la velocidad de carga de la página.

Aun así, si se han de utilizar, los elementos que la componen deben especificar todos los atributos necesarios para que no se pierda semántica y/o accesibilidad. Además, las dimensiones de cada elemento deben especificarse en términos de porcentaje o alguna otra medida de longitud estándar de CSS.

La declaración de los elementos de cabecera THEAD, y pie de tabla TFOOT deben establecerse antes que el elemento del contenido de la tabla TBODY para que el navegador pueda renderizar la información de contexto antes de recibir el detalle con todas las filas de datos, que pueden ser muchas.

Si se desea que la tabla sea accesible, debe especificarse el atributo SUMMARY, que representa un resumen del contenido de la tabla y el elemento CAPTION, que representa el título de la tabla justo después de la declaración del elemento TABLE y antes de la definición del THEAD.

Para que no se pierda compresión y legibilidad, se debe establecer el atributo HEADERS en cada elemento TD, con los valores del atributo ID de cada TH (separados por espacios en blanco) a los que se hace referencia para indicar a qué celda o celdas de encabezado se refiere el contenido.

Otro repunte importante para hacer que las tablas se vuelvan accesibles es establecer el atributo SCOPE. Esto es importante porque indica si una celda de encabezado es un encabezado para una columna, una fila o un grupo de columnas o filas.

## PRÁCTICA 6: PÁGINA “DÓNDE ESTAMOS” DE “UNIVERES”

### Código del ejemplo

<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-06/pages/donde-estamos>



### Objetivo de la práctica

Diseñar una página de contenidos en formato de una columna con imágenes, textos y una tabla.

## **Resultado**

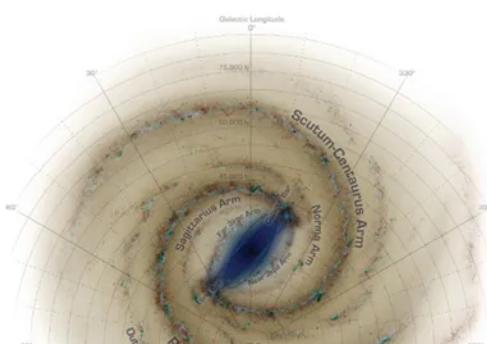


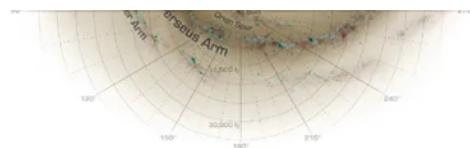
## DÓNDE ESTAMOS

El conocimiento de la ubicación de la Tierra en el universo se ha construido gracias a unos 400 años de observaciones realizadas con telescopio y se ha afinado sustancialmente en el siglo XX. Antiguamente se consideraba que la Tierra era el centro del universo, el cual se creía que estaba formado únicamente por los planetas visibles a simple vista y por una periferia de estrellas fijas.

Después de la aceptación del heliocentrismo en el siglo XVII, las observaciones de William Herschel y otros astrónomos mostraron que el Sol se encontraba dentro de una vasta galaxia con forma de disco y muchas otras estrellas. En el siglo XX, las observaciones de nebulosas espirales por Edwin Hubble revelaron que nuestra galaxia era una de miles de millones en un universo en expansión, agrupadas en cúmulos y supercúmulos. A finales del siglo XX, la estructura general del universo observable se estaba volviendo más clara, con supercúmulos formando en una vasta red de filamentos y vacíos. Los supercúmulos, filamentos y vacíos son las mayores estructuras coherentes en el Universo que podemos observar. A escalas aún más grandes (más de 1.000 megaparsecs) el Universo se vuelve homogéneo, es decir, que todas sus partes tienen, en promedio, la misma densidad, composición y estructura.

Desde que se cree que el universo no tiene ni centro ni límites, no hay un punto de referencia particular con el que trazar la ubicación general de la Tierra en el universo. Se puede hacer referencia a la posición de la Tierra con respecto a estructuras específicas que existen en diversas escalas. Numerosas hipótesis se han formulado sobre nuestro universo como su posible dimensión infinita o su posible pertenencia a un multiverso, sin embargo, aún no se han conseguido evidencias concluyentes sobre esas hipótesis.





Bien, pues la Tierra está localizada en el universo en el Supercúmulo de galaxias Virgo. Un Supercúmulo es un grupo de galaxias juntas por la gravedad. Dentro de este Supercúmulo estamos en un grupo más pequeño de galaxias llamado el Grupo Local. La Tierra está en la segunda galaxia más grande del Grupo Local—una galaxia llamada la Vía Láctea. La Vía Láctea es una galaxia espiral grande. La Tierra está localizada en uno de los brazos espirales de la Vía Láctea (llamado el brazo de Orión) el cual está alrededor de 2/3 partes del camino desde el centro de la galaxia. Nosotros somos parte del Sistema Solar—un grupo de nueve planetas, así como de numerosos cometas y asteroides que orbitan al Sol. Somos el tercer planeta desde el Sol en el Sistema Solar.

## Recurso

<https://pixabay.com/es/illustrations/universo-espacio-vía-láctea-1418354/>



*Para qué este recurso*

Es la imagen de cabecera. La versión que se deberá insertar es la que tiene unas dimensiones de 1920x1012.

Su texto alternativo será el “Vía-Láctea-cabecera”.

## NOTA

Aunque se podría seguir la misma línea de diseño que en la página de inicio, para que se diferencien bien los artículos de lo que es el índice o catálogo inicial, esta imagen la estableceremos con una altura de 250 píxeles en vez de a pantalla completa.

## Recurso

<https://pixabay.com/es/illustrations/vía-láctea-sistema-solar-espacio-60538/>



*Para qué este recurso*

Es la imagen que muestra el esquema de la Vía Láctea”. La versión que se deberá insertar es la que tiene unas dimensiones de 1920x1920.

Su texto alternativo será el “Esquema-Vía-Láctea”.

## **Recurso**

<https://www.publicdomainpictures.net/pictures/90000/velka/planet-earth-14014658804nR.jpg>



*Para qué este recurso*

Esta es la dirección de la imagen del planeta Tierra.

Su texto alternativo será el “Planeta-Tierra”.

En cuanto a la información textual a utilizar, las fuentes de dónde extraer los textos que necesitamos son:

## **Recurso**

[https://es.wikipedia.org/wiki/Ubicaci%C3%B3n\\_de\\_la\\_Tierra\\_en\\_el\\_universo](https://es.wikipedia.org/wiki/Ubicaci%C3%B3n_de_la_Tierra_en_el_universo)



*Para qué este recurso*

Esta es la dirección de dónde extraeremos la información para la primera parte de nuestra página. En concreto, los dos primeros párrafos.

## **Recurso**

<http://legacy.spitzer.caltech.edu/espanol/edu/askkids/earthlocation.shtml>



*Para qué este recurso*

Esta es la dirección de dónde extraeremos la información para la segunda parte de nuestra página que hay que poner justo después de la imagen con el esquema de la Vía Láctea.

## **Resolución**

### **Tipo y formato de página**

La página que se va a diseñar es la que redirige el enlace de “Dónde estamos” de nuestra página inicial de UniversES.

En comparación con dicha página, la diferencia más notable será que estará distribuida en una única columna, en vez de dos como era el caso anterior. Se mantendrán la cabecera y el pie de página, claro está, pero se eliminará la capa lateral referenciada por el elemento ASIDE, en la que se hacía referencia a los comentarios y entradas por mes.

En cuanto al BANNER, se eliminará, puesto que no aporta nada a nivel educativo y hará la práctica más legible. No obstante, la inclusión de banners y anuncios en las páginas puede llegar a ser una herramienta muy útil para promocionar productos o servicios, siempre y cuando, no se vuelvan una distracción y no provoquen molestias visuales ni pérdidas de contexto o continuidad.

Dicho esto, podríamos realizar un nuevo diseño de página sin ninguna base, sin embargo, para conseguir unos mejores pilares en lo que a aprendizaje se refiere, lo que haremos es añadir al elemento BODY una clase denominada ONE-COLUMN-PAGE que nos servirá para ir definiendo los diferentes cambios que se deben aplicar con respecto a los estilos ya definidos en anteriores prácticas y, de esta manera, poder reutilizar la mayor cantidad de código posible.

## Identificación de página

Para que el usuario no pierda la noción de dónde está, en el elemento TITLE de la cabecera del documento, estableceremos un nuevo descriptor de página con el texto “UNIVERSES - DÓNDE ESTAMOS” y, en la sección del menú de navegación cambiaremos la clase CSS ACTIVE del elemento LI que contiene el enlace de INICIO, al elemento LI que contiene el enlace de DÓNDE ESTAMOS.

## Contenido principal

En lo referente al contenido principal, en esta ocasión, insertaremos un elemento ARTICLE con una cabecera (HEADER) que contendrá una imagen, de altura 250 píxeles, y un elemento de título H1, en este orden. El motivo de hacerlo así es porque la nueva página que se va a diseñar tiene un comportamiento y objetivo diferente a la anterior y queremos que se distingan adecuadamente.

De hecho, mientras que, en la página de inicio, la imagen era representativa del título y se mostraba justo encima de la imagen, en esta nueva página, la imagen resulta ser más decorativa y el titular provee la entrada al contenido textual central.

Ahora bien, para que el texto central de la página se vea adecuadamente, antes, debemos hacer que desaparezcan los elementos de ASIDE y BANNER. Efectivamente, lo más sensato es eliminar todo rastro físico de ellos en el HTML, no obstante, si queremos asegurarnos de que no se muestren aun estando presentes, debemos ocultar dichos elementos de forma que no ocupen espacio físico en pantalla. Para ello, crearemos una nueva regla CSS que sobrescriba sus dimensiones a CERO y fuerce que el contenido desbordado no se muestre.

Todo esto nos obligará a que, el contenido expuesto en el elemento MAIN y su hijo directo ARTICLE deban ser reajustados a la totalidad del ancho de la pantalla.

Una vez que tengamos definida la base de diseño, insertaremos unos textos descriptivos y una imagen a modo de ilustración de la Vía Láctea, que ocupará el cien por cien del ancho disponible, siempre y cuando no supere los 600 píxeles de ancho. Además, se situará centrada horizontalmente y con un filtro de inversión de colores para conseguir una mejor concordancia con el fondo de la página.

Posteriormente, añadiremos algo más de texto y una tabla que contendrá algunos estilos básicos y elementos accesibles. Para ello, estableceremos el atributo SUMMARY, con la descripción corta del contenido de la tabla y, el elemento CAPTION, con el título de la misma. Asimismo, para que las herramientas de asistencia puedan informar adecuadamente sobre el contenido de cada celda, estableceremos la propiedad ID en las celdas de cabecera y la propiedad HEADERS en las de datos.

Para que esta tabla sea receptiva, cuando se presente en una resolución igual o menor a 1280 píxeles, aparecerá una barra de desplazamiento horizontal para facilitar la lectura al usuario. Sin embargo, cuando la resolución del dispositivo sea menor o igual a 800 píxeles, cambiaremos el comportamiento de la tabla para que parezca una lista.

Por último, y justo antes del pie de página, insertaremos unas nuevas secciones de referencias y artículos relacionados. Estas nuevas secciones no usarán ningún elemento UL, como cabría esperar, sino que se realizarán a través de párrafos y enlaces que tendrán definidas unas reglas CSS que les proporcionarán un aspecto de lista.

En lo referente a su diseño, para diferenciar la sección en la que está estableceremos un titular a través de un elemento H2 con degradado horizontal en tonos grises, un redondeado por la derecha de 100 píxeles.

Los elementos de la lista de “Referencias” serán presentados a modo de bloque y, los elementos de la lista de “Artículos relacionados” serán presentados a modo de bloque en línea con aspecto de etiqueta.

# 6

## FORMULARIOS

Un formulario es un conjunto de controles que permiten al usuario interactuar con el documento o página. El objetivo de esta interacción suele ser, a menudo, la solicitud de información adicional o un mero intercambio de datos a petición del usuario.



Entre los diferentes controles que se pueden insertar o agregar a los formularios podemos encontrar acciones directas vinculadas a botones, solicitud de entradas de texto de una única línea, solicitud de entradas de texto multilínea, casillas de verificación, botones de única elección o tipo radio, selección de objetos y/o ficheros, entre otros.

En este capítulo vamos a ver los tipos básicos de formulario, cómo definirlos, cómo enviarlos, cómo hacerlos receptivos y, gran medida, cómo hacerlos usables y accesibles.

## TIPOS DE FORMULARIO

Existen, fundamentalmente, cinco tipos de formulario que están definidos en función de su objetivo. Formularios de contacto, acceso, registro, suscripción y de entrada general.

No obstante, sea cual sea el tipo de formulario y su objetivo, para que cumpla con las expectativas de los clientes y con los requisitos establecidos por el RGPD (Reglamento General de Protección de Datos), deben incluir:

- Una casilla de verificación explícita con la que, los usuarios, puedan aceptar la política de privacidad del sitio. Por defecto, no puede estar seleccionada.
- Los enlaces pertinentes con la política de privacidad, el aviso legal, política de cookies, límite y responsabilidad, etcétera.
- Un texto, no demasiado largo, ni demasiado escueto, sobre qué datos se van a almacenar, quién será el responsable y cuál es el objetivo de dicho almacenamiento.
- Algún método para que el usuario ejerza su derecho a la eliminación de datos.
- Algún algoritmo de cifrado para que la información se gestione y manipule de forma encriptada. Esto es, que los datos que se envíen y/o almacenen estén codificados para evitar usos fraudulentos.

Únicamente se deben solicitar los campos que sean necesarios y estén bien justificados, es decir, no se debe solicitar una información concreta, como pueda ser tus hobbies, a no ser que se tenga una buena razón y esté justificada. Además, se debe tratar de conseguir que el medio sea lo suficientemente seguro y confiable, como para que invite a introducir los datos solicitados.

### Formularios de contacto

Los formularios de contacto pueden llegar a ser un elemento clave en un sitio web porque permiten, o hacen posible, que la comunicación entre los usuarios y los proveedores sea directa y privada.

En general, un formulario de contacto debe solicitar sólo la información esencial para el proceso de comunicación. Esto es, no se deben pedir datos como la razón social, sitio web o teléfono sólo por fines comerciales o estadísticos.

Por ejemplo, el siguiente formulario podría ser considerado no fiable, además de ser ilegal.

## CONTACTAR CON NOSOTROS

**NOMBRE****EMAIL****TELÉFONO****SITIO WEB****MENSAJE****ENVIAR MENSAJE**

Como se puede apreciar en la ilustración anterior, el sitio web no es un dato necesario y, aunque sea un dato opcional, puede suscitar desconfianza e impedir que el usuario haga uso de él. Además, como se ha mencionado anteriormente, es ilegal puesto que no presenta el checkbox de Política de Privacidad, entre otras cosas.

A continuación, se muestra el mismo ejemplo, pero corregido.

## CONTACTAR CON NOSOTROS

**NOMBRE****EMAIL****MENSAJE** **Sí, acepto la política de privacidad****ENVIAR MENSAJE**

*Texto explicativo sobre qué datos se almacenan, quién es el responsable, cuál es su objetivo y legitimación y, cómo se pueden eliminar los datos.*

## Formularios de suscripción

Los formularios de suscripción son, esencialmente, lo mismo que los formularios de contacto, pero más sencillos. Su objetivo es recuperar correos electrónicos para luego utilizarlos con fines comerciales o informativos.

En general, un formulario de suscripción sólo requiere de una caja de texto para recuperar el email, no obstante, es habitual pedir también el nombre para dirigirse a él. Por ejemplo, la siguiente ilustración podría ser una buena opción como formulario de suscripción.

## SUSCRIPCIÓN A LA NEWSLETTER

NOMBRE

EMAIL

Sí, acepto la política de privacidad

**SUSCRIBIRSE**

### INFORMACIÓN BÁSICA SOBRE PROTECCIÓN DE DATOS

*Texto explicativo sobre qué datos se almacenan, quién es el responsable, cuál es su objetivo y legitimación y, cómo se pueden eliminar los datos.*

## Formularios de acceso

Los formularios de acceso son, junto con los formularios de contacto, los más recurrentes y utilizados en el mundo web. Su objetivo, como su propio nombre indica, es proporcionar acceso a información, productos o servicios que no están disponibles por vía pública o sin proporcionar una identidad.

En general, un formulario de acceso se caracteriza por tener dos cajas de texto para el nombre de usuario y contraseña, una casilla de verificación o botón de tipo interruptor para mantener la sesión iniciada, un enlace para recuperar la contraseña en supuesto caso de olvido y, evidentemente, un botón para acceder.

Los nombres de usuario se presentan con el texto visible y, a menudo, son el correo electrónico de los usuarios, aunque puede valer cualquier tipo de identificador único como un DNI, código de cliente, nombre de usuario, etcétera.

Las contraseñas suelen presentarse con el texto oculto, con asteriscos o puntos y, a menudo, suelen proporcionar un modo de cambiar a modo visible para ver el texto escrito.

A continuación, se muestra un ejemplo:

FORMULARIO DE ACCESO

Email o NIF	Contraseña
<input type="text"/>	<input type="password"/> 
Mantener la sesión iniciada	<input type="checkbox"/> NO
<u>RECORDAR CONTRASEÑA</u>	<b>REGISTRARSE</b>
<b>ACCEDER</b>	

Si observamos la ilustración anterior, veremos que el campo contraseña tiene un ícono de ojo que funciona como botón para cambiar el modo de presentación de la misma. Si pulsamos una vez se podrán en modo

visible y seremos capaces de leer lo escrito, pero si pulsamos otra vez, volverá a ocultarse, mostrándose como la vemos ahora mismo.

Además, se ha puesto un botón de acceso al formulario de registro. Esta funcionalidad también es habitual, sobre todo, porque un registro también suele ser un acceso, una vez que ha finalizado el proceso.

## Formularios de registro

Los formularios de registro suelen ir de la mano con los de acceso. Si bien, no siempre se ofrece la posibilidad de registrarse online, si es lo más habitual.

En general, un formulario de registro se caracteriza por la presencia de campos propietarios como el nombre o email del usuario, pero también pueden contener elementos considerados información sensible como son la edad, género o país de nacimiento. La decisión de qué campos se deben solicitar al usuario debe estar regida por la lógica de negocio y el RGPD (Reglamento General de Protección de Datos). No obstante, recordemos que, cuantos menos campos se solicite, más cómodo y confiable podrá sentirse el usuario.

Los nombres de usuario para dicho registro son, a menudo, el correo electrónico de los usuarios, aunque puede valer cualquier tipo de identificador único como un DNI o código de cliente.

Las contraseñas suelen seguir un patrón estricto de definición para evitar la fácil recuperación a través de métodos como la fuerza bruta y uso fraudulento.

En general, podríamos decir que un buen patrón de contraseña es aquel que tiene, al menos, un carácter en mayúsculas, un carácter en minúsculas, un dígito o número, un carácter especial como pueda ser el símbolo de almohadilla o interrogación y con una longitud mínima de ocho caracteres.

Sobra decir que no se deben utilizar sustituciones de carácter a número como pueda ser sustituir la letra A por el número cuatro, ni fechas especiales como cumpleaños o aniversarios y, por supuesto, nada de nombres propios. Eso sí, hay que tratar de que sea fácil de recordar y difícil de adivinar.

A continuación, se muestra un ejemplo:

FORMULARIO DE REGISTRO

Nombre de usuario	Email		
Contraseña	<input type="button" value=""/>	Repita contraseña	<input type="button" value=""/>
Nombre Completo			
<input type="checkbox"/> He leído y acepto los <b>Términos y Condiciones y la Política de Privacidad</b> .			
<input type="checkbox"/> No quiero recibir correos electrónicos sobre nuevos productos, ofertas, ...			
<b>ACCEDER</b>			

Cabe destacar que, es bueno que los campos de un formulario de registro se soliciten de forma que el usuario vaya tomando confianza de menos a más. Es decir, se debe tratar de conseguir que los usuarios confíen un

poco más cada vez que se avanza en el proceso de registro. Esto es así porque puede pasar que un usuario no se sienta del todo cómodo si se le solicita primero la edad antes que el nombre de usuario y contraseña.

Por último, sólo hay que destacar una cosa más. Si el formulario de registro requiere de muchos campos, es mejor que sean agrupados por contexto y solicitados en varios pasos o pantallas.

## Formularios de entrada general

Los formularios de entrada general son aquellos formularios que tienen el propósito de recuperar una información concreta y que, por lo general, están pensados para cubrir las necesidades que no están cubiertas en las casuísticas anteriores.

Por ser algo más explícito, un formulario de propósito general podría ser un formulario de inserción de comentarios, de eventos o de inscripción a sorteos, concursos o juegos.

## ELEMENTOS DISPONIBLES EN HTML5

### Elemento form

El elemento FORM especifica que el contenido que se va a representar es un formulario, es decir, una estructura de interacción, formada por uno o varios elementos, que permiten introducir datos y enviarlos al servidor para su procesamiento.

Los formularios pueden albergar muy diversos elementos con diferentes formatos y estructuras, pero los más comunes quizás sean INPUT, TEXTAREA, BUTTON, SELECT, OPTION, OPTGROUP, FIELDSET, LABEL y OUTPUT, los cuales se pasarán a ver a continuación.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>accept-charset</b>	Especifica la codificación de caracteres que se debe usar cuando se realice el envío del formulario. Su valor predeterminado es UNKNOWN, que indica que la codificación de caracteres a utilizar debe ser la misma que la del documento actual. Entre sus valores más frecuentes podemos encontrar la codificación ISO-8859-1 y la codificación UTF-8
<b>action</b>	Especifica el destino dónde enviar los datos, es decir, la dirección a la que se redirigirá cuando se vaya a realizar la acción de enviar los datos de formulario.
<b>autocomplete</b>	Especifica si los elementos de formulario deben ser manipulados y presentados por agente de usuario de manera automática o no. Sus valores son ON y OFF.
<b>enctype</b>	Especifica cómo se deben codificar los datos a enviar. Sus posibles valores son: <ul style="list-style-type: none"><li>APPLICATION/X-WWW-FORM-URLENCODED: para indicar que se codifiquen todos los caracteres antes de enviarlos. Es la opción por defecto.</li><li>MULTIPART/FORM-DATA: para indicar que no se codifiquen los caracteres.</li><li>TEXT-PLAIN: para indicar que sólo los espacios sean codificados como símbolos de suma.</li></ul>
<b>method</b>	Especifica el método de envío por el que se deben enviar los datos.

Los posibles valores son GET y con La diferencia entre uno y otro es que:

- Mientras que el método GET envía los datos del formulario como parámetros de la propia dirección o URL, el método POST los agrega como parte del cuerpo de la solicitud HTTP.
- Mientras que el método GET permite una longitud máxima de 3000 caracteres, el método POST no posee limitaciones de tamaño.
- Mientras que el método GET permite la adición en la barra de direcciones o favoritos, el método POST no.

#### NOTA

No se debe utilizar el método GET cuando se envían datos confidenciales o sensibles, sin embargo, puede ser la mejor opción cuando se trabaja con datos que no requieren de seguridad alguna, como es el caso de las QUERIES STRINGS o cadenas de consulta de Google.

<b>name</b>	Especifica o asigna el nombre del elemento. El atributo NAME puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice CERO el primero y, N-1, el último.
<b>novalidate</b>	Especifica si se deben validar los elementos del formulario o no. Cabe destacar que este atributo tendrá efecto con sólo declararlo, es decir, en cuanto esté presente, e independientemente de su valor, desactivará la validación de todos los elementos del formulario.
<b>target</b>	Especifica a qué contexto o ventana se enviará la información. Entre los posibles valores que puede tomar, los más frecuentes son: <ul style="list-style-type: none"> <li>• _BLANK: para indicar que se abra en una nueva pestaña o contexto,</li> <li>• _SELF: para indicar que se abra en la misma pestaña o contexto,</li> <li>• _PARENT: para indicar que se abra en la pestaña o contexto padre,</li> <li>• _TOP: para que se abra en el primer elemento BODY de la ventana.</li> <li>• [NOMBRE]: para que se abra en el marco identificado con ese nombre.</li> </ul>

#### Ejemplo:

```
<form action="/action_page.php" method="get" name="frm">
<label for="name">Nombre Completo:</label>
<input type="text" id="name" name="name">
<label>
Nombre de usuario
<input type="text" id="username" name="username">
</label>
<label for="password">Contraseña:</label>
<input type="password" id="password" name="password">
<label for="email">Email de contacto:</label>
<input type="email" id="email" name="email">
<label for="phone">Teléfono:</label>
<input type="tel" id="phone" name="phone">
<button type="submit">
Guardar datos
</button>
</form>
```

## Elemento button

El elemento BUTTON especifica que el contenido que se va a representar es una acción. Como veremos, a diferencia del elemento INPUT, el elemento BUTTON puede albergar una gran variedad de contenidos como, por ejemplo, una imagen, un texto o, incluso, otros elementos HTML.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>autofocus</b>	Especifica si se debe tomar el foco automáticamente después de ser renderizado y/o proceso de carga del documento. Sus valores pueden ser TRUE o FALSE.
<b>disabled</b>	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor DISABLED.
<b>form</b>	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo ID del elemento FORM, de lo contrario, no se hará efectivo.
<b>formaction</b>	Especifica el destino dónde enviar los datos, es decir, la dirección a la que se redirigirá cuando se vaya a realizar la acción de enviar los datos de formulario. También es importante destacar que, cuando este atributo está presente, el atributo ACTION del elemento FORM es anulado. No obstante, será una buena herramienta cuando se desee que el formulario que se está definiendo puede ejecutar varias acciones que llevan a objetivos distintos.
<b>formenctype</b>	Especifica cómo se deben codificar los datos a enviar. También es importante destacar que, cuando este atributo está presente, el atributo ENCTYPE del elemento FORM es anulado. No obstante, será una buena herramienta cuando se desee enviar un mismo formulario con diferentes tipos de codificación. Sus posibles valores son los mismos que los definidos en el atributo ENCTYPE del elemento FORM.
<b>formmethod</b>	Especifica el método de envío por el que se deben enviar los datos. También es importante destacar que, cuando este atributo está presente, el atributo METHOD del elemento FORM es anulado. No obstante, será una buena herramienta cuando se desee enviar un mismo formulario con diferentes métodos. Sus posibles valores son los mismos que los definidos en el atributo METHOD del elemento FORM.
<b>formnovalidate</b>	Especifica si se deben validar los elementos del formulario o no. También es importante destacar que, cuando este atributo está presente, el atributo NOVALIDATE del elemento FORM es anulado. No obstante, será una buena herramienta cuando se desee validar el formulario en función de si se ejecuta una u otra acción. Cabe destacar que este atributo tendrá efecto con sólo declararlo, es decir, en cuanto esté presente, e independientemente de su valor, desactivará la validación de todos los elementos del formulario.
<b>formtarget</b>	Especifica a qué contexto o ventana se enviará la información. También es importante destacar que, cuando este atributo está presente, el atributo TARGET del elemento FORM es anulado. No obstante, será una buena herramienta cuando se desee enviar un mismo formulario a diferentes ventanas o contextos. Sus posibles valores son los mismos que los definidos en el atributo TARGET del elemento FORM.
<b>name</b>	Especifica o asigna el nombre del elemento. El atributo NAME puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del

	<p>servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice CERO el primero y, N-1, el último.</p>
<b>type</b>	<p>Especifica el tipo de botón. Sus posibles valores son:</p> <ul style="list-style-type: none"> <li>• SUBMIT: para indicar que se envíen los datos del formulario,</li> <li>• BUTTON: para que permita hacer clic, pero no envíe los datos del formulario,</li> <li>• RESET: para indicar que se restablezcan todos los elementos del formulario a sus valores por defecto o preestablecidos.</li> </ul>
<b>value</b>	Especifica el valor textual del elemento, por lo que no puede contener nada que no sea texto.

### **Ejemplo:**

```
<button type="submit"
formaction="./pages/login-test-2.php"
formmethod="POST"
formenctype="multipart/form-data"
formtarget="_blank">
value="Probar test 2"
</button>
```

## **Elemento datalist**

El elemento DATALIST especifica que el contenido que se va a representar es una lista de opciones predefinidas para un elemento INPUT.

La razón para utilizar este elemento es para proveer de una funcionalidad de autocompletado a los elementos INPUT. Los usuarios podrán ver las diferentes opciones como si de un desplegable se tratase, pero con la opción de ir buscando a través de coincidencias parciales proporcionadas mediante teclado.

Entre los atributos que admite en su configuración, cabe destacar que, el único atributo que necesita para funcionar, es el atributo ID. El valor de este atributo debe coincidir exactamente con el valor del atributo LIST declarado en el elemento INPUT al que está asociado.

### **Ejemplo:**

```
<input list="technologies">
<datalist id="technologies">
<option value="HTML5">
<option value="JavaScript">
<option value="CSS3">
<option value="SVG">
</datalist>
```

## **Elemento fieldset**

El elemento FIELDSET especifica que el contenido que se va a representar es una agrupación de elementos relacionados. En general, y salvo excepciones, todos los agentes de usuario dibujan un cuadro alrededor de este elemento, equivalente a un estilo de borde.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>disabled</b>	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor DISABLED.
<b>form</b>	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo ID del elemento FORM, de lo contrario, no se hará efectivo.
<b>name</b>	Especifica o asigna el nombre del elemento. El atributo NAME puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice CERO el primero y, N-1, el último.

### Ejemplo:

```
<form action="./login.php">
<fieldset>
<legend>Acceso Privado</legend>
<label for="username">Nombre de usuario</label>
<input id="username" name="username" />
<label for="password">Contraseña</label>
<input id="password" name="password" />
<button type="submit">Enviar</button>
</fieldset>
</form>
```

## Elemento input

El elemento INPUT especifica que el contenido que se va a representar es una entrada de datos.

Una de las cualidades más importantes que tiene el elemento INPUT es que tiene una gran variedad de validaciones nativas. Por ejemplo, es posible definir un elemento de entrada que sólo admita números, que sólo admita fechas en un formato específico o que valide las reglas de nombres de los correos electrónicos.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>accept</b>	Especifica los tipos de archivo válidos cuando se utiliza un elemento INPUT de tipo FILE. Todos los posibles valores que puede tomar este atributo pueden encontrarse en IANA Media Types ( <a href="http://www.iana.org/assignments/media-types/">http://www.iana.org/assignments/media-types/</a> ), aunque puede tomar un valor comodín como es AUDIO/*, VIDEO/* o IMAGE/*, que indican que se aceptan todas las extensiones de archivo de cada tipo.
<b>alt</b>	Permite especificar un texto alternativo para las imágenes que son definidas a través de un INPUT tipo IMAGE.
<b>autocomplete</b>	Especifica si los elementos de formulario deben ser manipulados y presentados por agente de usuario de manera automática o no. Sus valores son ON y OFF.
<b>autofocus</b>	Especifica si se debe tomar el foco automáticamente después de ser renderizado y/o proceso de carga del documento.

	Sus valores pueden ser TRUE o FALSE.
<b>checked</b>	Especifica si el elemento está chequeado o no. Cabe destacar que este atributo sólo es válido para los elementos INPUT de tipo RADIO y CHECKBOX y tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento se marcará como chequeado o seleccionado.
<b>disabled</b>	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor DISABLED.
<b>form</b>	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo ID del elemento FORM, de lo contrario, no se hará efectivo.
<b>formaction</b>	Especifica el destino dónde enviar los datos, es decir, la dirección a la que se redirigirá cuando se vaya a realizar la acción de enviar los datos de formulario. También es importante destacar que, cuando este atributo está presente, el atributo ACTION del elemento FORM es anulado. No obstante, será una buena herramienta cuando se desee que el formulario que se está definiendo puede ejecutar varias acciones que llevan a objetivos distintos.
<b>formenctype</b>	Especifica cómo se deben codificar los datos a enviar. También es importante destacar que, cuando este atributo está presente, el atributo ENCTYPE del elemento FORM es anulado. No obstante, será una buena herramienta cuando se desee enviar un mismo formulario con diferentes tipos de codificación. Sus posibles valores son los mismos que los definidos en el atributo ENCTYPE del elemento FORM.
<b>formmethod</b>	Especifica el método de envío por el que se deben enviar los datos. También es importante destacar que, cuando este atributo está presente, el atributo METHOD del elemento FORM es anulado. No obstante, será una buena herramienta cuando se desee enviar un mismo formulario con diferentes métodos. Sus posibles valores son los mismos que los definidos en el atributo METHOD del elemento FORM.
<b>formnovalidate</b>	Especifica si se deben validar los elementos del formulario o no. También es importante destacar que, cuando este atributo está presente, el atributo NOVALIDATE del elemento FORM es anulado. No obstante, será una buena herramienta cuando se desee validar el formulario en función de si se ejecuta una u otra acción. Cabe destacar que este atributo tendrá efecto con sólo declararlo, es decir, en cuanto esté presente, e independientemente de su valor, desactivará la validación de todos los elementos del formulario.
<b>formtarget</b>	Especifica a qué contexto o ventana se enviará la información. También es importante destacar que, cuando este atributo está presente, el atributo TARGET del elemento FORM es anulado. No obstante, será una buena herramienta cuando se desee enviar un mismo formulario a diferentes ventanas o contextos. Sus posibles valores son los mismos que los definidos en el atributo TARGET del elemento FORM.
<b>height</b>	Especifica el alto del elemento en píxeles.
<b>list</b>	Especifica la lista de opciones predefinidas que se debe asociar a un elemento INPUT. Cabe destacar que, para que un elemento INPUT acepte las opciones predefinidas del elemento DATALIST, el valor del atributo ID del DATALIST debe ser el mismo que el valor del atributo LIST del elemento INPUT.
<b>max</b>	Especifica el límite superior del rango de valores aceptado, es decir, el valor máximo que puede aceptar o se puede introducir en el elemento. Aunque este atributo es válido para los elementos INPUT de tipo METER y PROGRESS, no todos los agentes de usuario lo soportan. Internet Explorer 10 es uno de ellos.

<b>maxlength</b>	Especifica la longitud máxima, en caracteres, del valor que puede ser ingresado en el elemento.
<b>min</b>	Especifica el límite inferior del rango de valores aceptado, es decir, el valor mínimo que puede aceptar o se puede introducir en el elemento. Aunque este atributo es válido para los elementos INPUT de tipo METER y PROGRESS, no todos los agentes de usuario lo soportan. Internet Explorer 10 es uno de ellos.
<b>multiple</b>	Especifica que el elemento admite la selección o inserción de más de un valor. Cabe destacar que este atributo sólo es válido para los elementos INPUT de tipo FILE. Además, tendrá efecto con sólo declararlo, es decir, en cuanto esté presente e independientemente de su valor, el elemento permitirá la selección de múltiples valores.
<b>name</b>	Especifica o asigna el nombre del elemento. El atributo NAME puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice CERO el primero y, N-1, el último.
<b>pattern</b>	Especifica la expresión regular con la que se validará la entrada de datos en el elemento. La forma de trabajar con expresiones regulares es similar a la de JavaScript, no obstante, en la web de HTML5 Pattern, ubicada en la dirección <a href="http://html5pattern.com/">http://html5pattern.com/</a> , se pueden encontrar multitud de ejemplos aptos para ser utilizados. Cabe destacar que este atributo sólo es válido para los elementos INPUT de tipo TEXT, DATE, SEARCH, URL, TEL, EMAIL y PASSWORD.
<b>placeholder</b>	Especifica el texto que se debe mostrar como pista de datos válidos cuando el elemento no contenga un valor. Aunque algunos agentes de usuario pueden no aceptar este atributo, en general es una buena idea hacerlo para ayudar a la usabilidad y accesibilidad web. Si va asociado con otro atributo como PATTERN, la pista a proporcionar debe estar en concordancia con la validez de la entrada.
<b>readonly</b>	Especifica que el elemento es de sólo lectura, es decir, que no permite la inserción de nuevos valores ni la modificación de su valor actual. Cabe destacar que este atributo tendrá efecto con sólo declararlo, es decir, en cuanto esté presente, e independientemente de su valor, provocará que se ponga en modo de sólo lectura. No obstante, no quiere decir que no se pueda seleccionar, resaltar o copiar.
<b>required</b>	Especifica que el elemento debe contener valor, aunque este sea un espacio en blanco, es decir, que no puede ser nulo ni vacío.
<b>size</b>	Especifica la ubicación del recurso externo que se desea cargar o mostrar. Cabe destacar que este atributo sólo es válido para los elementos INPUT de tipo IMAGE.
<b>src</b>	Especifica la ubicación del recurso externo que se desea cargar o mostrar. Este atributo sólo es válido para los elementos INPUT de tipo IMAGE.
<b>step</b>	Especifica el intervalo de incremento o decremento sobre el valor actual del elemento, es decir, el número de pasos que se deben sumar o restar al valor actual del elemento. Cabe destacar que este atributo sólo es válido para los elementos INPUT de tipo NUMBER, RANGE, DATE, DATETIME, DATETIME-LOCAL, MONTH, TIME, y WEEK.
<b>type</b>	Especifica el tipo de elemento INPUT. Sus posibles valores son: <ul style="list-style-type: none"> <li>• <b>BUTTON:</b> Para indicar que es un botón.</li> <li>• <b>CHECKBOX:</b> Para indicar que es una casilla de verificación.</li> </ul>

- DATE: Para indicar que es una fecha.
- DATETIME: Para indicar que es una fecha con hora. Sólo está soportado por Safari y Opera.
- DATETIME-LOCAL: Para indicar que es una fecha con hora sin zona horaria. Sólo está soportado por Chrome, Microsoft Edge y Opera.
- EMAIL: Para indicar que es un correo electrónico.
- FILE: Para indicar que es un control para subir archivos al servidor.
- HIDDEN: Para indicar que es un campo oculto.
- IMAGE: Para indicar que es una imagen.
- MONTH: Para indicar que es un mes. Sólo está soportado por Chrome, Microsoft Edge y Opera.
- NUMBER: Para indicar que es un número real.
- PASSWORD: Para indicar que es una contraseña.
- RADIO: Para indicar que es un botón de radio.
- RESET: Para indicar que es un botón de reinicialización que establece los valores por defecto o preestablecidos.
- SEARCH: Para indicar que es un campo de búsqueda.
- TEL: Para indicar que es un teléfono. Actualmente sólo soportado por Safari.
- TEXT: Para indicar que es una caja o campo de texto.
- TIME: Para indicar que es una hora. Actualmente no soportado por Safari.
- URL: Para indicar que es una dirección web o URL.
- WEEK: Para indicar que es un valor de semana. Sólo está soportado por Chrome, Microsoft Edge y Opera.

<b>value</b>	Especifica el valor textual del elemento, por lo que no puede contener nada que no sea texto.
<b>width</b>	Especifica el ancho del elemento en píxeles.

### Ejemplo:

```

<form action="/action_page.php" method="get" name="frm">
<label for="field1">Campo de texto:</label>
<input type="text" id="field1" name="field1">
<label for="field2">Campo sólo números:</label>
<input type="number" id="field2" name="field2">
<label for="field3">Campo fecha:</label>
<input type="date" id="field3" name="field3">
<label for="field4">Campo URL:</label>
<input type="url" id="field4" name="field4">
<label for="field5">Campo para emails:</label>
<input type="email" id="field5" name="field5">
<label for="field6">Campo para teléfonos:</label>
<input type="tel" id="field6" name="field6">
<label for="field7">Campo para imágenes:</label>
<input type="image" id="field7" name="field7">
<label for="field8">Elemento para adjuntar archivos:</label>
<input type="file" id="field8" name="field8">
<label for="field9">Casilla de verificación:</label>
<input type="checkbox" id="field9" name="field9">
<label for="field10">Campo tipo radio:</label>
<input type="radio" id="field10" name="field10">
<label for="field11">Campo oculto:</label>
<input type="hidden" id="field11" name="field11">
<label for="submit">Botón enviar:</label>
<input type="submit" id="submit" name="submit">
<button type="submit">
Guardar datos

```

```
</button>
</form>
```

## Elemento label

El elemento LABEL especifica que el contenido que se va a representar es una etiqueta para un elemento de formulario INPUT, METER, PROGRESS, SELECT o TEXTAREA.

Cuando se utiliza elemento LABEL, la estructura del documento se vuelve más legible y consistente. Además, beneficia tanto a la usabilidad web, como a la accesibilidad web porque los elementos pequeños como son las casillas de verificación pueden manipularse a través del elemento LABEL.

Además, al tener el control de formulario una etiqueta asociada, las herramientas de asistencia, como los lectores de pantalla, pueden leerla y comunicárselo a los usuarios con discapacidad visual total o parcial.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>for</b>	Especifica el ID del elemento de formulario con el que está vinculada la etiqueta. El contenido al que se hace referencia a través de este atributo funcionará como texto descriptivo de la entrada de datos.
<b>form</b>	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo ID del elemento FORM, de lo contrario, no se hará efectivo.

### Ejemplo:

```
<form action="/action_page.php" method="get" name="frm">
<label for="name">Nombre:</label>
<input type="text" id="name" name="name">
<!-- ... -->
</form>
```

## Elemento legend

El elemento LEGEND especifica que el contenido que se va a representar es un título para un elemento FIELDSET.

### Ejemplo:

```
<form action=".//login.php">
<fieldset>
<legend>Acceso a ejemplo.com</legend>
<!-- ... -->
</fieldset>
</form>
```

Aunque el uso del elemento LEGEND no está recomendado fuera del elemento FIELDSET, se puede aplicar siempre que se deseé. No obstante, a efectos de semántica web, sólo tendrá efecto cuando se encuentre dentro de un elemento FIELDSET.

## Elemento meter

El elemento METER especifica que el contenido que se va a representar es una medición escalar, es decir, como una barra de progreso con rango conocido o valor fraccional.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>form</b>	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo ID del elemento FORM, de lo contrario, no se hará efectivo.
<b>high</b>	Especifica el valor a partir del cual se considera que es un valor alto, siendo este, igual o menor que el valor del atributo MAX.
<b>low</b>	Especifica el valor a partir del cual se considera que es un valor bajo, siendo este, igual o menor que el valor del atributo MIN.
<b>max</b>	Especifica el límite superior del rango de valores aceptado, es decir, el valor máximo que puede aceptar o se puede introducir en el elemento.
<b>min</b>	Especifica el límite inferior del rango de valores aceptado, es decir, el valor mínimo que puede aceptar o se puede introducir en el elemento.
<b>optimum</b>	Especifica el valor a partir del cual se considera que es un valor óptimo, siendo este, mayor que el valor del atributo MIN y menor que el atributo MAX.
<b>value</b>	Especifica el valor numérico inicial del elemento.

### Ejemplo:

```
<form action="/action_page.php" method="get" name="frm">
<label for="available-space">Espacio disponible en disco:</label>
<meter id="available-space" value="324" min="0" max="1024">
324MB libres de 1024 MB
</meter>
<!-- ... -->
</form>
```

Cabe destacar que, aunque pueden parecerse, una medición escalar no es una barra de progreso. Por esta razón, el elemento METER no debe utilizarse para mostrar valores de progreso. El correcto uso de este elemento es para uso o capacidad de disco, para indicar el valor de tareas que tuvieron éxito en un conjunto definido o situaciones similares.

## Elemento optgroup

El elemento OPTGROUP especifica que el contenido que se va a representar es un grupo de opciones relacionadas de un elemento SELECT.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>disabled</b>	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor DISABLED.
<b>label</b>	Especifica la etiqueta que se mostrará para diferenciar el grupo de opciones relacionadas. Este atributo sólo es aplicable para el elemento OPTGROUP, descendiente directo del elemento SELECT.

### Ejemplo:

```

<form action="/action_page.php" method="get" name="frm">
<label for="motorcycles">Motos:</label>
<select id="motorcycles">
<optgroup label="Harley Davidson">
<option value="o1">Sportster</option>
<option value="o2">Softail</option>
<option value="o3">Touring</option>
</optgroup>
<optgroup label="Indian">
<option value="o4">Chief</option>
<option value="o5">Springfield</option>
<option value="o6">Scout Sixty</option>
</optgroup>
</select>
<!-- ... -->
</form>

```

## Elemento option

El elemento OPTION especifica que el contenido que se va a representar es una opción para elemento SELECT.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>disabled</b>	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor DISABLED.
<b>label</b>	Especifica la etiqueta que se mostrará para identificar o etiquetar la opción. Cuando este atributo está presente, su valor será mostrado en vez de su contenido interno. No obstante, no todos los agentes de usuario proporcionan soporte nativo a este atributo.
<b>selected</b>	Especifica si la opción debe marcarse como seleccionada.
<b>value</b>	Especifica el valor textual del elemento, por lo que no puede contener nada que no sea texto.

### Ejemplo:

```

<form action="/action_page.php" method="get" name="frm">
<label for="rate">Calificación:</label>
<select id="rate">
<option value="o1">Mala</option>
<option value="o2">Media</option>
<option value="o3">Buena</option>
</select>
<!-- ... -->
</form>

```

## Elemento output

El elemento OUTPUT especifica que el contenido que se va a representar es el resultado de una operación. Esto puede ser una buena opción a implementar cuando se trata de mostrar el resultado de una operación en donde el usuario introduce varios valores a través de elementos INPUT.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>for</b>	Especifica los ID de los elementos de formulario con los que se operará para mostrar el resultado en el elemento OUTPUT.
<b>form</b>	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo ID del elemento FORM, de lo contrario, no se hará efectivo.
<b>name</b>	Especifica o asigna el nombre del elemento. El atributo NAME puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice CERO el primero y, N-1, el último.

### Ejemplo:

```
<form oninput="res.value=parseInt(op1.value) + parseInt(op2.value)">
<label>Suma de dos operandos</label>
<input type="range" id="op1" value="50">100
+
<input type="range" id="op2" value="50">
=
<output name="res" for="op1 op2">100</output>
</form>
```

## Elemento progress

El elemento PROGRESS especifica que el contenido que se va a representar es una barra de progreso.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>max</b>	Especifica el límite superior del rango de valores aceptado, es decir, el valor máximo que puede aceptar o se puede introducir en el elemento.
<b>value</b>	Especifica el valor numérico inicial del elemento.

### Ejemplo:

```
<form oninput="res.value=parseInt(op1.value) + parseInt(op2.value)">
<label>Progreso de instalación</label>
<progress id="progress" value="54" max="100"> 54% </progress>
</form>
```

Cabe destacar que, aunque puedan parecerse, una barra de progreso no es una medición escalar. Por esta razón, el elemento PROGRESS no debe utilizarse para mostrar valores indicadores. El correcto uso de este elemento es únicamente para mostrar cómo progresa, o ha progresado, una tarea o proceso.

## Elemento select

El elemento SELECT especifica que el contenido que se va a representar es un desplegable con una lista de opciones predefinida.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>autofocus</b>	Especifica si se debe tomar el foco automáticamente después de ser renderizado y/o proceso de carga del documento. Sus valores pueden ser TRUE o FALSE.
<b>disabled</b>	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor DISABLED.
<b>form</b>	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo ID del elemento FORM, de lo contrario, no se hará efectivo.
<b>multiple</b>	Especifica que el elemento admite la selección o inserción de más de un valor. Cabe destacar que este atributo tendrá efecto con sólo declararlo, es decir, en cuanto esté presente e independientemente de su valor, el elemento permitirá la selección de múltiples valores.
<b>name</b>	Especifica o asigna el nombre del elemento. El atributo NAME puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice CERO el primero y, N-1, el último.
<b>required</b>	Especifica que el elemento debe contener valor, aunque este sea un espacio en blanco, es decir, que no puede ser nulo ni vacío.
<b>size</b>	Especifica el ancho en caracteres que debe tener el elemento.

### Ejemplo:

```
<form action="/action_page.php" method="get" name="frm">
<label for="rate">Calificación:</label>
<select id="rate">
<option value="01">Mala</option>
<option value="02">Media</option>
<option value="03">Buena</option>
</select>
<!-- ... -->
</form>
```

## Elemento textarea

El elemento TEXTAREA especifica que el contenido que se va a representar es una caja de texto con la opción de multilínea, es decir, un control de entrada de datos de múltiples líneas.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>autofocus</b>	Especifica si se debe tomar el foco automáticamente después de ser renderizado y/o proceso de carga del documento. Sus valores pueden ser TRUE o FALSE.
<b>cols</b>	Especifica la anchura, en caracteres, del elemento.

<b>disabled</b>	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor DISABLED.
<b>form</b>	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo ID del elemento FORM, de lo contrario, no se hará efectivo.
<b>maxlength</b>	Especifica la longitud máxima, en caracteres, del valor que puede ser ingresado en el elemento.
<b>name</b>	Especifica o asigna el nombre del elemento. El atributo NAME puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice CERO el primero y, N-1, el último.
<b>placeholder</b>	Especifica el texto que se debe mostrar como pista de datos válidos cuando el elemento no contenga un valor. Aunque algunos agentes de usuario pueden no aceptar este atributo, en general es una buena idea hacerlo para ayudar a la usabilidad y accesibilidad web. Si va asociado con otro atributo como PATTERN, la pista a proporcionar debe estar en concordancia con la validez de la entrada.
<b>readonly</b>	Especifica que el elemento es de sólo lectura, es decir, que no permite la inserción de nuevos valores ni la modificación de su valor actual. Cabe destacar que este atributo tendrá efecto con sólo declararlo, es decir, en cuanto esté presente, e independientemente de su valor, provocará que se ponga en modo de sólo lectura. No obstante, no quiere decir que no se pueda seleccionar, resaltar o copiar.
<b>required</b>	Especifica que el elemento debe contener valor, aunque este sea un espacio en blanco, es decir, que no puede ser nulo ni vacío.
<b>rows</b>	Especifica la altura del elemento en líneas de caracteres.
<b>wrap</b>	Especifica si el texto debe incluir o no la definición de nuevas líneas cuando se realiza el envío del formulario para que se ajuste al ancho del elemento establecido. Sus posibles valores son HARD y SOFT, los cuales indican que se agreguen nuevas líneas al texto enviado o no, respectivamente. El carácter de nueva línea añadido al texto será en base al valor del atributo COLS, es decir, cuando el texto llegue a la longitud establecida por COLS, agregará o no un nuevo salto de línea en función de si el atributo WRAP está o no establecido a HARD. El uso de este atributo no se recomienda si se desea mantener un buen nivel de usabilidad y accesibilidad web puesto que puede provocar pérdidas en la legibilidad de los datos si no se utiliza adecuadamente.

## Ejemplo:

```
<form action="/action_page.php" method="post" name="frm">
<label>
Descripción
<textarea id="desc" rows="24" cols="80"></textarea>
</label>
<!-- ... -->
</form>
```

## ELEMENTOS DISPONIBLES EN CSS

CSS no presenta ninguna propiedad específica para elementos de formulario, sin embargo, cabe destacar que estos elementos pueden aprovecharse de las mismas ventajas que cualquier otro elemento de bloque o caja. Es decir, pueden ser personalizados con propiedades de todo tipo, como son FONT-FAMILY, TEXT-ALIGN, POSITION, DISPLAY, WIDTH, HEIGHT, PADDING, MARGIN, BORDER, BACKGROUND, ...

No obstante, entre todas ellas, cabe destacar dos:

### Propiedad box-sizing

Especifica cómo deben asignarse y calcularse el alto y ancho de los elementos. Esto es, si deben incluir los márgenes internos (padding) y/o los bordes, o no. Entre sus posibles valores podemos encontrar:

- **CONTENT-BOX:** Indica que se debe incluir sólo el contenido, e ignorar los márgenes internos y bordes. Es el valor por defecto.
- **BORDER-BOX:** Indica que se deben incluir el contenido, padding y bordes.

#### NOTA

En general, se puede afirmar que, trabajar con cajas o capas incluyendo los márgenes internos y los bordes es más fácil de manejar y facilita el diseño adaptativo, aunque no siempre.

### Propiedad resize

Especifica si los elemento TEXTAREA deben permitir la manipulación del tamaño del elemento desde la interfaz que presenta el agente de usuario. Sus posibles valores son:

- **NONE:** Indica que el usuario no puede cambiar el tamaño del elemento.
- **HORIZONTAL:** Indica que el usuario puede cambiar, únicamente, el tamaño del elemento horizontalmente.
- **VERTICAL:** Indica que el usuario puede cambiar, únicamente, el tamaño del elemento verticalmente.
- **BOTH:** Indica que el usuario puede cambiar el tamaño del elemento en ambas direcciones.

#### NOTA

Aunque esta propiedad puede resultar muy útil, su soporte está muy limitado. De hecho, no está soportado por Internet Explorer, Microsoft Edge 78 o inferior, ni Opera 14 o inferior.

### Ejemplo:

```
textarea { resize: none; }
```

## VALIDACIÓN DE FORMULARIOS

La validación de formularios se realiza a través de JavaScript y, hasta no hace tanto, no era casi personalizable ni eficiente. Ahora, sin embargo, gracias a la amplia gama de propiedades que poseen los elementos de formulario, junto con los métodos de notificación y manipulación que nos provee HTML5, podemos realizar validaciones de forma bastante rápida y sencilla.

## La interfaz `ValidityState`

La interfaz `VALIDITYSTATE` es un objeto que representa todos los posibles estados por los que puede pasar un elemento de formulario. Además, suele indicar la razón o el motivo por el que se encuentra en ese estado.

Entre sus propiedades más frecuentes podemos encontrar:

Propiedad	Descripción
<b>badInput</b>	Esta propiedad devuelve true si ha habido algún problema con la conversión del dato introducido.
<b>customError</b>	Esta propiedad devuelve TRUE si el elemento contiene asignado un error definido por el usuario establecido a través del método <code>SETCUSTOMVALIDITY</code> .
<b>patternMismatch</b>	Esta propiedad devuelve true si el elemento no cumple el patrón definido. Si su valor es true, la pseudo-clase <code>:INVALID</code> de CSS también se activará.
<b>rangeOverflow</b>	Esta propiedad devuelve true si el elemento contiene un valor mayor al provisto por la propiedad MAX. Si su valor es TRUE, las pseudo-clases <code>:INVALID</code> y <code>:OUT-OF-RANGE</code> de CSS también se activarán.
<b>rangeUnderflow</b>	Esta propiedad devuelve true si el elemento contiene un valor menor al provisto por la propiedad MIN. Si su valor es TRUE, las pseudo-clases <code>:INVALID</code> y <code>:OUT-OF-RANGE</code> de CSS también se activarán.
<b>stepMismatch</b>	Esta propiedad devuelve TRUE si el elemento contiene un valor que no concuerda con el paso provisto por la propiedad STEP. Si su valor es TRUE, las pseudo-clases <code>:INVALID</code> y <code>:OUT-OF-RANGE</code> de CSS también se activarán.
<b>tooLong</b>	Esta propiedad devuelve TRUE si el elemento tiene una longitud mayor que la provista por el atributo <code>MAXLENGTH</code> . Si su valor es TRUE, las pseudo-clases <code>:INVALID</code> y <code>:OUT-OF-RANGE</code> de CSS también se activarán.
<b>tooShort</b>	Esta propiedad devuelve TRUE si el elemento tiene una longitud menor a la provista por el atributo <code>MINLENGTH</code> . Si su valor es TRUE, las pseudo-clases <code>:INVALID</code> y <code>:OUT-OF-RANGE</code> de CSS también se activarán.
<b>typeMismatch</b>	Esta propiedad devuelve TRUE si el elemento contiene una sintaxis incorrecta. Sólo es válido para los tipos de INPUT EMAIL y URL. Si su valor es TRUE, la pseudo-clase <code>:INVALID</code> de CSS también se activará.
<b>valid</b>	Esta propiedad devuelve TRUE si el elemento cumple todas las restricciones requeridas. Si su valor es TRUE, la pseudo-clase <code>:VALID</code> de CSS también se activará.
<b>valueMissing</b>	Esta propiedad devuelve TRUE si el elemento es requerido y se encuentra vacío. Si su valor es TRUE, la pseudo-clase <code>:INVALID</code> de CSS también se activará.

## Propiedades y métodos

A continuación, se muestran los principales métodos y propiedades que pueden ser utilizados en la validación de formularios.

### PROPIEDAD `VALIDITY`

Esta propiedad resulta ser un objeto que devuelve un conjunto de datos de tipo `VALIDITYSTATE` y permite conocer el resultado de todos los posibles problemas que se han producido tras realizar una comprobación de validación.

## MÉTODO SETCUSTOMVALIDITY

El método SETCUSTOMVALIDITY permite definir mensajes de error personalizados en los elementos de formulario. El mensaje, proporcionado como parámetro, es guardado en la propiedad VALIDATIONMESSAGE.

## PROPIEDAD VALIDATIONMESSAGE

Esta propiedad contiene el mensaje generado tras el proceso de validación. Si el elemento no ha sido validado aún o ha pasado con éxito todo el proceso de validación, el contenido de esta propiedad estará establecida a cadena vacía. Si, por el contrario, existe algún error o problema con la validación, esta propiedad mostrará el mensaje de error o información sobre el problema.

## MÉTODO CHECKVALIDITY

Este método comprueba si se cumplen las restricciones que tiene definido el elemento de formulario. Si todo es correcto, es decir, que pasa la validación, devolverá TRUE, en cualquier otro caso, devolverá FALSE.

## Eventos

A continuación, se muestran los principales eventos que pueden ser utilizados en la validación de formularios.

### EVENTO INVALID

Cada vez que se solicita la acción de enviar un formulario, se realiza una acción de validación previamente. Si el proceso de validación no tuvo éxito, el navegador lanza una especie de excepción que marca al elemento como inválido y le asigna la pseudo-clase de CSS :INVALID.

Pues bien, si además de controlar la validación interna queremos o necesitamos gestionarla de forma externa, para esto, tenemos el evento INVALID.

El evento INVALID es lanzado cuando el elemento realiza el proceso de validación y no cumple alguna de sus restricciones. Una vez, dentro de este evento podemos, por ejemplo, mostrar los mensajes personalizados que hemos creado para nuestra aplicación en el lugar de los predefinidos.

## EJEMPLO DE VALIDACIÓN

Imaginemos que deseamos comprobar que el valor introducido en un INPUT denominado STATUS concuerde con uno de los tres posibles valores: "Asignado", "En Progreso" o "Finalizado".

Para ello, lo primero que necesitaremos es declarar el código HTML con un LABEL, un campo de entrada de texto y dos elementos adicionales para poder establecer los posibles mensajes de error.

```
<h1>Prueba de validación</h1>
<form name="frm" enctype="multipart/form-data" method="get">
<label>
Estado:
<input id="status" type="text" oninput="check(this)" />
</label>
<h2>Mensaje tras validación</h2>
<div id="validateMsg"></div>
<h2>Listado de errores de validity</h2>
<div id="validity"></div>
```

```
</form>
```

Una vez insertado el HTML, necesitamos declarar la funcionalidad de JavaScript que realizará la validación. Para ello, podríamos hacer algo como:

```
function check(input) {
    // Recuperamos el valor introducido y los elementos a manipular
    var val = input.value;
    var vm = document.getElementById("validateMsg");
    var va = document.getElementById("validity");
    // Comprobamos si el valor es válido
    if (input.value != "Asignado" &&
        input.value != "En Progreso" &&
        input.value != "Finalizado") {
        // Como no es válido, establecemos un mensaje de error entendible
        // y se lo asignamos a la función setCustomValidity
        var msg = "" + val + " no es un estado.";
        input.setCustomValidity(msg);
        // Asignamos el mensaje de error a nuestro objeto validityMsg
        vm.innerHTML = input.validationMessage;
        va.innerHTML = "";
        // Mostramos el listado de las restricciones incumplidas
        for(var key in input.validity){
            var status = input.validity[key];
            if(status){
                va.innerHTML += key + ": " + status.toString();
                va.innerHTML += "<br/>";
            }
        }
    } else {
        // Todo correcto.
        // Eliminamos el mensaje y el listado.
        input.setCustomValidity("");
        vm.innerHTML = "";
        va.innerHTML = "";
    }
    // Recuperamos el elemento con ID status y le asignamos un evento para
    // comprobar su validez
    var s = document.getElementById("status");
    s.addEventListener("invalid", check, true);
}
```

## USABILIDAD Y ACCESIBILIDAD EN LOS FORMULARIOS

Siempre que se pueda, hay que agrupar la información que se solicita al usuario y ordenarla por relevancia para evitar que se sienta incómodo con la información que se le está solicitando.

Un ejemplo gráfico de esto podría ser es la solicitud de direcciones en donde, por lo general, se solicitan los datos con el siguiente orden: Tipo de Vía, Dirección, Número, Código Postal, Localidad y Provincia.

También es bueno tener presente la sensibilidad de los datos y establecer un orden de petición adecuado. Por ejemplo, en los formularios de registro se suele pedir primero el nombre de usuario, después el email y luego, más tarde, sus datos personales como el nombre completo del usuario. Es una forma de irle guiando, poco a poco, y evitar el abandono por preguntas como ¿por qué me solicitan esto ahora?.

Otra norma que se suele establecer a la hora de solicitar información en los formularios es solicitar primero los campos que sean requeridos y establecer la solicitud de información por pasos, para evitar que los usuarios puedan abrumarse o frustrarse y abandonar.

## **Autocompletado**

La opción de autocompletado es como si se añadiese una capa predictiva sobre el campo. Los campos de los formularios suelen tener la opción de autocompletar para evitar reescribir los valores que ya se habían escrito con anterioridad. Esta opción no se recomienda desactivarla.

El autocompletado también suele referirse a la acción de llenado causado por otra acción. Una buena praxis es que, si hay campos que pueden ser llenados de forma automática, se haga. Un ejemplo de ello es el autocompletado de la provincia ya que puede extraerse a partir de la IP del visitante o a través del código postal.

## **Estructuración y optimización**

Una de las prácticas más habituales en el diseño de formularios es que se construyan en formato tabla con varias columnas. Esto puede ser una buena praxis salvo cuando se muestran en un dispositivo móvil ya que, una sobrecarga de información puede hacer que se vuelva ilegible.

En dispositivos de escritorio, no se deben establecer más de tres campos, con sus respectivas etiquetas, por fila, es decir, no debe haber más de seis columnas entre etiquetas y campos.

En dispositivos móviles, con resoluciones menores a 640 píxeles de ancho, los campos deberán organizarse en formato de una única columna, a no ser que sean de un tamaño tal que permita la representación clara sin pérdida de legibilidad.

Otro factor importante es que esté bien armado. Esto se vuelve especialmente importante en dispositivos móviles ya que el espacio de visión está muy reducido.

## **Capacidades de los dispositivos**

Siempre que se pueda, hay que utilizar las capacidades que proporcionan los dispositivos y estudiar si van a ser una mejora o un impedimento. De hecho, hay algunas características que ya se sabe que experiencia de usuario proporcionan.

Por ejemplo, si se está manejando un dispositivo móvil:

- Se debe evitar en la medida de lo posible que el usuario tenga que utilizar el teclado móvil para llenar formularios.
- Siempre que se pueda, hay que sustituir los campos de tipo texto por botones de opción, verificación o desplegables.
- Los dispositivos móviles soportan gran número de tipos de datos en los formularios. La elección de un color, una fecha, un rango, o introducción de e-mails se hace mucho más sencillo cuando se utilizan los tipos de datos predefinidos.

## **Combos o desplegables**

El único caso dónde el uso de desplegables está justificado es en aquellos casos en los que la respuesta está predefinida y el número de opciones es pequeño. La principal razón de esta afirmación es que, en muchas ocasiones, es más rápido escribir que seleccionar una opción.

El uso de desplegables muy grandes ralentizan la página e incrementa su peso por lo que el tiempo de latencia aumenta y se tarda más tiempo en acceder a los datos.

## **Número de campos**

Está demostrado que el número de campos de los formularios puede crear frustración e inseguridad. Por esta razón, se debe acotar al mínimo posible, es decir, sólo hay que solicitar los campos imprescindibles.

Si el número de campos necesarios es muy extenso se deben dividir en varias páginas (no pestañas) intentando agrupar los campos relacionados y mostrando, primero los requeridos y después los opcionales.

## **Nombres de campo**

Establecer nombres de campos claros y compresibles es fundamental para que el usuario no cometa errores y proporcione la información con más fluidez. No se deben utilizar palabras técnicas, ni hacer preguntas complejas.

Para el diseño de formularios, los campos deben estar incrustados dentro de elementos LABEL o estar asociados a ellos mediante el atributo FOR. La razón de utilizar esta forma es porque, además de estructurar, ayudan a asegurar la accesibilidad web.

Para facilitar la comprensión de lo que se tiene que introducir o seleccionar, el nombre del campo debe establecerse encima del campo salvo cuando el dato solicitado se muestra a través de casillas de verificación o a través de botones de opción única.

## **Tipos de datos estándar**

Los formularios tienen una gran cantidad de tipos de datos distintos. Cada tipo de dato está definido para un fin concreto y, normalmente, para conseguir una mejor experiencia de usuario.

Cuando se diseñan las interfaces para entornos de escritorio, la cantidad de tipos de datos disponibles se ve mermada por la incompatibilidad que existe entre los diferentes agentes de usuario. Los tipos de datos que funcionan en uno pueden no funcionar en otro. Por esta razón, en las interfaces de escritorio se tiende a utilizar el tipo texto para muchas de las necesidades.

Si el diseño es para dispositivos móviles, la cosa cambia. Prácticamente todos los tipos de datos que existen están disponibles en los navegadores móviles. Los campos de tipo fecha, por ejemplo, muestran un calendario que ayuda a introducir los datos casi sin esfuerzo.

Siempre que se vaya a diseñar un formulario se debe estudiar la compatibilidad de los tipos de datos en todos los dispositivos dónde se va a poder utilizar y usarlos con inteligencia.

## Botones y enlaces

Los botones deben tener apariencia de botón y no ser extravagantes. Un diseño no formal puede confundir a los usuarios y provocarles frustración.

Se debe separar los botones por funcionalidad y/o contexto. Un buen ejemplo es establecer las acciones de “guardar” o “aceptar” en un extremo de la pantalla y las acciones de “volver” o “cancelar” en el otro. Esto evitará que los usuarios cometan errores tales como pinchar por “accidente” en una opción que no era la deseada.

Los botones deben tener un tamaño suficientemente grande para tener un mejor acceso y diferenciarse por su estilo y forma del resto de contenidos. Por ejemplo, los botones de acción como “guardar” o “aceptar” deben tener un estilo diferente a los botones de “volver” o “cancelar” y al resto de enlaces.

Los títulos de acciones como “Ver más”, “Seguir leyendo” o “Click aquí” empobrecen la usabilidad y ralentizan el reconocimiento visual de los usuarios ya que pueden perder el contexto de la acción. Además, puede llegar a ser un problema grave de accesibilidad web.

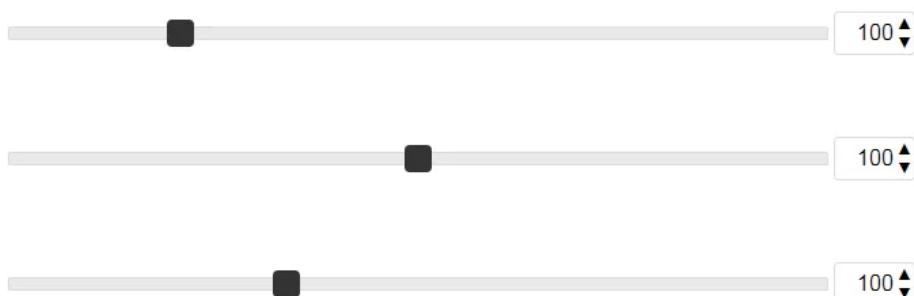
Se debe poder distinguir los diferentes estados de los elementos. Por ejemplo, si se definen estilos diferentes para los estados de un enlace, los usuarios prestarán más atención a los contenidos ya que no perderán el tiempo en realizar reconocimientos visuales para averiguar lo tienen seleccionado o que lo han visitado con anterioridad.

## Rangos de valores

Los rangos de valores se pueden utilizar cuando el valor a recuperar no es relevante para el usuario, como pasa con la edad o cuando se solicitan intervalos de datos cortos fácilmente seleccionables a través de un efecto de arrastrar y soltar, como pueden ser las puntuaciones de un valor porcentual.

Los rangos deben tener un tamaño adecuado para ser utilizados con el pulgar si se trata de un dispositivo móvil.

Además, deben tener asociado un campo de entrada de datos de tipo numérico que tenga la misma funcionalidad que un INPUT de tipo NUMBER. Es decir, que no permita introducir letras o símbolos, que pueda ser incrementado con los cursores de arriba y abajo del teclado o pulsando con el ratón en los símbolos de arriba o abajo, que impida que, este campo numérico, admita valores no contemplados por el rango de valores y que ambos, rango y campo, estén sincronizados si se produce un cambio en el valor de cualquiera de los dos.



Ejemplo control de rango de valores estilizados

## PRÁCTICA 7: PÁGINA DE ACCESO DE “UNIVERSES”

### Código del ejemplo

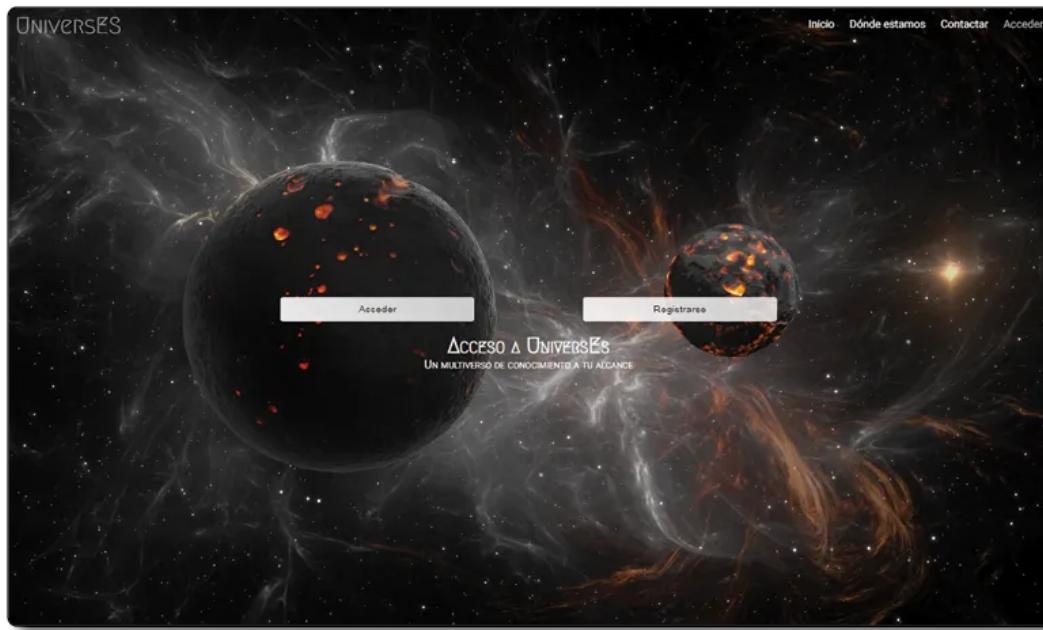
<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-07/pages/login/>



### Objetivo de la práctica

Diseñar un formulario de acceso y registro combinado.

### Resultado



### Recursos para hacer la práctica

#### Recurso

<https://pixabay.com/es/photos/la-astronomia-espacio-luna-galaxy-3199541/>



## *Para qué este recurso*

Imagen de fondo. La versión que se insertará es la que tiene un tamaño de 1920x1080. Su texto alternativo será el “acceso-a-universES”.

En esta página seguiremos un poco la misma línea de diseño que en la página de inicio, es decir, tras la carga aparecerá a pantalla completa sin deformarse y, cuando se muestre uno de los formularios se establecerá a una altura de 250 píxeles.

## **Resolución**

Ya que es el primer contacto con los formularios, lo primero que haremos es una página de acceso a zona privada sencilla. Cabe mencionar que, aunque el HTML y JavaScript utilizados son bastante triviales, el CSS es algo más complicado. No obstante, trataremos de explicar todas las partes con el mayor rigor posible.

Se trata de una página que mostrará una imagen a pantalla completa con un título y los botones de acción de acceso y registro. Cuando el usuario pulse en uno de los dos botones, se reajustará dicha imagen al ancho de la pantalla y mostrará el formulario correspondiente.

El formulario de acceso tendrá los campos nombre de usuario y contraseña, junto con una casilla de verificación para indicar si se debe mantener la sesión iniciada o no. Además, se ofrecerán dos acciones para poder recuperar la contraseña y, por supuesto, poder acceder normalmente.

El formulario de registro tendrá los campos nombre de usuario, contraseña y repetir contraseña (para poder ratificar que la contraseña es correcta), junto con una casilla de verificación para indicar si se está de acuerdo con la Política de Privacidad y Términos de uso o no. Como posibles acciones, sólo se ofrecerá un botón para poder registrarse.

## **EXPLICACIÓN DEL CÓDIGO /PAGES/LOGIN/INDEX.HTML**

En lo referente a la configuración del documento y cabecera no vamos a decir nada puesto que únicamente cambia el título del mismo y la clase ACTIVE en el menú de navegación y, con respecto al cuerpo del documento, como siempre, definiremos un elemento MAIN que contendrá un elemento ARTICLE con una cabecera y un elemento agrupador FIELDSET.

En la cabecera del artículo (elemento ARTICLE > HEADER), como en anteriores ocasiones, definiremos una imagen, un título y un slogan.

En el elemento FIELDSET expuesto a continuación de dicha cabecera, estableceremos los dos formularios de acceso y registro con los campos mencionados anteriormente. Cada campo o elemento de formulario irá dentro de un elemento LABEL, para ayudar a contextualizar los diferentes tipos de contenido y con todos los atributos pertinentes.

Si nos fijamos en el código de dicho elemento, podremos observar que se utilizan clases que nunca se han definido y algunos estilos en línea para ajustar al comportamiento deseado. Esto último (los estilos en línea) deben usarse con cautela puesto que puede llegar a ser una mala praxis. Esto es, sólo se deben establecer estilos en línea cuando sea estrictamente necesario o cuando la declaración de estilo no vaya a perjudicar la herencia del código.

Por último, sólo nos resta copiar el elemento FOOTER de nuestro documento, que resulta ser exactamente idéntico a todos los anteriores e incluir el código JavaScript localizado en la carpeta JS.

## CAMBIOS EN EL ARCHIVO STYLES.CSS

En lo referente al CSS, lo primero que realizaremos es la definición de los elementos de formulario, uno a uno. Para todos los elementos INPUT que no sean de tipo CHECKBOX, se les establecerá un estilo común con fondo blanco y sólo con el borde inferior definido. Además, tendrán una altura de 40 píxeles, un estilo de negrita de peso medio y ocuparán el cien por cien del ancho del contenedor.

A los textos asociados a los campos (elementos SPAN de cada LABEL), se les dotará de un efecto de movimiento cuando estén rellenos o enfocados. En otras palabras, cuando el usuario tenga enfocado el elemento INPUT, o haya introducido un valor, el texto proporcionado por los SPAN, se quedará encima del valor que introduzca el usuario. En caso de que esté vacío o desenfocado, aparecerá como si fuese un valor de la propiedad PLACEHOLDER.

Los elementos LABEL tendrán un posicionamiento relativo para poder fijar el posicionamiento absoluto de los SPAN, que son los que utilizaremos como sustituto de la propiedad PLACEHOLDER. Recodemos que, la posición (0,0) de un elemento que posee posicionamiento absoluto, coincide con la posición actual del último elemento padre que tiene posicionamiento relativo.

El elemento SPAN que está contenido dentro del LABEL estará situado a 20 píxeles del borde superior del elemento padre y tendrá un efecto de animación realizado a través de propiedad TRANSITION. Esta propiedad, y otras, se verán más adelante en un capítulo posterior, pero lo que importa en este caso es que, el efecto, provocará un movimiento suave que durará 300 milisegundos modificando la propiedad TOP hasta un valor de CERO.

Los elementos INPUT de tipo CHECKBOX, RADIO y FILE son un tipo especial de elemento de formulario porque, entre otras cosas permiten la personalización a través de CSS mediante los pseudo-elementos BEFORE y AFTER.

En términos generales, la personalización de un elemento de tipo CHECKBOX o RADIO se basa en ocultar el estilo predefinido por los agentes de usuario mediante el pseudo-elemento BEFORE y personalizar el símbolo de verificación a través del pseudo-elemento AFTER.

En nuestro ejemplo, cuando el elemento no esté seleccionado, se aplicará un estilo de fondo blanco con borde gris y un redondeado del borde mediante el pseudo-elemento BEFORE únicamente. Sin embargo, cuando el elemento esté seleccionado se modificará el fondo a negro a través de la combinación de la pseudo-clase CHECKED y pseudo-elemento BEFORE.

Algo muy similar sucede con el pseudo-elemento AFTER. Por defecto, se definirá una figura que representa el símbolo de verificación mediante técnicas de CSS y que, esencialmente, trata los bordes y juega con la orientación del elemento a través de la propiedad TRANSFORM. Para hacer que aparezca y desaparezca el símbolo de verificación, cuando el elemento no esté seleccionado se le aplicará la propiedad OPACITY con un valor 0 y, cuando lo esté, se le aplicará la propiedad OPACITY con un valor 1.

Ahora vamos con los botones y enlaces. Como ya se definió un estilo de botón en la página de inicio, reutilizaremos parte de ese código. Si nos fijamos en el diseño propuesto para esta página, los botones de

ACCEDER y REGISTRARSE no parecen similares a los propuestos en la página de inicio, sin embargo, sólo cambia el color de fondo, si exceptuamos unos pocos ajustes adicionales aplicables al caso actual.

En lo referente a los botones con aspecto de enlace y enlaces en sí, lo único que haremos es definir unos estilos básicos para que no pierda la concordancia con el resto de los elementos sin perder el significado de lo que son. Recordemos que un enlace debe parecer un enlace porque, en caso contrario, los usuarios pueden no saber qué es y frustrarse.

Por fin hemos llegado a la definición de estilos propia de la página que queremos diseñar. En este caso, lo primero que haremos es definir el fondo del documento a negro y establecer los elementos MAIN y ARTICLE al cien por cien del ancho de la pantalla.

En la cabecera de ese elemento ARTICLE configuraremos una imagen para que se ajuste al tamaño de la pantalla, pero cuando el usuario pulse en uno de los botones, se reajuste a una altura de 250 píxeles, sin deformarse. Dichos botones aparecerán centrados en la pantalla y se ocultarán cuando se muestre uno de los formularios asociados.

Al elemento FIELDSET que contiene los formularios, le definiremos un fondo blanco con un borde transparente de 10 píxeles que, por defecto, aparecerá oculto, gracias, en parte, a que la propiedad MAX-HEIGHT contendrá el valor CERO.

Si nos fijamos en el código de dicho elemento, veremos que se hace referencia a unas clases denominadas SHOWING-SIGNIN y SHOWING-SIGNUP que indican cuál es el formulario es el que se desea mostrar y se manipulan mediante una pequeña función en JavaScript que es invocada por el evento ONCLICK de los botones de acción.

Esa función de JavaScript está definida en el archivo SCRIPTS.JS y, básicamente, lo que hace es establecer una clase en el elemento BODY para que se muestre uno u otro formulario a través de CSS.

## **PRÁCTICA 8: EXPERIMENTO DE PÁGINA DE CONTACTO DE “UNIVERES”**

### **Código del ejemplo**

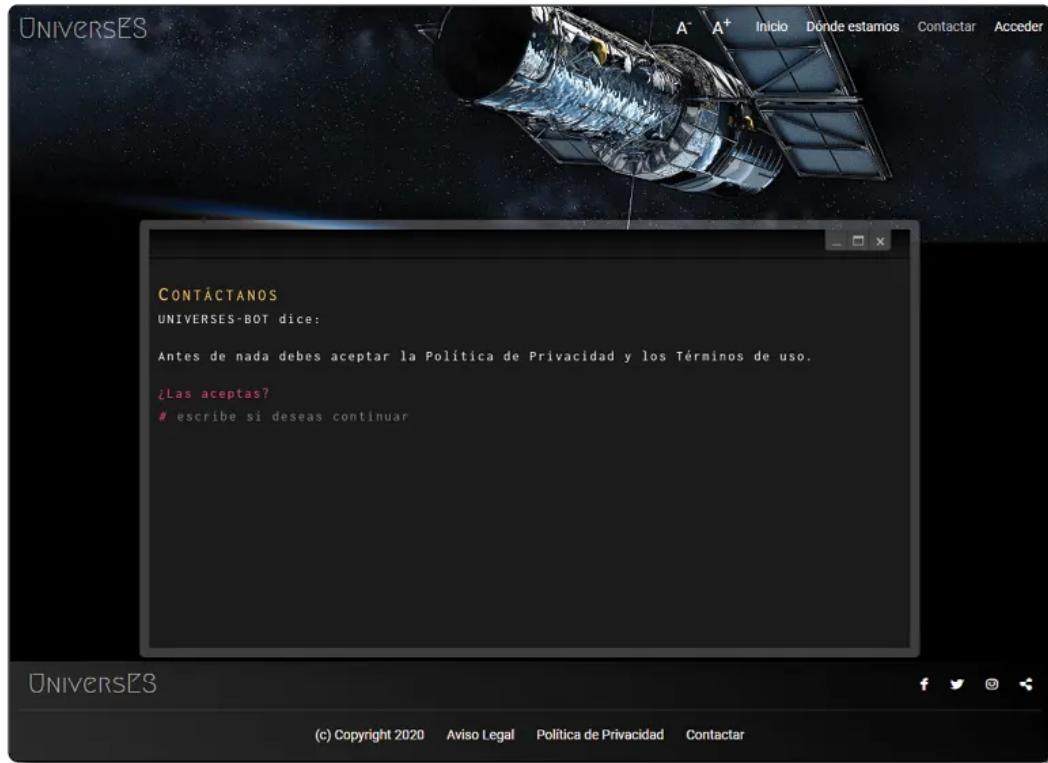
<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-08/pages/contactar/>



### **Objetivo de la práctica**

Diseñar un formulario de contacto experimental.

## Resultado



## Recursos para hacer la práctica

### Recurso

<https://pixabay.com/es/illustrations/telescopio-hubble-universo-1347645/>



### Para qué este recurso

Esta es la dirección de la imagen para la imagen de cabecera. La versión que se insertará es la que tiene un tamaño de 1920x1080. Su texto alternativo será el “Telescopio-Hubble-cabecera”.

En esta página seguiremos un poco la misma línea de diseño que en la página de “Dónde estamos”, es decir, la imagen tendrá una altura de 250 píxeles, pero ocupando el cien por cien del ancho de la pantalla sin deformarse.

## Resolución

La idea de esta práctica es simular una aplicación parecida a un símbolo de sistema o línea de comandos que ofrecían los terminales de los años 80, aunque algo más actual. Para conseguir este efecto, se diseñará un formulario que irá lanzando preguntas cada vez que se pulse la tecla de retorno de carro.

Primero se solicitará si acepta la política de privacidad y términos de uso. Despues, el nombre y el email para realizar el contacto y, finalmente, un campo de texto para que escriba el mensaje.

## **EXPLICACIÓN DEL CÓDIGO /PAGES/CONTACTAR/INDEX.HTML**

Al igual que sucede con la práctica anterior, en lo referente a la configuración del documento y cabecera no vamos a decir nada puesto que, únicamente, cambia el título del mismo y la clase ACTIVE en el menú de navegación.

Con respecto al cuerpo del documento, como siempre, definiremos un elemento MAIN que contendrá un elemento ARTICLE con una cabecera y un elemento agrupador FIELDSET. En la cabecera del artículo, como en anteriores ocasiones, definiremos una imagen, sin embargo, en esta ocasión no se mostrará ningún título ni slogan.

Lo que sí que cambiará, y bastante, será el contenido del elemento FIELDSET, el cual contendrá un elemento LEGEND con un texto y tres botones y, un elemento FORM con los diferentes elementos necesarios.

Dentro de ese elemento LEGEND colocaremos un elemento de título H1 y un elemento de párrafo como entradillas a la petición de datos. Seguidamente, mostraremos un texto de aviso indicando que, si no se aceptan la política de privacidad y los términos de uso, no se podrá continuar.

Una vez hayamos definido los textos introductorios, se deberán establecer una serie de bloques o sentencias que conformarán cada una de las distintas entradas de datos. Cada uno de estos bloques contendrá los siguientes elementos:

- Un elemento LABEL con un atributo FOR que identificará a qué INPUT está asociado.
- Un elemento I que servirá como decorador de línea de comandos.
- El elemento INPUT que será utilizado para la recogida de datos
- Un elemento SPAN que servirá como contenedor de mensajes de error.

Además, cada uno de los elementos INPUT tendrá una serie de atributos que lo especificarán su cometido y nos ayudarán a validar la entrada de datos. En general, cada uno de ellos podrá contener:

- Un atributo ID para identificar el elemento de formulario y asociarlo a la etiqueta LABEL.
- Un atributo NAME para que pueda ser enviado y recibido por el servidor.
- Un atributo REQUIRED para indicar que es obligatorio y validable.
- Un atributo PLACEHOLDER para ayudar al usuario a contextualizar la entrada de datos y facilitar su comprensión, lo que evitará errores involuntarios o de entendimiento.
- Un atributo ONKEYDOWN para controlar cuando se pulsa la tecla ENTER o retorno de carro.
- Un atributo MINLENGTH para indicar la longitud mínima de caracteres.
- Un atributo MAXLENGTH para indicar la longitud máxima de caracteres.
- Un atributo PATTERN que se utilizará para validar que se insertan los valores correctos.
- El atributo AUTOFOCUS para indicar que se establezca el cursor en ese elemento tras la carga de la página.

Como se podrá apreciar en el código del formulario, se define una estructura cada vez que se desea solicitar una información y se caracteriza porque tiene asignada la clase SENTENCE. Algunas de estas estructuras presentan la clase NEXT que, además de ser útiles a la hora manipular los distintos bloques en JavaScript, identificarán los siguientes pasos por los que se pasará un usuario cuando entre en la página y permanecerán ocultos hasta el momento de utilizarlos.

Seguidamente, copiamos el elemento FOOTER de documentos o prácticas anteriores y el código JavaScript genérico. En este punto, y sólo para este documento, añadiremos una etiqueta SCRIPT que contendrá el JavaScript necesario para su correcto funcionamiento.

## CÓDIGO ARCHIVO /JS/CONTACT.JS

Una vez que hemos declarado el HTML de la página, es hora de declarar el código JavaScript necesario para que se produzca una buena experiencia cuando el usuario interactúe con la página.

Primero declararemos una función denominada WHENKEYDOWN que controlará la tecla de retorno de carro o aceptar y que, en JavaScript, se corresponde con el código 13 del sistema de codificación ASCII.

A continuación, se comprobará que la entrada de datos es válida y, de ser así, se recuperará el siguiente bloque a mostrar a través de la clase NEXT. Este proceso, también nos permitirá verificar si el elemento que pulsó la tecla de aceptar es el elemento EMAIL y, si es así, en el elemento siguiente a él, se sustituirá la palabra clave @NAME por el nombre solicitado con anterioridad. Además, si todo ha ido bien, se almacenarán localmente todos los datos solicitados a través de la API LOCALSTORAGE de HTML5 para no tener que pedirle los datos todas las veces que entre.

Ahora que tenemos controlado el mostrado de sentencias y la acción de aceptar, definiremos un método llamado SEND que será quién envíe al servidor el mensaje por el que el usuario desea contactar con nosotros. No obstante, como esto es una maqueta, sólo mostraremos un mensaje de confirmación y crearemos otra entrada de datos de forma dinámica mediante la función APPENDNEWSENTENCE.

Esta función APPENDNEWSENTENCE, será quién replique el bloque de la última sentencia y vaya modificando los atributos y valores para que se pueda seguir utilizando la página.

Como hemos visto antes, se ha realizado una llamada a una función denominada SAVEDATA. Esta función es quién almacena en la LOCALSTORAGE la información del usuario para reutilizarla más tarde.

Antes de finalizar con este script, definiremos una función o método denominado TOGGLEMAXIMIZE y un manejador de eventos para cuando se realiza un cambio de tamaño en la resolución de pantalla disponible a través de WINDOW.ONRESIZE. Lo que nos permitirán estos métodos es gestionar la API SCREEN de HTML5 para cambiar el modo de presentación del contenido de la página. Esto es, a pantalla completa sin la barra de direcciones, menús, marcadores ni botones o, en modo normal, como es lo habitual.

Finalmente, dado que guardamos la información en local, definiremos un proceso de recuperación tras la carga de la página. Este proceso lo que realizará es la ocultación de los campos ya almacenados, personalizar la entrada con el nombre y poner el cursor en la sentencia que solicita el motivo del contacto.

## CAMBIOS EN EL ARCHIVO STYLES.CSS

En lo referente al CSS, lo primero que realizaremos es la definición de unos estilos generales para el documento. Dado que esta página es especial, modificaremos los estilos asociados al elemento BODY, al elemento MAIN y al elemento ARTICLE. No obstante, no serán los únicos elementos a modificar o retocar, también deberemos actualizar los estilos aplicados a la imagen de la cabecera y de los elementos de párrafo.

Al igual que en prácticas anteriores, el contenido del artículo se estableció a través de un elemento FIELDSET. Este elemento será tratado, en esta práctica, de manera especial porque será quién proporcione la apariencia de “terminal”. Además, para que nos recuerde a los monitores de los 80, se le dotará de un efecto de escaneado vertical a través de la propiedad BEFORE.

Si nos fijamos en el código, veremos que para hacer el efecto del escaneado y del fondo del terminal se ha recurrido a gradientes, uno repetitivo y otro no. Además, para que se ajuste a la mayor parte de resoluciones de pantalla se han definido varias consultas de medios que establecen el ancho en función de la altura.

Ahora, por comodidad, lo que haremos es establecer unos estilos generales para todos los elementos que estén dentro de nuestro FIELDSET. Entre estos estilos, estableceremos un color por defecto, un formato de letra con peso y tamaño específicos y una separación entre caracteres similar a la utilizada en las resoluciones de entonces. Además, si el dispositivo es móvil, le aumentaremos el tamaño de letra para mejorar su lectura y legibilidad.

Como se puede observar, en la regla .CONTACT FIELDSET \*, se utiliza una fuente llamada “Inconsolata”, por lo que se deberá añadir a nuestra hoja de estilos al principio de la misma. A continuación, se muestran las fuentes que deberían estar declaradas después de estos últimos cambios.

Al igual que los programas de la época, definiremos una botonera en la parte superior con las acciones de minimizar, maximizar y cerrar. Todas estas acciones se definirán dentro del elemento LEGEND del FIELDSET.

Por último, sólo nos queda definir los estilos del FIELDSET, incluyendo sus elementos internos, cuando está en modo maximizado. Es decir, como el modo maximizado ocultará la cabecera y pie de la página, le pondremos el identificador del sitio web en la leyenda, cambiaremos los anchos y altos del FIELDSET para que se ajuste al cien por cien, y definiremos un nuevo estilo para el botón TOGGLE ya que ahora está maximizado y el símbolo de restaurar muy diferente.

# 7

## IFRAMES Y OBJETOS

Los marcos (frames) y los objetos son utilizados para insertar contenidos de otras tecnologías en un documento HTML y, en ocasiones, incluso para insertar otro documento HTML.

Históricamente se usaban para poder tener los elementos de cabeceras, menús de navegación y pies de página comunes y, de esta forma, actualizar el menor contenido posible, provocando que la carga fuese más rápida.

En este capítulo vamos a ver cómo definir algunos iframes y marcos de manera genérica, cómo hacerlos adaptativos o Responsive y, si cabe, cómo hacerlos usables y accesibles.

### ELEMENTOS DISPONIBLES EN HTML5

#### Elemento iframe

El elemento IFRAAME especifica que el contenido que se va a representar es un marco en línea que contiene otro documento dentro del propio documento actual.

```
<iframe src="https://api.perlego.com/assets/asset?  
session=73694023674e702820461842421bcf8105ceb462cc054fb7bbaf5ae29d81841&fileId=d632670ed6fd8727f4f100268add149f">  
El navegador no soporta el elemento iframe.  
</iframe>
```

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>height</b>	Especifica el alto en píxeles del marco.
<b>name</b>	Especifica o asigna el nombre del elemento.  El atributo NAME puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS.  Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los

	distintos elementos a través de un índice, siendo, el índice CERO el primero y, N-1, el último.
<b>sandbox</b>	Especifica las restricciones para el contenido del IFRAME. Entre otras opciones, se puede controlar, el envío de formularios, ventanas modales y emergentes, la ejecución de scripts, la navegación por el contexto de nivel superior o que sea tratado como si fuese del mismo origen. Si no se especifica valor, el marco permitirá todas las restricciones. Si se desea especificar uno o varias restricciones deberá hacerse a través de la lista de valores admitidos separados por espacios.
<b>src</b>	Específica la URL del documento que se desea incrustar en el marco.
<b>srcdoc</b>	Especifica el contenido HTML de la página que se desea mostrar en el elemento IFRAME. Cabe destacar que, cuando este atributo está presente, el atributo SRC es anulado de forma automática. Además, puede no estar soportado por Internet Explorer y se suele utilizar de manera conjunta con el atributo SANDBOX.
<b>width</b>	Especifica el ancho en píxeles del marco.

## Elemento object

El elemento OBJECT especifica que el contenido que se va a representar es un objeto incrustado que contiene otro documento dentro del propio documento actual.

Aunque no es frecuente, el elemento OBJECT permite la inserción de una página web, sin embargo, este elemento es más recomendable para aquellos contenidos que sean de otras tecnologías diferentes a HTML. Si el objeto o documento que se desea agregar es HTML es mejor incrustarlo en línea o como parte del propio documento a través de alguna tecnología que permita la inclusión de contenidos externos, como pueda ser JavaScript.

Tampoco es una buena idea insertar imágenes a través de este elemento porque puede no ser procesada de manera correcta y, de hacerlo, su aplicación puede tener efectos no deseados tanto en temas de posicionamiento SEO, como en accesibilidad web.

```
<object width="384" height="288" data="presentacion.swf"></object>
```

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

<i>Atributo</i>	<i>Descripción</i>
<b>data</b>	Especifica la dirección o URL del objeto a incrustar.
<b>form</b>	Especifica el formulario al que pertenece el objeto, pero actualmente no está soportado por ningún navegador.
<b>height</b>	Especifica el alto del objeto en píxeles.
<b>name</b>	Especifica o asigna el nombre del elemento. El atributo NAME puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los

	distintos elementos a través de un índice, siendo, el índice CERO el primero y, N-1, el último.
<b>type</b>	Especifica el tipo de medio, antes conocido como MIME type, del documento vinculado. Los posibles valores que puede tomar se encuentran en la URL <a href="http://www.iana.org/assignments/media-types/">http://www.iana.org/assignments/media-types/</a> .
<b>usemap</b>	Especifica que el objeto tiene asociado un mapa de imágenes con áreas en las que se puede hacer clic. El valor de este atributo debe ser el mismo que el atributo NAME del elemento MAP.
<b>width</b>	Especifica el ancho del objeto en píxeles.

Cabe destacar que, el elemento OBJECT puede utilizarse en formularios y que su declaración ya no puede hacerse dentro del elemento HEAD de un documento HTML.

## Elemento embed

El elemento EMBED especifica que el contenido que se va a representar es una aplicación externa o contenido interactivo.

```
<embed src="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbc8105ceb462cc054fb7bbaf5ae29d81841&fileId=375f43f1c4716e2a74e39713391c87c1"
type="audio/mid" autostart="false">
```

El elemento EMBED es frecuente utilizarlo para reproducir películas Flash, o sonidos multimedia a través de la Interfaz Digital de Instrumentos Musicales (MIDI).

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>height</b>	Especifica el alto del objeto en píxeles.
<b>src</b>	Especifica la dirección o URL del objeto a incrustar.
<b>type</b>	Especifica el tipo de medio, antes conocido como MIME type, del documento vinculado. Los posibles valores que puede tomar se encuentran en la URL <a href="http://www.iana.org/assignments/media-types/">http://www.iana.org/assignments/media-types/</a> .
<b>width</b>	Especifica el ancho del objeto en píxeles.

## ELEMENTOS DISPONIBLES EN CSS

CSS no presenta ninguna propiedad específica para marcos y objetos puesto que tienen un tratamiento especial. Sólo el elemento EMBED puede variar de las propiedades que CSS provee.

De hecho, los elementos IFRAME y OBJECT no pueden utilizar casi ninguna de las propiedades básicas o genéricas de CSS, si lo comparamos con otros elementos HTML. Lo más frecuente es que sólo apliquen unos estilos de tamaños de caja, color de fondo, bordes y márgenes externos o internos.

## **USABILIDAD Y ACCESIBILIDAD EN LOS MARCOS**

El término marco o frame tiene varios significados según a qué se refiera. En usabilidad, por ejemplo, se refiere a elementos que permiten dividir la pantalla en áreas independientes con el objetivo de cargar contenidos diferentes pertenecientes a otros sistemas o interfaces.

Entre los atributos que se deben definir para que estos elementos sean algo más usables y accesibles, el más importante quizás sea TITLE, porque puede servir para proporcionar una descripción del marco.

Entre los elementos que debe contener un marco, el más importante quizás sea el elemento A (enlace), porque permite el acceso al contenido si, por cualquier razón, no pudiese cargarse el marco en la página.

Puede parecer que son una buena opción, sin embargo, el uso de marcos no está recomendado porque limitan las funcionalidades básicas conocidas, provocan desplazamientos innecesarios y pueden confundir al usuario, hasta el punto de rechazar su uso. Además, son susceptibles a ataques de phishing, pueden dar problemas de seguridad y de posicionamiento SEO y son totalmente inaccesibles.

En consecuencia, tanto el elemento OBJECT, como el elemento EMBED y el elemento IFRAME deben utilizarse con cuidado y ser el último recurso para insertar un contenido externo. De hecho, deben escogerse en este orden, es decir, siempre será mucho más usable y accesible el elemento OBJECT que el resto.

## **IFRAMES Y OBJETOS MULTIMEDIA RESPONSIVE**

Uno de los pocos usos que tienen los objetos hoy día es la incrustación de vídeos de YouTube o Vimeo. El proceso de inserción en una web es francamente sencillo, sin embargo, conseguir que estos contenidos multimedia se vean de forma adecuada puede ser algo tedioso.

Para conseguir que los contenidos multimedia insertados a través de IFRAME u OBJECT se vuelvan receptivos y, por tanto, más usables, lo primero que se debe hacer embeberlos en una capa externa a modo de contenedor.

```
<div class="video-responsive">
<iframe src="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbc8105ceb462cc054fb7bbaf5ae29d81841&fileId=c1bdd254d8a4719b40523a8eb9a2482e"
frameborder="0"
allow="accelerometer; autoplay; encrypted-media;
gyroscope; picture-in-picture"
allowfullscreen>
</iframe>
</div>
```

Una vez hecho esto, al contendor se le asignará una relación de aspecto intrínseca para el vídeo. La forma de conseguir esta relación de aspecto es a través de la propiedad PADDING, la cual permite que una caja tome una relación de aspecto determinada en función del ancho de la capa contenedora.

```
.video-responsive {  
    overflow: hidden;  
    padding-bottom: 56.25% !important;  
    padding-top: 25px !important;  
    position: relative !important;  
}
```

La razón de por qué este PADDING es 56.25%, es porque la relación del video esperada es 16:9. De hecho, calcular este valor resulta tan sencillo como aplicar una sencilla regla de tres:

$$PADDING_{BOTTOM} = 9 * \frac{100}{16} = 56.25\%$$

Sin embargo, si la relación de aspecto esperada fuese 4:3, el valor del PADDING sería muy diferente:

$$PADDING_{BOTTOM} = 3 * \frac{100}{4} = 75\%$$

Ahora bien, la razón de por qué el PADDING-TOP es 25 píxeles es muy diferente. La altura del cromo es estática, independientemente de resolución del vídeo y, por ello, hay que ajustarlo de forma fija.

Si ahora aprovechamos las ventajas de que nos da el posicionamiento absoluto sobre el último relativo y ponemos el elemento IFRAME al cien por cien del ancho y alto del contendor, ya tenemos un vídeo totalmente responsive.

```
.video-responsive iframe,  
.video-responsive object,  
.video-responsive embed {  
    height: 100%;  
    left: 0;  
    position: absolute;  
    top: 0;  
    width: 100%;  
}
```

Si quisiéramos hacer esto mismo a través de OBJECT, en vez de IFRAME, sólo tendríamos que cambiar el atributo SRC por el atributo DATA.

```
<div class="video-responsive">  
<object data="https://api.perlego.com/assets/asset?  
session=73694023674e702820461842421cbc8105ceb462cc054fb7bbaf5ae29d81841&fileId=c1bdd254d8a4719b40523a8eb9a2482e">
```

```
frameborder="0"
allow="accelerometer; autoplay; encrypted-media;
gyroscope; picture-in-picture"
allowfullscreen>
</object>
</div>
```

Como se puede apreciar, no es tan difícil como se podría pensar, sin embargo, esta solución no es la única. Existen otras opciones que presentan algunas modificaciones, pero que resultan útiles en situaciones particulares.

Otra posible forma de hacer que este tipo de recursos se vuelvan adaptables a cualquier dispositivo es hacer lo siguiente:

```
<style>
.video-container {
    overflow: hidden;
    position: relative;
    width: 100%;
}
.video-container::after {
    padding-top: 56.25%; /* Esto es porque 9 es el 56.25% de 16 */
    display: block;
    content: "";
}
.video-container iframe {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
}
</style>
<div class="video-container">
<iframe allow="accelerometer; autoplay; encrypted-media; gyroscope;
picture-in-picture"
allowfullscreen
frameborder="0"
src="https://api.perlego.com/assets/asset?
session=73694023674e702820461842421cbef8105ceb462cc054fb7bbaf5ae29d81841&fileId=eaf09a1ce987e8fc9e310658403bb9fa">
</iframe>
</div>
```

En esta ocasión, dado que el video está en una relación de aspecto 16:9, la opción por la que se ha optado es establecer un contenedor que tiene asignado un posicionamiento relativo, sin posibilidad de desbordamiento y unos márgenes internos que mantengan las proporciones del vídeo. Luego, al marco, se le dota de un posicionamiento absoluto para que se ajuste al 100% del contenedor permitiendo, así, que se ajuste a todos los dispositivos y/o resoluciones.

# 8

## ANIMACIONES, TRANSICIONES Y EFECTOS

### TRANSFORMACIONES

Una transformación es un cambio de “estado” sobre la forma que la ejecuta. Esto se suele traducir en cambios en la escala, rotación, sesgado o desplazamiento del elemento en base a un sistema de coordenadas de dos dimensiones, aunque es posible conseguir efectos en sistemas de coordenadas de 3 dimensiones.

Por intentar ser más preciso y claro, a partir de este momento y para los ejemplos de esta sección, supondremos un elemento DIV al que se le han definido unos estilos de 100 píxeles de ancho, 100 píxeles de alto, un fondo con degradado vertical de blanco a negro y un borde de 2 píxeles negros, sin márgenes internos ni externos. Esto es:

```
div {  
width: 100px;  
height: 100px;  
background: linear-gradient(odeg, black 0, black 50%, white);  
border: 2px solid #000;  
margin: 0;  
padding: 0;  
}
```

#### Función de escalado (scale)

La función SCALE permite cambiar el tamaño del objeto respecto de su tamaño original.

La manera de especificar la relación de tamaño es a través de un valor en tanto por uno. Este valor de relación de tamaño puede especificarse a nivel global, es decir, para ambos ejes X e Y o, de manera independiente, es decir, un valor concreto para cada eje, lo que permite romper la relación de aspecto de la figura u objeto original.

		
<b>Forma original</b>	<b>Valor 0.5</b>	<b>Valor 2.0</b>

### Ejemplos:

```
div { transform: scale(0.5); }
div { transform: scale(0.5, 1.5); }
```

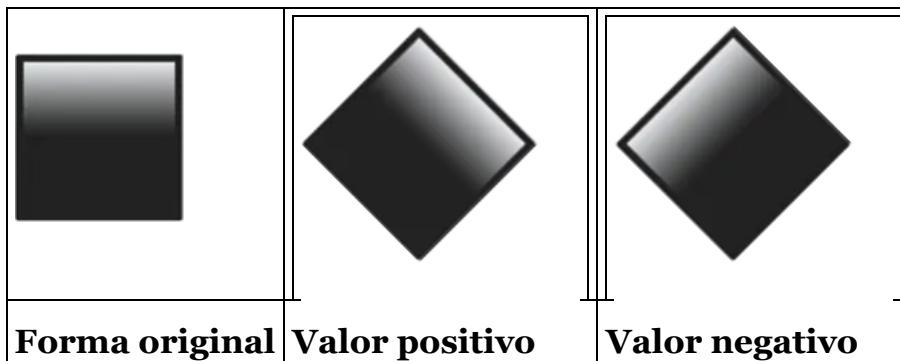
### Función de rotación (rotate)

La función ROTATE permite girar un objeto respecto a cualquiera de los ejes de un sistema de coordenadas tridimensional.

La manera más sencilla de especificar la cuantía del giro es a través de un valor en grados, es decir, mediante un valor comprendido entre 0 y 360, seguido del sufijo DEG. No obstante, también es posible especificar el valor de la rotación en radianes. La forma de calcular los radianes a partir de los grados es:

$$RAD = Grados * \frac{\pi}{180} = 1^\circ * \frac{3.14159265359...}{180} = 0.0174533 \text{ radianes}$$

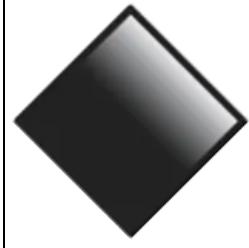
Si se especifica un valor positivo, el giro se realizará en el sentido de las agujas del reloj y, si se especifica un valor negativo, el giro se realizará en el sentido contrario a las agujas del reloj.



### Ejemplos:

```
div { transform: rotate(45deg); }
div { transform: rotate(-45deg); }
```

Si se desea que el giro se realice en un eje en particular, se debe recurrir a ROTATEX, ROTATEY o ROTATEZ.

			
<b>Forma original</b>	<b>rotateX(45deg)</b>	<b>rotateY(45deg)</b>	<b>rotateZ(45deg)</b>

## Función de sesgado (skew)

La función SKEW permite inclinar un objeto respecto a cualquiera de los ejes de un sistema de coordenadas bidimensional.

Al igual que sucede con la función de ROTATE, la manera más sencilla de especificar la cuantía de la inclinación es a través de un valor en grados, es decir, mediante un valor comprendido entre 0 y 360, seguido del sufijo DEG. No obstante, también es posible especificar el valor de la rotación en radianes. La forma de calcular los radianes a partir de los grados es:

$$RAD = Grados * \frac{\pi}{180} = 1^\circ * \frac{3.14159265359...}{180} = 0.0174533 \text{ radianes}$$

Si se especifica un valor positivo, la inclinación se realizará de modo que, la parte superior, se irá posicionando más hacia la izquierda y, la parte inferior, más hacia la derecha. Por el contrario, si se especifica un valor negativo, la inclinación se realizará de modo que, la parte superior, se irá posicionando más hacia la derecha y, la parte inferior, más hacia la izquierda.

		
<b>Forma original</b>	<b>Inclinación positiva</b>	<b>Inclinación negativa</b>

## Ejemplos:

```
div { transform: skew(45deg); }
div { transform: skew(-45deg); }
```

## Función de traslado (translate)

La función TRANSLATE permite mover un objeto respecto en cualquiera de los ejes de un sistema de coordenadas tridimensional.

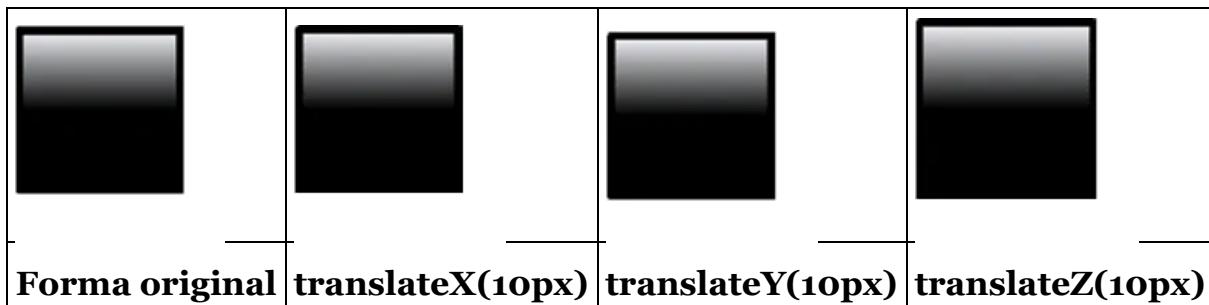
La manera habitual de especificar este desplazamiento es en píxeles o porcentajes, no obstante, es posible especificarlo en cualquiera de las unidades de medida compatibles con CSS.



### Ejemplos:

```
div { transform: translate(100px); }
div { transform: translate(-100px); }
```

Si se desea que el desplazamiento se realice en un eje en particular, se debe recurrir a TRANSLATEX, TRANSLATEY o TRANSLATEZ según el eje que se quiera manipular.



Por último, hay que destacar que para que el desplazamiento en el eje Z se lleve a cabo, hay que especificar, al menos, un valor de perspectiva, es decir, un valor máximo para el eje Z. Por intentar ser algo más claros, si el contenedor dónde está el elemento es de 100x100 píxeles, lo habitual es establecer la perspectiva a 100 píxeles:

### Ejemplos:

```
div { transform: perspective(100px) translateZ(10px); }
```

## FILTROS O EFECTOS

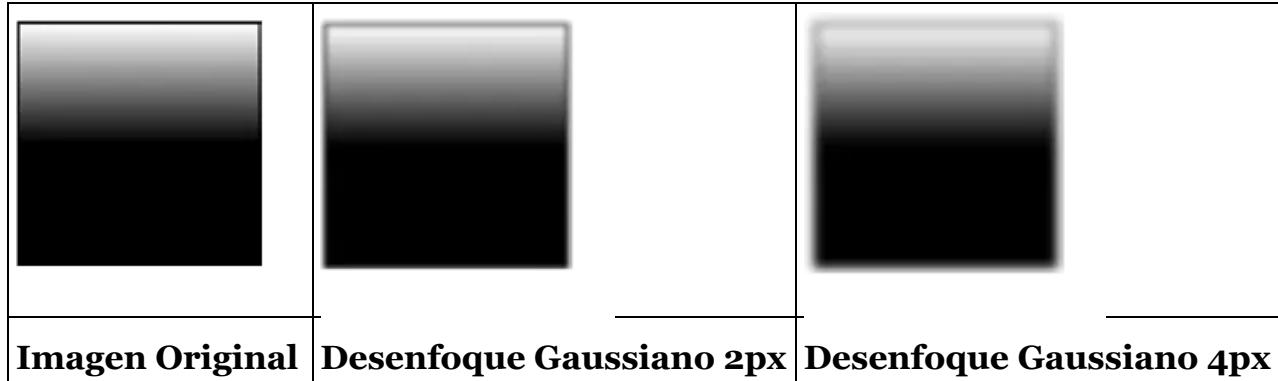
Un filtro de CSS es una transformación en la representación gráfica del elemento u objeto. En general, se aplican únicamente a imágenes rasterizadas, aunque también es posible aplicarlos a imágenes vectoriales.

### Desenfoque (blur)

La función BLUR tiene como objetivo realizar un desenfoque gaussiano tomando como parámetro una desviación estándar. Esto significa que cuanto mayor sea este valor, mayor será el efecto de desenfoque.

Para quién no lo sepa, un desenfoque gaussiano es un efecto de suavizado realizado a través de un algoritmo o fórmula matemática y que se basa en calcular un nuevo color a partir del color de un pixel dado y los que están a su alrededor.

La consecuencia directa de esta nueva imagen será la pérdida de algunos detalles con respecto a la imagen original provocando una sensación de pérdida de nitidez o claridad en los bordes del pixel tratado.



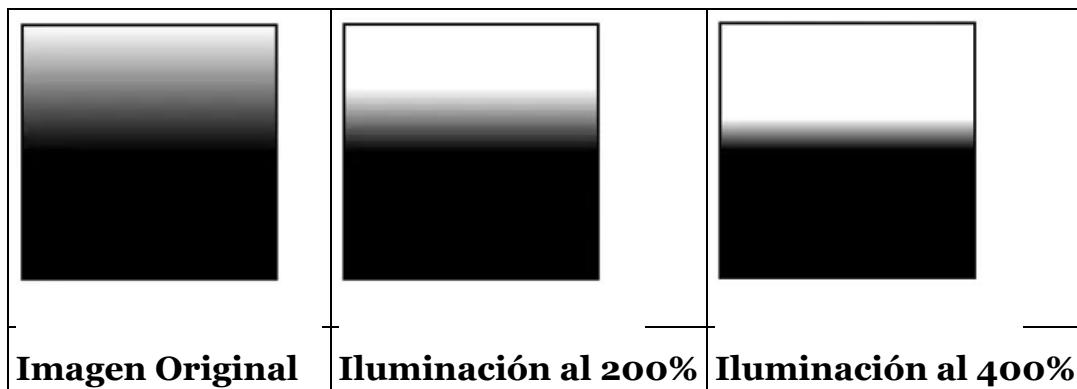
### Ejemplos:

```
div { filter: blur(0px); /* Imagen izquierda */ }
div { filter: blur(2px); /* Imagen central */ }
div { filter: blur(4px); /* Imagen derecha */ }
```

### Brillo o iluminación (brightness)

La función BRIGHTNESS tiene como objetivo realizar una compensación de iluminación regular a todo el objeto o imagen. Los valores de esta función se suelen especificar en tanto por uno, aunque también es posible especificar sus valores en porcentajes.

Si estamos en la unidad de medida en tanto por uno, significa que, si el valor es menor de uno, la imagen se volverá más oscura y, si el valor es mayor que uno, la imagen se volverá más clara.



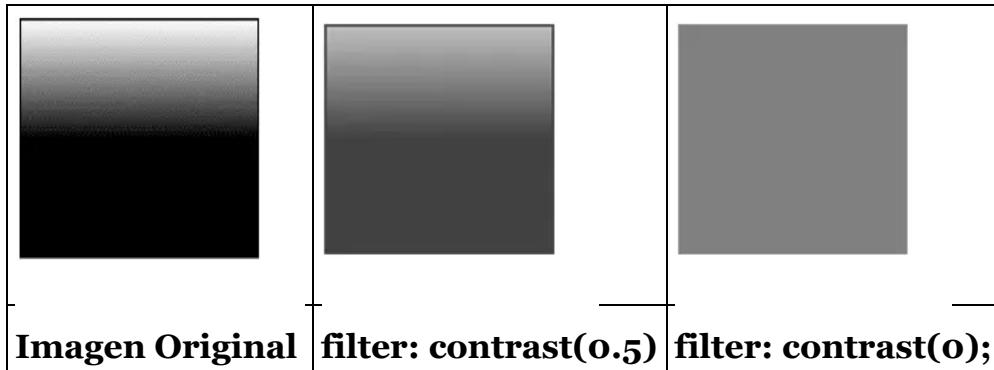
### Ejemplos:

```
div { filter: brightness(0); /* Imagen izquierda */ }
div { filter: brightness(2); /* Imagen central */ }
div { filter: brightness(4); /* Imagen derecha */ }
```

## Contraste (contrast)

La función CONTRAST tiene como objetivo realizar una compensación de contraste a todo el objeto o imagen. Los valores de esta función se suelen especificar en tanto por uno, aunque también es posible especificar sus valores en porcentajes.

Si estamos en la unidad de medida en tanto por uno, los valores permitidos son entre 0 y 1. Esto significa que, si el valor es cero, la imagen eliminará todo el contraste disponible, por lo que el negro se volverá gris puro y, si el valor es uno, la imagen no aplicará ningún ajuste de contraste, por lo que se verá la imagen original.



### Ejemplos:

```
div { filter: contrast(1); /* Imagen izquierda */ }
div { filter: contrast(0.5); /* Imagen central */ }
div { filter: contrast(0); /* Imagen derecha */ }
```

## Sombra paralela (drop-shadow)

La función DROP-SHADOW tiene como objetivo realizar un efecto de sombra paralela sobre el objeto o imagen.

Por entendernos, una sombra podría definirse como una copia desenfocada e independiente del canal alfa con una tonalidad de color determinada y situada por debajo de la imagen original.

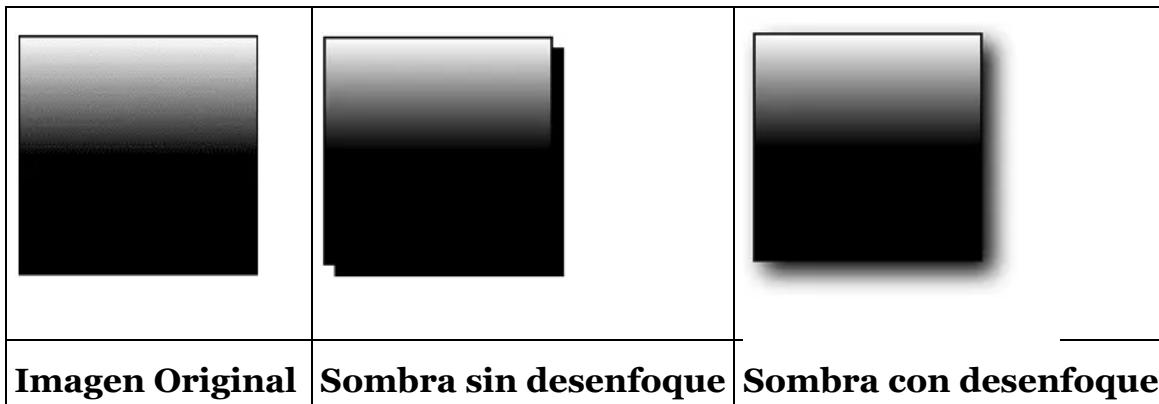
Los valores de esta función se suelen especificar en píxeles, aunque también es posible especificar sus valores en cualquiera de las unidades de medida compatibles de CSS.

La función DROP-SHADOW se alimenta de dos valores de posición, un valor para indicar el radio de desenfoque, un valor para indicar el radio de propagación y código de color.

Los dos primeros valores de posicionamiento son las coordenadas X e Y dónde se trasladará la sombra con respecto a la posición original. Para un valor de X positivo, la sombra se desplazará hacia la derecha, mientras que, para un valor negativo, la sombra se desplazará hacia la izquierda. Algo similar pasa con el eje Y. Para un valor de Y positivo, la sombra se desplazará hacia abajo, mientras que, para un valor negativo, la sombra se desplazará hacia arriba.

El parámetro de radio de desenfoque, el tercer valor, producirá un mayor desenfoque cuanto mayor sea su valor. Un valor de 0 no aplicará ningún efecto de desenfoque y, como consecuencia de su lógica, no admitirá valores negativos.

El parámetro de radio de propagación, el cuarto valor, provocará que la sombra se expanda o contraiga en función de su valor. Por defecto es 0 lo que producirá que se aplique una sombra con un tamaño equivalente al tamaño de la imagen original. No obstante, este parámetro no es compatible con muchos navegadores, por lo que si se incluyen en la definición de la sombra, puede que no se realice el efecto por un error de sintaxis.



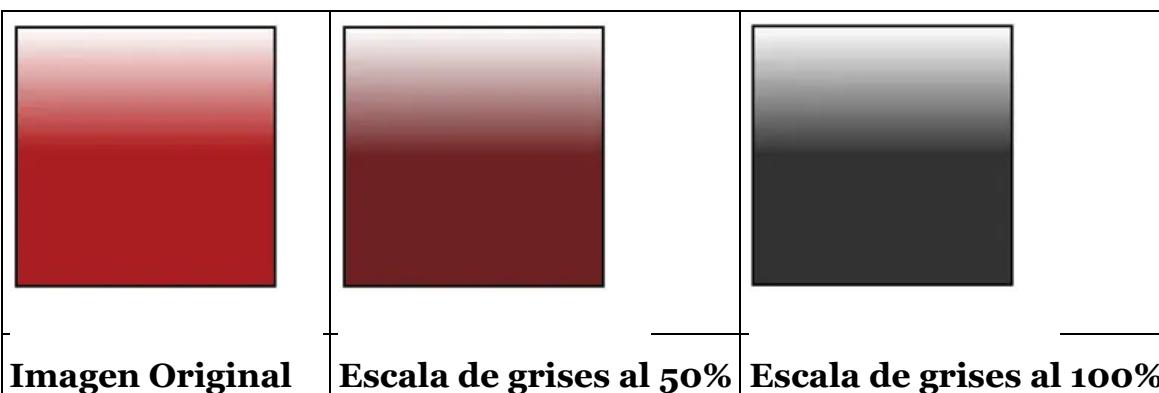
### Ejemplos:

```
div { filter: drop-shadow(0px 0px 0px #000); /* Imagen izquierda */ }
div { filter: drop-shadow(10px 10px 0px #000); /* Imagen central */ }
div { filter: drop-shadow(10px 10px 10px #000); /* Imagen derecha */ }
```

### Escala de grises (grayscale)

La función GRayscale tiene como objetivo eliminar de la representación gráfica de un objeto sus matices de color. Los valores de esta función se suelen especificar en tanto por uno, aunque también es posible especificar sus valores en porcentajes.

Si estamos en la unidad de medida en tanto por uno, los valores permitidos son entre 0 y 1. Esto significa que, si el valor es cero, la imagen tendrá todos los matices de color y, si es uno, se aplicará el filtro de manera completa dejando la imagen en escala de grises.



Aunque en una impresión en blanco y negro no se perciba, si estableciésemos un degradado rojo en imagen de la izquierda, la imagen del centro se percibiría como rojo oscuro con matices marrones y, la imagen de la derecha se percibiría como se ve en el papel, es decir, en tonos grisáceos.

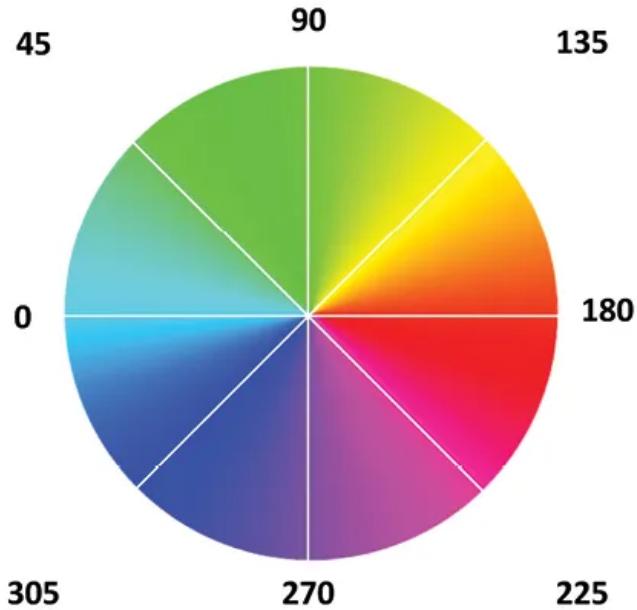
### Ejemplos:

```
div { filter: grayscale(0); /* Imagen izquierda */ }
div { filter: grayscale(0.5); /* Imagen central */ }
div { filter: grayscale(1); /* Imagen derecha */ }
```

## Rotación de color (hue-rotate)

La función HUE-ROTATE tiene como objetivo aplicar una modificación en la tonalidad de los colores en base a una desviación calculada a través de un ángulo.

Para entender mejor este concepto pongamos un ejemplo. Si tomamos una imagen cualquiera y nos fijamos en un color concreto de la misma, veremos que dicho color tiene una posición concreta dentro de lo que se denomina “círculo cromático natural degradado”.

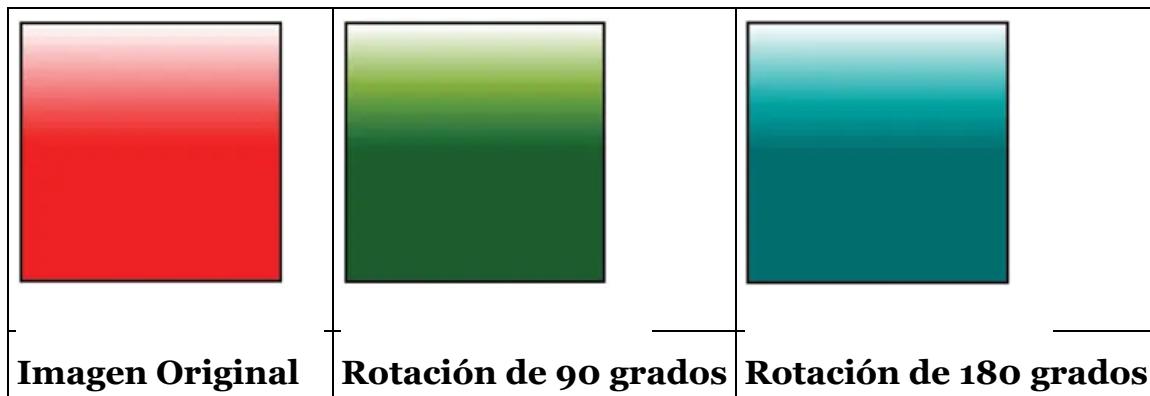


Círculo cromático natural degradado. Fuente: Wikipedia, la enciclopedia libre.

Por ejemplo, si tomamos como referencia el color amarillo, vemos que está situado, más o menos, a la 1 y media, es decir, formando un ángulo de 135 grados. Pero, si tomamos como referencia el color rojo, vemos que está situado, más o menos, a las 3 en punto, o formando un ángulo de 180 grados.

Al aplicar una rotación de color con un valor determinado de grados, lo que se consigue es, por decirlo así, que el círculo cromático gire en el sentido de las agujas del reloj, provocando que, el amarillo se convierta en turquesa y que, el rojo, se convierta en un verde.

Cabe destacar que, puede que, al aplicar este filtro, los resultados no se parezcan a los colores expuestos en el gráfico. Esto es porque la rotación de color sólo afecta al color y no al contraste, iluminación o saturación.



### Ejemplos:

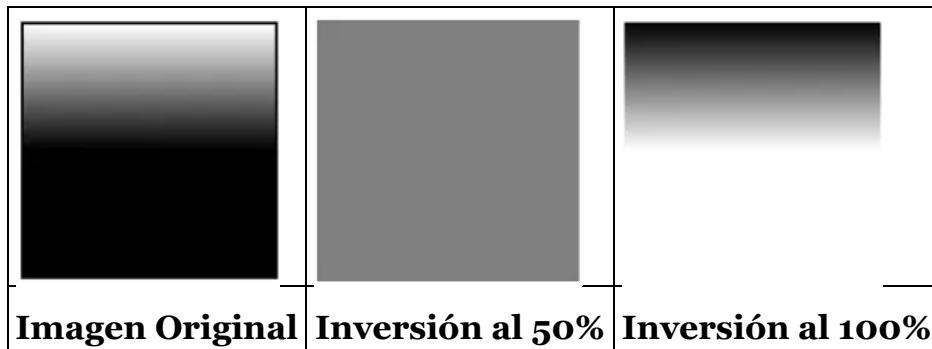
```
div { filter: hue-rotate(0deg); /* Imagen izquierda */ }
div { filter: hue-rotate(90deg); /* Imagen central */ }
div { filter: hue-rotate(180deg); /* Imagen derecha */ }
```

Si observamos el resultado de aplicar una rotación de color de 180 grados sobre el color rojo, lo que veremos es que se transforma en su color complementario, al igual que hace la función INVERT.

### Inversión (invert)

La función INVERT tiene como objetivo transformar un color dado en su opuesto o complementario. Los valores de esta función se suelen especificar en tanto por uno, aunque también es posible especificar sus valores en porcentajes.

Si estamos en la unidad de medida en tanto por uno, los valores permitidos son entre 0 y 1. Esto significa que, si el valor es cero, la imagen será representada con sus colores originales y, si es uno, la imagen será representada con sus colores complementarios.



Como se puede apreciar, cuando la inversión es equitativa (al 50 por ciento o 0.5), sea cual sea el color del pixel, se convierte en negro al 50%, es decir, un color RGB(128, 128, 128) o #808080 en

hexadecimal. La razón de este resultado es porque la mezcla de un color con su complementario siempre genera un gris puro.

### Ejemplos:

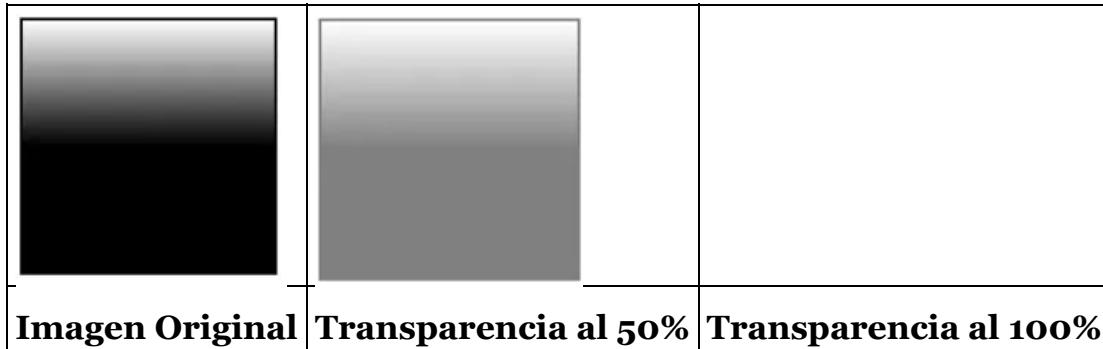
```
div { filter: invert(0); /* Imagen izquierda */ }
div { filter: invert(0.5); /* Imagen central */ }
div { filter: invert(1); /* Imagen derecha */ }
```

## Opacidad (opacity)

La función OPACITY tiene como objetivo transformar el canal alfa de los colores para que se vuelva transparente. Esta función es básicamente la misma que la propiedad OPACITY de CSS, no obstante, es preferible aplicar esta funcionalidad a través de FILTER puesto que, algunos navegadores aprovechan la aceleración hardware en este filtro.

Los valores de esta función se suelen especificar en tanto por uno, aunque también es posible especificar sus valores en porcentajes.

Si estamos en la unidad de medida en tanto por uno, los valores permitidos son entre 0 y 1. Esto significa que, si el valor es cero, la imagen será totalmente transparente y, si es uno, la imagen será totalmente opaca.



### Ejemplos:

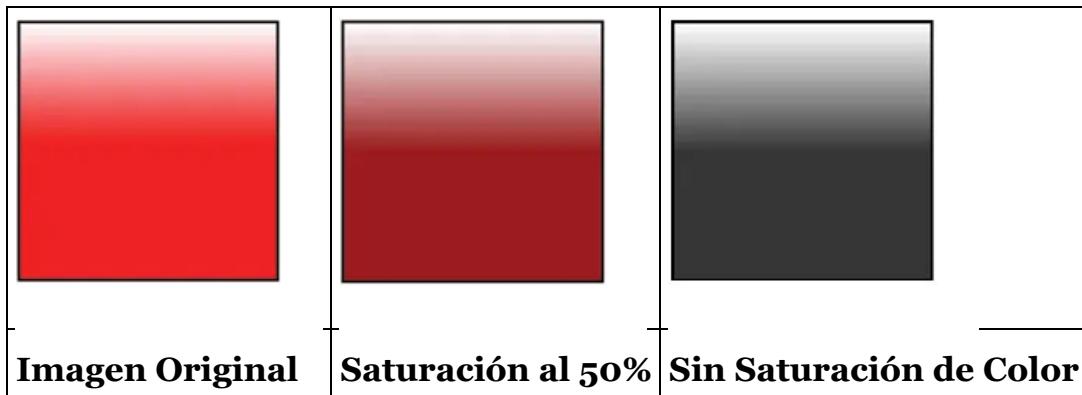
```
div { filter: opacity(1); /* Imagen izquierda */ }
div { filter: opacity(0.5); /* Imagen central */ }
div { filter: opacity(0); /* Imagen derecha */ }
```

## Saturación (saturate)

La función SATURATE tiene como objetivo modificar la saturación de un color dado. Los valores de esta función se suelen especificar en tanto por uno, aunque también es posible especificar sus valores en porcentajes.

Si estamos en la unidad de medida en tanto por uno, significa que, si el valor es menor de uno, la imagen perderá saturación y, si el valor es mayor que uno, la imagen se irá sobresaturando. El

valor cero, la convertirá en una imagen a escala de grises, equivalente a la función GRayscale(1).



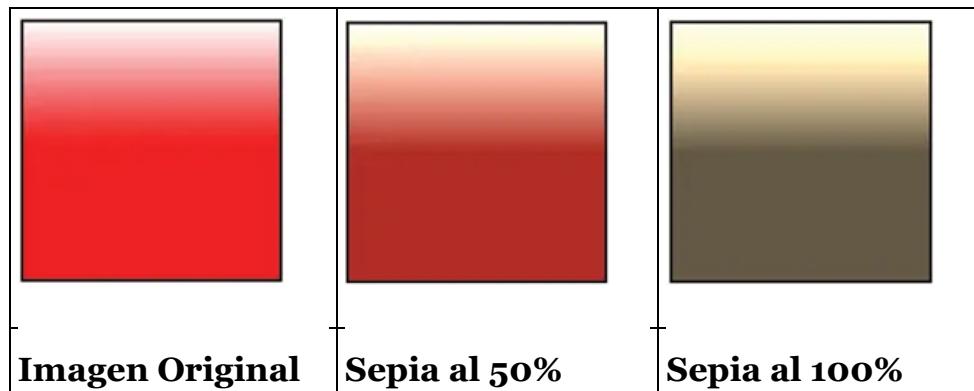
### Ejemplos:

```
div { filter: saturate(1); /* Imagen izquierda */ }
div { filter: saturate(0.5); /* Imagen central */ }
div { filter: saturate(0); /* Imagen derecha */ }
```

## Sepia (sepia)

El efecto SEPIA se suele corresponder con los pigmentos obtenidos de la tinta de la sepia y, en CSS, tiene como objetivo transformar los colores de forma que tomen un color rojo anaranjado con poca o muy poca saturación, como si de un envejecimiento o pérdida de pigmentos de color se tratase. Los valores de esta función se suelen especificar en tanto por uno, aunque también es posible especificar sus valores en porcentajes.

Si estamos en la unidad de medida en tanto por uno, los valores permitidos son entre 0 y 1. Esto significa que, si el valor es cero, la imagen será representada con sus colores originales y, si es uno, la imagen será representada con la aplicación del efecto por completo.



### Ejemplos:

```
div { filter: sepia(0); /* Imagen izquierda */ }
div { filter: sepia(0.5); /* Imagen central */ }
div { filter: sepia(1); /* Imagen derecha */ }
```

## TRANSICIONES

Una transición es un efecto de animación que permite realizar cambios en las propiedades de un objeto de manera progresiva o escalonada controlando el tiempo y la velocidad. Por ejemplo, frecuentemente vemos que los botones cambian el color de fondo cuando el dispositivo señalador pasa por encima de ellos. Muchas veces este cambio de color se realiza de forma abrupta, es decir, el cambio de un color a otro se realiza en el mismo instante en el que el cursor se sitúa sobre el botón sin ningún tipo de mezcla ni degradación. Sin embargo, si definimos una transición, podemos conseguir que el cambio de color se realice de manera gradual en base a una medición de tiempo y velocidad determinadas.

Cabe destacar que, aunque en general, las transiciones se realizan de manera gradual, dependiendo de qué propiedades intervengan en el proceso, podrían producirse cambios de estado bruscos no graduales.

También es importante aclarar que, aunque todos los navegadores actuales ya soportan la especificación de la W3C, puede que sea necesario la utilización de prefijos específicos como son -O-TRANSITION, -MOZ-TRANSITION y -WEBKIT-TRANSITION.

### Propiedad transition-delay

Especifica el retraso de empiece, es decir, el tiempo que se debe esperar para iniciar la transición cuando el elemento sufre un cambio en la propiedad o propiedades indicadas por la propiedad TRANSITION-PROPERTY. Entre sus posibles valores podemos encontrar:

- **[TIEMPO]**: Es un valor decimal que expresa el número de segundos o milisegundos que se debe esperar antes de iniciar la transición. El valor por defecto es 0, lo que indica que empiece inmediatamente sin retraso alguno, pero, además, permite el establecimiento de valores negativos, que representan un valor de tiempo que indica la duración de la transición como si ya hubiese estado reproduciéndose de antes.

#### Ejemplos:

```
div { transition-delay: 1s; }
div { transition-delay: 1.25s; }
div { transition-delay: 25ms; }
```

### Propiedad transition-duration

Especifica la duración de la transición, es decir, el tiempo total que se debe invertir para el cambio en la propiedad o propiedades indicadas por la propiedad TRANSITION-PROPERTY. Entre sus posibles valores podemos encontrar:

- **[TIEMPO]**: Es un valor decimal que expresa el número de segundos o milisegundos que se debe tardar en realizar la transición. Su valor por defecto es 0.

## Ejemplos:

```
div { transition-duration: 1s; }
div { transition-duration: 0.75s; }
div { transition-duration: 250ms; }
```

## Propiedad transition-property

Especifica la propiedad, o lista de propiedades, que disparará la transición, es decir, el nombre de la propiedad o propiedades que, cuando cambien, provocarán que un efecto de transición se lleve a cabo. Entre sus posibles valores podemos encontrar:

- **ALL**: Indica que cualquier cambio en cualquier propiedad lanzará un efecto de transición.
- **NONE**: Indica que ninguna propiedad lanzará un efecto de transición.
- **[PROPIEDAD]**: Lista de propiedades CSS, separadas por coma, para las cuales el efecto de transición se llevará a cabo.

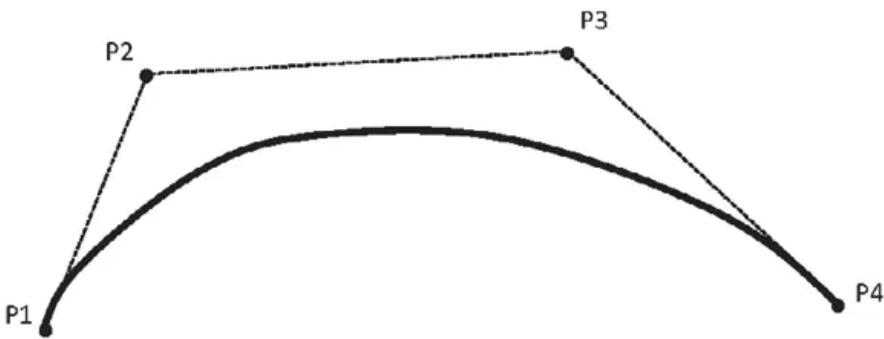
## Ejemplos:

```
div { transition-property: all; }
div { transition-property: transform; }
div { transition-property: width, height, padding; }
```

## Propiedad transition-timing-function

Especifica la función de sincronización de tiempo de la transición, es decir, cuál será la curva de velocidad que se deberá seguir durante el tiempo que dure la transición. Entre sus posibles valores podemos encontrar:

- **CUBIC-BEZIER**: Indica los valores decimales de 0.0 a 1.0 para los cuatro puntos de la curva de Bézier. En la siguiente ilustración se puede observar, a modo de ejemplo, una curva de Bézier que muestra cómo cambia la velocidad durante la ejecución de la transición.



- **EASE:** Indica que el efecto de transición debe empezar lento, luego volverse rápido y, al final, nuevamente lento. Es equivalente a una curva de Bézier cúbica CUBIC-BEZIER (0.25, 0.1, 0.25, 1). Es el valor por defecto.
- **EASE-IN:** Indica que el efecto de transición debe empezar lento, equivalente a una curva de Bézier cúbica CUBIC-BEZIER (0, 0, 1, 1).
- **EASE-IN-OUT:** Indica que el efecto de transición debe empezar y terminar lento, equivalente a una curva de Bézier cúbica CUBIC-BEZIER (0.42, 0, 0.58, 1).
- **EASE-OUT:** Indica que el efecto de transición debe empezar lento, equivalente a una curva de Bézier cúbica CUBIC-BEZIER (0, 0, 1, 1).
- **LINEAR:** Indica que el efecto de transición debe permanecer constante durante el tiempo que dure la transición, equivalente al valor EASE-OUT y a una curva de Bézier cúbica CUBIC-BEZIER (0, 0, 1, 1).
- **STEPS:** Indica una función de pasos con dos parámetros. El primero especifica el número de intervalos mediante un valor positivo distinto de cero y, el segundo, una de las palabras clave START o END que especifican el punto dónde se debe producir el cambio de intervalo.
- **STEP-END:** Indica que es un único paso que se produce al final, equivalente a STEPS (1, END).
- **STEP-START:** Indica que es un único paso que se produce en el comienzo, equivalente a STEPS (1, START).

### Ejemplos:

```
div { transition-timing-function: linear; }
div { transition-timing-function: ease-in-out; }
div { transition-timing-function: cubic-bezier(0.20, 0.1, 0.50, 1); }
```

# Animaciones

## PROPIEDAD ANIMATION-DELAY

Especifica el retraso de empiece, es decir, el tiempo que se debe esperar para iniciar la animación. Entre sus posibles valores podemos encontrar:

- **[TIEMPO]**: Es un valor decimal que expresa el número de segundos o milisegundos que se debe esperar antes de iniciar la animación. El valor por defecto es 0, lo que indica que empiece inmediatamente sin retraso alguno y permite el establecimiento de valores negativos, que representan u valor de tiempo que indica la duración de la animación como si ya hubiese estado reproduciéndose de antes.

### Ejemplos:

```
div { animation-delay: 1.50s; }  
div { animation-delay: 25ms; }
```

## PROPIEDAD ANIMATION-DIRECTION

Especifica la dirección de la animación, es decir, si debe reproducirse hacia adelante, hacia atrás o en ciclos alternos. Entre sus posibles valores podemos encontrar:

- **ALTERNATE**: Indica que la animación debe reproducirse primero hacia adelante y, después, hacia atrás.
- **ALTERNATE-REVERSE**: Indica que la animación debe reproducirse primero hacia atrás y, después, hacia adelante.
- **NORMAL**: Indica que la animación debe reproducirse hacia adelante. Es el valor por defecto.
- **REVERSE**: Indica que la animación debe reproducirse hacia atrás.

### Ejemplos:

```
div { animation-direction: reverse; }  
div { animation-direction: alternate; }  
div { animation-direction: alternate-reverse; }
```

## PROPIEDAD ANIMATION-DURATION

Especifica la duración de la animación, es decir, el tiempo total que se debe invertir para desarrollarla. Entre sus posibles valores podemos encontrar:

- **[TIEMPO]**: Es un valor decimal que expresa el número de segundos o milisegundos que se debe tardar en realizar la animación. Su valor por defecto es 0.

### Ejemplos:

```
div { animation-duration: 1s; }
div { animation-duration: 1.25s; }
div { animation-duration: 25ms; }
```

### PROPIEDAD ANIMATION-FILL-MODE

Especifica el estilo que debe presentar el elemento cuando la animación no se está reproduciendo, es decir, el estilo antes de empezar, después de terminar o en ambos casos. Entre sus posibles valores podemos encontrar:

- **BACKWARDS**: El elemento aparecerá con los valores establecidos por el primer fotograma definido por una regla arroba KEYFRAMES, dependiendo de la propiedad ANIMATION-DURATION, y se mantendrá durante el periodo que esté establecido por la propiedad ANIMATION-DELAY.
- **FORWARDS**: El elemento aparecerá con los valores establecidos por el último fotograma definido por una regla arroba KEYFRAMES, dependiendo de la propiedad ANIMATION-DURATION y del contador o recuento de iteraciones establecido por la propiedad ANIMATION-DELAY.
- **BOTH**: La animación seguirá las reglas en ambos sentidos de la animación.
- **NONE**: El elemento NO recibirá ningún estilo antes o después de la animación.

### Ejemplos:

```
div { animation-fill-mode: both; }
div { animation-fill-mode: forwards; }
div { animation-fill-mode: backwards; }
```

### PROPIEDAD ANIMATION-ITERATION-COUNT

Especifica el número de veces que la animación debe reproducirse. Entre sus posibles valores podemos encontrar:

- **INFINITE**: Indica que la animación NO debe parar nunca de reproducirse.
- **[NÚMERO]**: Es un valor entero que indica el número total de ocasiones que debe reproducirse. Por defecto, está establecido a 1.

### Ejemplos:

```
div { animation-iteration-count: infinite; }
span { animation-iteration-count: 3; }
aside { animation-iteration-count: 3; }
```

## PROPIEDAD ANIMATION-NAME

Especifica el nombre de la animación. Este nombre está directamente asociado al identificador definido por una regla arroba KEYFRAMES. Entre sus posibles valores podemos encontrar:

- **NONE**: Indica que la animación NO se debe desarrollar ninguna animación.
- **[NOMBRE]**: Es un valor de tipo cadena sin comillas simples o dobles que indica el nombre de la animación a utilizar. Debe ser el mismo que uno de los identificadores definidos por una regla arroba KEYFRAMES.

### Ejemplos:

```
div { animation-name: ejemplo1; }
span { animation-name: FadeIn; }
aside { animation-name: movetotop; }
```

## PROPIEDAD ANIMATION-PLAY-STATE

Especifica cuándo la animación debe ejecutarse (o continuar su ejecución) o debe ponerse en modo pausa. Entre sus posibles valores podemos encontrar:

- **PAUSED**: Indica que la animación se ponga en pausa.
- **RUNNING**: Indica que la animación se ejecute o siga ejecutándose.

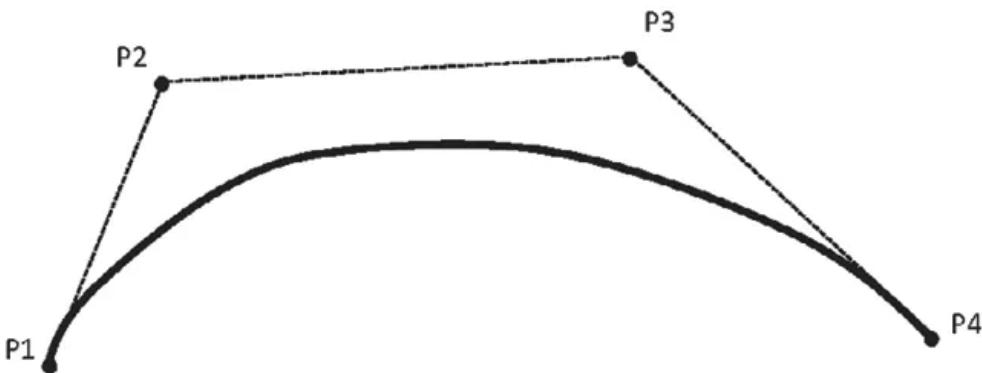
### Ejemplos:

```
div { animation-play-state: running; }
div:hover { animation-play-state: pause; }
```

## PROPIEDAD ANIMATION-TIMING-FUNCTION

Especifica la función de sincronización de tiempo de la animación, es decir, cuál será la curva de velocidad que se deberá seguir durante el tiempo que dure la animación. Entre sus posibles valores podemos encontrar:

- **CUBIC-BEZIER**: Indica los valores decimales de 0.0 a 1.0 para los cuatro puntos de la curva de Bézier. En la siguiente ilustración se puede observar, a modo de ejemplo, una curva de Bézier que muestra cómo cambia la velocidad durante la ejecución de la animación.



- **EASE:** Indica que el efecto de animación debe empezar lento, luego volverse rápido y, al final, nuevamente lento. Es equivalente a una curva de Bézier cúbica CUBIC-BEZIER (0.25, 0.1, 0.25, 1). Es el valor por defecto.
- **EASE-IN:** Indica que el efecto de animación debe empezar lento, equivalente a una curva de Bézier cúbica CUBIC-BEZIER (0, 0, 1, 1).
- **EASE-IN-OUT:** Indica que el efecto de animación debe empezar y terminar lento, equivalente a una curva de Bézier cúbica CUBIC-BEZIER (0.42, 0, 0.58, 1).
- **EASE-OUT:** Indica que el efecto de animación debe empezar lento, equivalente a una curva de Bézier cúbica CUBIC-BEZIER (0, 0, 1, 1).
- **LINEAR:** Indica que el efecto de animación debe permanecer constante durante el tiempo que dure la animación, equivalente al valor EASE-OUT y a una curva de Bézier cúbica CUBIC-BEZIER (0, 0, 1, 1).
- **STEPS:** Indica una función de pasos con dos parámetros. El primero especifica el número de intervalos mediante un valor positivo distinto de cero y, el segundo, una de las palabras clave START o END que especifican el punto dónde se debe producir el cambio de intervalo.
- **STEP-END:** Indica que es un único paso que se produce al final, equivalente a STEPS (1, END).
- **STEP-START:** Indica que es un único paso que se produce en el comienzo, equivalente a STEPS (1, START).

### Ejemplos:

```
div { animation-timing-function: linear; }
div { animation-timing-function: ease-in-out; }
div { animation-timing-function: cubic-bezier(0.20, 0.1, 0.50, 1); }
```

## REGLA KEYFRAMES

La regla @KEYFRAMES permite controlar todos y cada uno de los pasos que se producen en una secuencia de animación. Esto es útil cuando se desea que, el navegador, no controle la animación, como sucede con las transiciones en dónde se gestiona la evolución de la animación de forma automática a partir de unos parámetros iniciales.

La manera de especificar el número de pasos o partes de la animación se puede establecer a través de porcentajes o, mediante las palabras reservadas FROM y TO.

```
@keyframes desplazamiento-lento-rapido {  
from { top: 0; }  
to { top: 100px; }  
}
```

Si se decide por realizar la declaración de la animación a través de porcentajes, es obligatorio que se definan los estados inicial y final, es decir, los valores 0% y 100% puesto que, de no ser así, la animación podría no realizarse. A continuación, se muestra un ejemplo de animación con tres pasos:

```
@keyframes desplazamiento-lento-rapido {  
0% { top: 0; }  
50% { top: 30px; }  
100% { top: 100px; }  
}
```

Si analizamos el ejemplo anterior, veremos que, sea cual sea la velocidad y duración de la animación, durante la primera mitad de la secuencia, el objeto se moverá hacia abajo 30 píxeles, pero en la segunda mitad de la animación se desplazará 70. Esto producirá un efecto de, más lento al principio, más rápido al final.

En este punto, es importante aclarar que, únicamente las propiedades que sean definidas en el inicio y fin de la secuencia serán animadas, por lo que, si introducimos una propiedad entre medias, será ignorada. Este caso es el que se da en el siguiente ejemplo:

```
@keyframes desplazamiento-lento-rapido {  
0% { top: 0; }  
50% { right: 50px; }  
100% { top: 100px; }  
}
```

Aunque se haya definido la propiedad RIGHT en el paso intermedio, la animación sólo desplazará el objeto desde la posición 0 hasta la posición 100.

También es importante destacar que, aunque todos los navegadores actuales ya soportan la especificación de la W3C, podría ser necesario la utilización de prefijos específicos como son -O-TRANSITION, -MOZ-TRANSITION y -WEBKIT-TRANSITION.

## PRÁCTICA 9: DISEÑO DE UN PLANETA EN 3D

### Código del ejemplo

<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-09>



### Objetivo de la práctica

Creación de un planeta en 3D con su movimiento de rotación.

### Resultado



# Recursos para hacer la práctica

## Recurso

<https://pixabay.com/es/illustrations/el-fondo-un-chorrito-de-el-arte-de-1568716/>



### *Para qué este recurso*

Esta es la dirección de la imagen que se utilizará como textura. La versión que se insertará es la que tiene un tamaño de 640x340. Dado que es una imagen decorativa y estará asignada dentro de una propiedad BACKGROUND, no hará falta definir un texto alternativo.

## Resolución

Cuando se trata de animaciones, básicamente, se realizan dos tipos, 2D y 3D. Estas últimas son las que menos se utilizan debido a que, fundamentalmente, consumen muchos recursos y ralentizan la máquina. No obstante, pueden llegar a ser muy interesantes y espectaculares, además, de ser una forma divertida de aprender.

Dicho lo cual, lo primero que necesitaremos definir es una estructura que contenga la definición de la esfera que da forma al planeta y dos efectos complementarios de brillo e iluminación. A la sección contenadora la distinguiremos con la clase PLANET, que contendrá una capa que definirá la esfera y, dentro de esta, dos capas más con los efectos de iluminación.

Ahora, como os podréis imaginar, lo que queda es definir los estilos para cada uno de los elementos de esta estructura. El objeto PLANET será una forma cuadrada de 300x300 a la que se le asignará un valor de perspectiva con posicionamiento centrado.

Si nos fijamos en el valor de perspectiva, veremos que se ha definido un valor de 3.5 veces el valor el ancho del objeto. Ese valor está representando la distancia que hay entre el observador y la figura, es decir, lo lejos o cerca que se encuentra del objeto el usuario.

Por lo tanto, podemos deducir que, si el valor es bajo, se producirá un efecto 3D más intenso que si posee un valor alto. En otras palabras, si el valor es de 100 píxeles, se provocará un resultado más evidente que si el valor es de 1000 píxeles. Sin embargo, en este caso, como no se están realizando transformaciones en el eje Z, estas propiedades podrían haberse omitido.

Ahora vamos a definir los estilos para la bola o esfera. Esta esfera va a tener como fondo una textura de acuarela con un efecto de desenfoque de 1 píxel. Esto provocará un efecto similar a la textura de un planeta.

Además, se establecerá la propiedad TRANSFORM-STYLE para indicar que todos los elementos anidados mantengan su posición con respecto al espacio 3D.

En la capa BEFORE de este elemento, lo que definiremos es un efecto de sombra en la parte derecha a través de la propiedad BOX-SHADOW y, para darle un poco de color, en la capa AFTER, definiremos un degradado radial que irá desde abajo hacia arriba.

Ya sólo nos quedan los efectos de iluminación que comentábamos al principio, la luz focal y la ambiental. Para la luz ambiental, lo que haremos es definir una sombra de color blanco de afuera a dentro.

Para el punto de luz focal, el destello, lo que haremos es mover la capa hacia un punto concreto mediante un posicionamiento absoluto y establecer un color de fondo con una sombra difuminada de dentro hacia afuera.

Y, por último, pero no por ello menos importante, definiremos la propia animación. En este caso se trata de trasladar el fondo de la imagen desde la izquierda, hasta la derecha. El único problema que se presenta en este tipo de animaciones es que se debe encontrar el punto de unión entre ambos extremos de la imagen, no obstante, dar con el valor deseado es cuestión de paciencia.

## PRÁCTICA 10: CREACIÓN DE UN LOADER

### Código del ejemplo

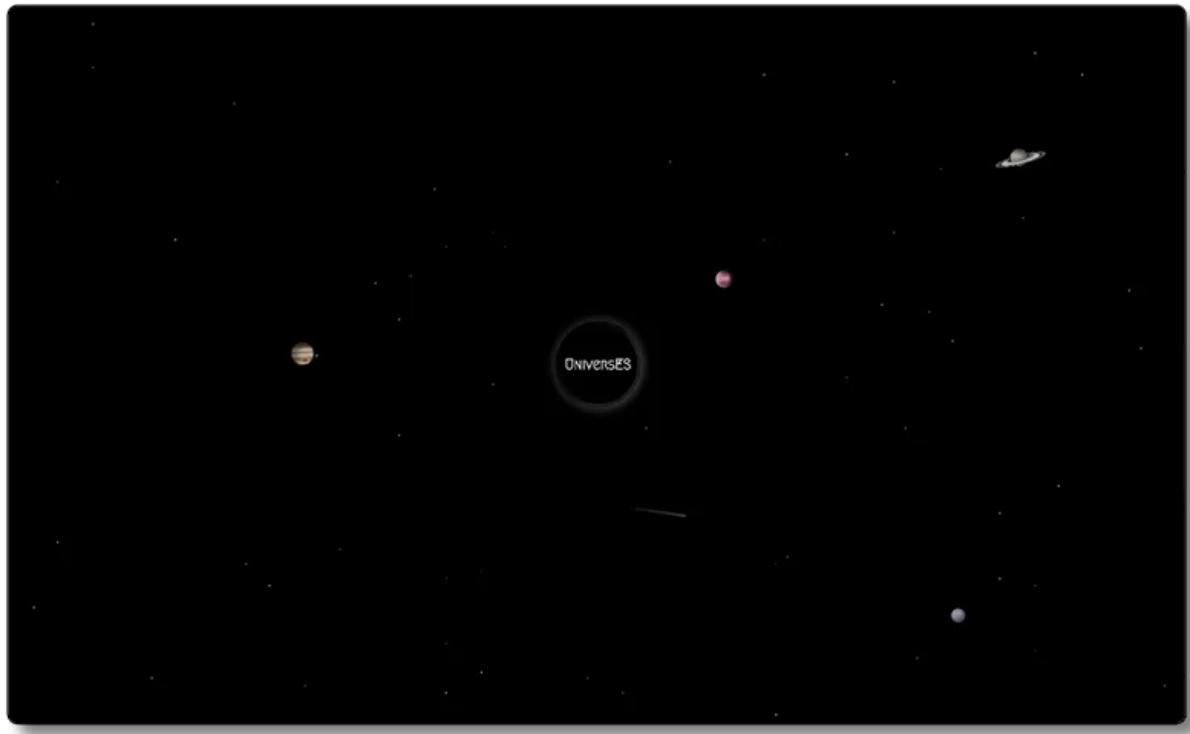
<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-10>



### Objetivo de la práctica

Creación de un planeta en 3D con su movimiento de rotación.

## Resultado



## Resolución

Esta es una práctica sencilla que requiere de unos pocos cambios para que se aplique en las diferentes páginas. Se trata de un componente de “carga en proceso”, también conocido como loader.

Un loader no es más que un elemento o componente que provoca la distracción del usuario y su mejora la experiencia sin provocar la pérdida de contexto. Es decir, que un componente de este tipo siempre debe ser diseñado con la temática del sitio o la página para que el usuario no pierda el hilo de dónde está.

Si bien los componentes de “carga en proceso” o “proceso en ejecución” se suelen crear para dar una sensación de modernidad, una imagen corporativa más sólida y una mejor experiencia de usuario, su principal objetivo no es más que mostrar una animación mientras el sistema realiza operaciones en segundo plano, reestructura la página cuando se modifica la interfaz o añadir componentes nuevos a la misma.

Cabe destacar que, este componente debe ser ligero y rápido, por lo que no suele ser una buena elección utilizar imágenes ni llamadas Ajax que puedan bloquear o ralentizar el proceso de carga. No obstante, puede haber casos en los que la pérdida de rendimiento o ralentización sean despreciables.

Por tanto, teniendo en cuenta todo esto, lo que haremos es un loader que simule una especie de eclipse lunar con unos cuantos planetas alrededor y alguna que otra estrella fugaz. Esto provocará que no se pierda el contexto de la página que estamos diseñando (UniversES) y, posiblemente, genere un efecto WOW.

## CÓDIGO HTML

Para empezar, lo primero que haremos es crear un contenedor que contendrá todos y cada uno de los elementos que conforman el componente. Esto es, un elemento que hará de imagen de fondo simulando un cielo nocturno estrellado, un elemento con que producirá un efecto de centelleo (es decir, creará la ilusión de que las estrellas brillan), un elemento con una animación que actuará como indicador de que se está produciendo un proceso en segundo plano y una etiqueta identificadora con el nombre del sitio web.

## NOTA

El siguiente código debe añadirse como primer elemento hijo del elemento BODY. La razón de ponerlo ahí es porque debe ser el primer componente en cargarse para que no provoque destellos ni parpadeos en la página durante el proceso de carga.

Como se puede apreciar, el código anterior es bastante liviano. Un contenedor que posee una serie de elementos, cada uno de los cuales, representa una parte o elemento del componente de “carga en proceso”.

Si no fijamos, la colocación de los elementos dentro del contenedor está dispuesta en un orden muy concreto, porque, de no ser así, los elementos declarados posteriormente podrían ocultar elementos que están declarados anteriormente cuando reciban los estilos. Recordemos que, CSS es “heredado”, por lo que las propiedades hijas, heredan las propiedades de los padres y, esto, puede provocar algún que otro dolor de cabeza.

## CÓDIGO CSS

En lo referente al CSS, lo primero que haremos es definir los estilos del contenedor del componente.

Seguidamente, pueden definirse los estilos en el orden que se desee, sin embargo, nosotros empezaremos por los estilos del cielo estrellado y el elemento que provoca el efecto de centelleo en las estrellas, principalmente, porque estos elementos deben estar por debajo del resto para evitar que oculten a los demás objetos.

Como se puede apreciar en el código, el elemento que hace de imagen de fondo se ha definido con el nombre de clase BG, ocupa toda la pantalla y presenta un fondo negro opaco.

La capa BEFORE, es quien “dibuja” las estrellas, una a una. Para posicionarlas, nos hemos ayudado de un algoritmo que genera valores de manera aleatoria y, el resultado lo concatena con la definición de la propiedad BOX-SHADOW. El algoritmo es el siguiente:

```
function rColor() {  
    let r = Math.floor(Math.random() * 255)  
    let g = Math.floor(Math.random() * 255)  
    let b = Math.floor(Math.random() * 255)  
    let a = (Math.floor(Math.random() * (100 - 20)) + 20) / 100;  
    return `rgba(255, 255, 255, ${a})`  
}  
  
var bs = 'box-shadow: ';  
for(var i = 0; i < 60; i++){  
    var l = Math.floor(Math.random() * (100 - 0)) + 0;  
    var t = Math.floor(Math.random() * (100 - 0)) + 0;  
    var b = Math.floor(Math.random() * (2 - 0)) + 0;  
    var s = Math.floor(Math.random() * (2 - 0)) + 0;  
    var c = rColor();  
    bs += l + 'vw ' + t + 'vh ' + b + 'px ' + s + 'px ' + c + ', '  
}  
}
```

Si analizamos el código, veremos que se va creando una cadena con valores aleatorios y que, el resultado, es una cadena con 60 puntos blancos de diferente opacidad y posicionamiento en pantalla.

La capa AFTER, es quien “oculta” las estrellas con efecto de centelleo. Para ello, lo que se ha hecho es crear un algoritmo similar al anterior que genera formas redondas degradadas de manera aleatoria y asignando, igualmente, el resultado a la propiedad BOX-SHADOW. El algoritmo que se ha utilizado es el siguiente:

```
function rColor() {  
    let r = Math.floor(Math.random() * 255)  
    let g = Math.floor(Math.random() * 255)  
    let b = Math.floor(Math.random() * 255)  
    let a = (Math.floor(Math.random() * (100 - 20)) + 20) / 100;  
    return `rgba(0, 0, 0, ${a})`  
}  
  
var bs = 'box-shadow: ';  
for(var i = 0; i < 50; i++){  
    var l = Math.floor(Math.random() * (100 - 0)) + 0;  
    var t = Math.floor(Math.random() * (100 - 0)) + 0;  
    var b = Math.floor(Math.random() * (4 - 0)) + 0;  
    var s = Math.floor(Math.random() * (4 - 0)) + 0;  
    var c = rColor();  
    bs += l + 'vw ' + t + 'vh ' + (s*2) + 'vw ' + s + 'vw #0000, '  
}  
}
```

Una vez que tenemos el fondo con las estrellas y sus respectivas capas, lo que haremos es definir el elemento que representará al componente de “carga en proceso”, puesto que, los demás, son elementos decorativos que acataremos más adelante.

Al ver el código expuesto, veremos que la clase principal define un círculo con un degradado negro que proporciona algo de profundidad y un par de sombras. La primera, genera una sombra interna más oscura y otra externa que provoca el efecto de resplandor exterior.

Aunque el nombre del sitio se ha definido como último elemento del contendor, lo definimos en esta misma parte puesto que es un elemento independiente que no tiene ninguna posible herencia que le pueda afecte, si exceptuamos los estilos generales de la página.

Y, por último, lo que haremos es definir los estilos de los planetas y de las animaciones que intervienen en todo el componente.

Para realizar el diseño de Saturno, lo que haremos es utilizar la capa principal para definir o diseñar lo que es en sí el planeta y, en la capa BEFORE lo que son los anillos.

Como información aclaratoria, podemos ver que la textura del planeta se ha realizado a través de un gradiente lineal que usa los colores similares a los que se cree que tiene Saturno y, la sombra que se la ha adjudicado es para conseguir un poco de profundidad.

Para los anillos, si nos fijamos en el código definido por la regla CSS denominada .SATURN:BEFORE, lo que se ha hecho es rotarlos tanto horizontal como verticalmente y generar unas sombras sin difuminar de diferentes anchos y color para conseguir una cierta semejanza.

Para diseñar a Júpiter, lo que haremos es utilizar la capa principal para definir o diseñar lo que es en sí el planeta y, en la capa AFTER lo que es el ojo.

Como se podrá apreciar en la definición de la regla CSS .JUPITER, para crear la textura del planeta se ha realizado un gradiente lineal que consta de 8 partes y usa los colores similares a los del planeta. Si observamos dichas partes veremos que se definen de forma que generen la ilusión de las franjas que posee dicho planeta. La sombra que se la ha adjudicado se ha orientado a la inversa que Saturno para conseguir un cierto equilibrio con los demás planetas.

Para el ojo de Júpiter, lo que haremos es definir un gradiente radial con un tamaño y posición concretas a través de posicionamiento absoluto.

Para diseñar a Mercurio, sólo utilizaremos la capa principal. Dado que lo dibujaremos a una muy pequeña escala y que su textura es a base de grises, simplemente realizaremos un gradiente linear con cuatro puntos de ruptura y una sombra de cuatro píxeles. Además, no usaremos ningún efecto de transformación, ni de saturación.

Para diseñar a Marte, sólo utilizaremos la capa principal. En este caso, para crear la textura, lo que haremos es jugar con la función de gradientes radiales y sus diversas funciones. Para las sombras, oscureceremos bastante la parte derecha y aclararemos la parte izquierda. Además, usaremos los efectos de saturación, contraste y brillo.

Para diseñar la estrella fugaz, lo que haremos es definir una figura alargada de 78x3 píxeles con un fondo correspondiente a un degradado linear de blanco a transparente que estará oculta en la parte superior izquierda de la pantalla con una inclinación de 188 grados con respecto al oeste.

Por último, sólo nos resta definir las animaciones que se utilizan en el componente de loader. La animación del loader, no deja de ser una rotación de 360 grados común. Para conseguir que esta animación se perciba algo más, le agregaremos una sombra alrededor del círculo con unos tonos blanquecinos simulando un resplandor.

Para conseguir el centelleo, se creará una animación que se basará en mover, de un lado a otro, el contenido de la capa AFTER del elemento LOADER.

Y, cómo no, la animación de la estrella fugaz lo que hará es trasladar el objeto desde una posición inicial hasta la parte inferior derecha con una opacidad del cero por ciento, es decir, totalmente transparente.

# 9

## METADATOS

Los metadatos se pueden definir de varias maneras, pero la más común y fácil de entender es la que los define como “datos acerca de los datos”. Por decirlo así, son unas declaraciones escritas que especifican una información representativa y característica sobre el contenido que va a ser representado, procesado y/o mostrado.

Los metadatos pueden proporcionar una gran cantidad de información útil a los robots y a los usuarios que lo necesiten. Pensemos que, a través de estos metadatos, se pueden especificar desde datos referentes a la apariencia, hasta datos que pueden ser utilizados o interpretados por los agentes de usuario para automatizar o ayudar en procesos y tareas concretas.

Como ya se dijo en capítulos anteriores, el elemento META especifica una información concreta mediante unos pares de nombre-valor que permiten personalizar tanto el contenido, como el tipo de información que proporciona. Esto es, normalmente, un par define el tipo de información a definir y, otro par, el contenido.

## **ATRIBUTOS ASOCIADOS A LOS METADATOS**

Aunque existen gran cantidad de metadatos posibles y pueden ser de muy diversa índole, los atributos con los que proporcionar estos datos acerca de los datos son muy concretos. De hecho, cualquier metadato puede definirse a través de los atributos NAME, HTTP-EQUIV, CHARSET, SCHEME e ITEMPROP.

### **Atributo name**

El atributo NAME especifica el nombre clave que va a asignarse al metadato y asociarse a su par nombre-valor determinado por el atributo CONTENT.

En general, podemos decir que, el atributo NAME permite definir el tipo de metadato que se va a especificar. Esto es, por ejemplo, una descripción, un autor, las palabras clave (KEYWORDS) que clasifican el documento o, el tamaño y escalamiento de la ventana gráfica.

```
<meta name="description" content="Descripción de metadatos">
<meta name="keywords" content="HTML5, HTML, XHTML">
<meta name="author" content="Pablo E. Fernández Casado">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

### **El atributo content**

El atributo CONTENT especifica el valor de los metadatos referidos a la página o a directivas pragma.

El atributo de CONTENT sólo debe establecerse si se define el atributo NAME o el atributo HTTP-EQUIV por lo que, si el metadato que se está definiendo no contiene ninguno de estos atributos, se debe obviar su declaración para evitar errores de procesado, accesibilidad y/o usabilidad web.

```
<meta name="author" content="Pablo E. Fernández Casado">
```

### **El atributo charset**

El atributo CHARSET se utiliza para especificar la codificación de caracteres en la que viene definido el documento. Entre sus valores más recurrentes podemos encontrar el valor UTF-8 y el valor ISO-8859-1.

El atributo CHARSET sólo puede ser interpretado una única vez por documento, es decir, que debe ser declarado una única vez en la página. Además, debe estar declarado al principio del documento, entre los primeros 512 bytes, dentro de la etiqueta HEAD porque, en caso contrario, puede ser ignorado.

Cabe destacar que, aunque existen multitud de posibles codificaciones, ningún agente de usuario las soporta todas, por lo que es importante que uno se informe antes de establecer una codificación de

caracteres específica.

```
<meta charset="UTF-8">
```

## El atributo http-equiv

El atributo HTTP-EQUIV se dice que es una directiva pragma y vienen a ser simulaciones de encabezados de respuesta HTTP. En otras palabras, se utiliza para comunicarse con los servidores y adaptar sus respuestas al documento.

Este atributo era utilizado antes de la versión 5 de HTML para especificar la codificación de caracteres, pero ahora ya no está permitido. A continuación, se muestra una lista con los valores actuales más representativos que admite este atributo.

- **CONTENT-SECURITY-POLICY:** Especifica una política de contenido que sirve para prevenir y disminuir algunos tipos de ataque como la inyección de datos o XSS (Cross Site Scripting) o para definir algún comportamiento que, se desea, se cumpla. Por ejemplo, si se desea que todo el contenido provenga del mismo origen, se puede establecer esta directiva a DEFAULT-SRC 'SELF'. Si, además, se desea que incluya todos sus subdominios, podría definirse la directiva a DEFAULT-SRC 'SELF' \*.COMPONENTS.COM. (Para más información puede visitarse la URL <https://developer.mozilla.org/es/docs/Web/HTTP/CSP>).
- **DEFAULT-STYLE:** Especifica la hoja de estilos por defecto o preferida. El valor de CONTENT debe ser exactamente el mismo que el definido en el atributo TITLE del elemento LINK o elemento STYLE.
- **REFRESH:** Especifica un intervalo de tiempo, tras el cual, el documento se actualiza automáticamente.

```
<meta http-equiv="refresh" content="60">
```

## El atributo scheme

El atributo SCHEME no está soportado por HTML5, no obstante, antes se utilizaba para especificar el esquema que se debía emplear para interpretar el valor de una propiedad. En otras palabras, especificaba el esquema de formato o URI que se debía emplear para interpretar el valor del atributo CONTENT.

```
<meta name="date" content="01-01-2020" scheme="DD-MM-YYYY">
<meta name="identifier" content="0-2345-6634-6" scheme="ISBN">
```

## PRINCIPALES METADATOS HTML

Los metadatos no sólo son importantes porque aporten un poco más de semántica web a los documentos, también son muy interesantes porque facilitan la búsqueda de resultados y ayudan a mejorar el Posicionamiento SEO.

El Posicionamiento SEO sirve para mejorar la visibilidad de los documentos web en los resultados orgánicos de los motores búsqueda, lo que, por ejemplo, se traduce más visitas sin tener que invertir dinero extra.

## Allow

Permite especificar los métodos HTTP soportados o habilitados en el servidor.

```
<meta http-equiv="allow" content="HEAD,GET" />
```

Entre los posibles valores que se pueden especificar en el atributo CONTENT están HEAD, GET, PUT, DELETE y OPTIONS.

## Application-Name

El metadato APPLICATION-NAME especifica el nombre de la aplicación web a la que se está accediendo. Puede ser cualquier valor de texto que sea representativo y sólo puede definirse una única vez.

```
<meta name="application-name" content="Servicio Web de Islavisual.com" />
```

En lo referente a su importancia para mejorar el Posicionamiento SEO, se encuentra en un nivel MEDIO.

## Author

El metadato AUTHOR especifica el autor que creó el documento. Puede ser cualquier valor de texto y sólo puede definirse una única vez.

```
<meta name="author" content="Pablo E. Fernández Casado">
```

En lo referente a su importancia para mejorar el Posicionamiento SEO, se encuentra en un nivel ALTO.

## Cache-Control

Permite especificar si se debe almacenar la página en caché y, de ser así, en cuál de ellas.

```
<meta http-equiv="cache-control" content="no-cache" />
```

Entre los posibles valores que admite esta directiva tenemos:

Valor	Significado / Descripción
<b>public</b>	Indica que se almacene en cache pública.
<b>private</b>	Indica que se almacene sólo en la cache privada.
<b>no-cache</b>	Indica que no se almacene en caché.
<b>no-store</b>	Indica que se almacene en cache, pero sin archivarse

En lo referente a su importancia para mejorar el Posicionamiento SEO, no tiene ningún efecto.

## Content-Disposition

Permite especificar qué contenido debe enviarse al servidor y cómo debe procesarse o mostrarse. Esto puede ser útil cuando se desea solicitar la descarga de un documento desde el servidor sin que se abra en el agente de usuario.

```
<meta http-equiv="content-disposition" content="inline;filename=html5.pdf">
```

Entre los posibles valores que admite esta directiva tenemos:

Atributo	Significado / Descripción
<b>attachment</b>	Indica que la solicitud sea procesada como una descarga.
<b>inline</b>	Indica que la solicitud sea procesada por el navegador.
<b>filename</b>	Indica el nombre del archivo o documento.
<b>creation-date</b>	Indica la fecha de creación del documento.
<b>modification-date</b>	Indica la fecha de actualización del documento.
<b>read-date</b>	Indica la fecha de apertura del documento.
<b>size</b>	Indica el tamaño del documento o archivo.

En lo referente a su importancia para mejorar el Posicionamiento SEO, no tiene ningún efecto.

## Content-Encoding

Permite especificar el tipo de compresión, codificación o encriptación utilizado en los datos devueltos.

```
<meta http-equiv="content-encoding" content="gzip" />
```

Entre los posibles valores que admite esta directiva tenemos:

Atributo	Significado / Descripción
<b>gzip</b>	Especifica que no se comprima utilizando el algoritmo de compresión de datos sin pérdida que usa el algoritmo adaptativo Lempel-Ziv-Coding (LZ77).
<b>compress</b>	Especifica que no se comprima utilizando el algoritmo de compresión de datos sin pérdida que usa el algoritmo adaptativo Lempel-Ziv-Welch (LZW).
<b>deflate</b>	Especifica que no se comprima utilizando el algoritmo de compresión de datos sin pérdida que usa una combinación de codificación LZSS y Huffman.
<b>identity</b>	Especifica que no se comprima ni modifique

En lo referente a su importancia para mejorar el Posicionamiento SEO, no tiene ningún efecto.

## Content-Language

Permite especificar el lenguaje utilizado en el documento, es decir, indica el idioma de los usuarios al que va dirigido el documento. Los valores que puede contener este atributo son los contemplados por la

```
<meta http-equiv="content-language" content="es-ES" />
```

Como se puede apreciar en la ilustración anterior, el código de idioma va referenciado a través de dos caracteres en minúsculas que indican el idioma genérico y dos caracteres en mayúsculas que indican la variación o dialecto utilizado.

Cabe destacar que, en lo referente a su importancia para mejorar el Posicionamiento SEO, se encuentra en un nivel BAJO y que, en general, suele ser un único descriptor de lenguaje, aunque puede contener varios valores.

## **Content-Script-Type**

Permite especificar el lenguaje de scripting utilizado por defecto en el documento. Los posibles valores que puede contemplar este encabezado o directiva están dispuestos en la lista de IANA Media Types.

```
<meta http-equiv="content-script-type" content="text/javascript" />
```

En general, el valor de esta directiva no suele cambiar, ya que el JavaScript es prácticamente el único que está actualmente en uso en todo sistema o aplicación web y no tiene ninguna influencia en el posicionamiento en buscadores.

## **Copyright**

Especifica que el contenido está protegido y puede ser cualquier valor de texto representativo y, en ocasiones, especifica el nombre del propietario que posee los derechos.

```
<meta name="copyright" content="Copyright 2010 a 2020">
```

No tiene ninguna influencia en el posicionamiento en buscadores.

## **Default-Style**

Permite especificar las hojas de estilos a utilizar. Esto es útil cuando se disponen de varias definiciones de estilo en el mismo documento.

```
<link rel="stylesheet" type="text/css"
      href="styles.css" title="por-defecto">
<link rel="alternate stylesheet" type="text/css"
      href="high-contrast.css" title="alto-contraste">
<link rel="alternate stylesheet" type="text/css"
      href="low-contrast.css" title="bajo-contraste">
<meta http-equiv="default-style" content="por-defecto" />
```

La propiedad CONTENT es la que selecciona, por defecto, todas las hojas de estilo por defecto a través del atributo TITLE, por lo que, si lo que se definen varias hojas de estilo, las asignará todas.

El único problema que se presenta aquí es que, si, más tarde, el usuario decidiese cambiar el estilo por defecto a cualquier otro de los declarados, debería realizarse a través de JavaScript u otro lenguaje de scripting similar. Un ejemplo de esta operación podría ser:

```
var meta = document.head.querySelector("meta[http-equiv='default-style']");
meta.content = "alto-contraste";
```

## Description

El metadato DESCRIPTION especifica una descripción más o menos corta que indica el objetivo o tema del documento o aplicación web. Puede ser cualquier valor de texto de entre 50 y 160 caracteres de longitud y sólo puede definirse una única vez.

```
<meta name="description"
content="Blog de creación de aplicaciones y páginas web ejemplo.com">
```

El metadato DESCRIPTION es, posiblemente, uno de los metadatos más importantes puesto que los directorios de páginas y motores de búsqueda se basan en él para establecer y mostrar los resultados a los usuarios. Por esta razón, en lo referente a su importancia para mejorar el Posicionamiento SEO, se encuentra en un nivel ALTO.

## Date

Permite especificar la fecha y hora en que fue publicada el documento.

```
<meta http-equiv="date" content="Friday, 13-Mar-20 14:15:00 GMT" />
<meta http-equiv="date" content="Thu, 12-Mar-20 14:15:00 GMT" />
```

Como se puede apreciar en la ilustración anterior, la fecha debe estar especificada con ajuste de zona horaria GMT.

## Expires

Permite especificar la fecha y hora en que el documento deberá ser recargado y almacenado de nuevo en la caché. Es decir, especifica la fecha de validez hasta la cual el documento será cargado desde la caché. Una vez pasada esa fecha, deberá volver a cachearse con una fecha válida.

```
<meta http-equiv="expires" content="Friday, 13-Mar-20 14:15:00 GMT" />
<meta http-equiv="expires" content="Thu, 12-Mar-20 14:15:00 GMT" />
```

Como se puede apreciar en la ilustración anterior, la fecha debe estar especificada con ajuste de zona horaria GMT.

## Generator

Especifica el nombre del software o herramienta con el que se ha creado el documento. Puede ser cualquier valor de texto que permita identificar el software utilizado para generar el documento y sólo puede definirse una única vez.

```
<meta name="generator" content="Visual Studio Code 1.32">
```

En lo referente a su importancia para mejorar el Posicionamiento SEO, se encuentra en un nivel ALTO.

## **Keywords**

Especifica el conjunto de palabras que resultan relevantes para describir el documento. Puede ser cualquier valor de texto que no supere los 256 caracteres y, como mínimo 5 palabras. Cada palabra dentro de este metadato debe estar separado por coma.

```
<meta name="keywords" content="HTML5, HTML, XHTML, CSS, SVG">
```

Aunque sigue teniendo importancia, este metadato ya no tiene tanta relevancia como hace años. Por ello, en lo referente a su importancia para mejorar el Posicionamiento SEO, se encuentra en un nivel BAJO.

De todas formas, antes de establecer el contenido de KEYWORDS en una página, se debe realizar un estudio sobre las palabras clave. Esto es casi una obligación porque, de no hacerlo, podríamos estar confiando más en la suerte que en los datos y su análisis.

Un posible método para asignar las palabras clave es pensar en qué consiste, o de que va, el contenido de la página que se está escribiendo. También se puede recurrir a terceras personas y preguntarles de que creen ellos que va el artículo o página en pocas palabras.

También hay que tener en cuenta que, una palabra clave no es mejor porque tenga un mayor volumen de búsqueda. Esto es así porque, las grandes empresas pueden ser una fuerte competencia y no provocar el fruto esperado. A veces, una palabra con menor volumen de aparición o búsqueda puede ayudar a posicionar las páginas de forma más rápida.

También es interesante que se incluya la palabra más importante en la etiqueta TITLE de HTML. Esto es bueno porque, tanto los robots, como los usuarios situarán mejor el contexto del contenido de la página.

Por último, hay que destacar que Google Analytics y Ahrefs disponen de herramientas para ayudar a la buena elección de palabras clave.

## **Last-modified**

Permite especificar fecha y hora en la que se modificó el documento por última vez.

```
<meta http-equiv="last-modified" content="Monday, 9-Mar-20 14:15:00 GMT" />
```

Como se puede apreciar en la ilustración anterior, la fecha debe estar especificada con ajuste de zona horaria GMT.

## **Location**

Permite especificar una nueva dirección de ubicación a la que ser redirigido.

```
<meta http-equiv="location" content="10; new-index.html" />
```

El valor numérico inicial dentro de la propiedad CONTENT, establece el número de segundos que tardará en realizar el redireccionamiento. En este caso, se redirigirá a new-index.html pasados 10

segundos.

## Rating

Aunque esta directiva no se suele utilizar mucho, especifica el tipo de público para el que va dirigido el contenido web. Sus posibles valores son GENERAL, MATURE, RESTRICTED, 14 YEARS y SAFE FOR KIDS.

```
<meta name="rating" content="General">
```

Cabe destacar que, si utiliza esta directiva de forma incorrecta, los motores de búsqueda podrían marcar el contenido web como inapropiado e, incluso, eliminar todo registro de él. Por esta razón, no influye de ninguna forma en los motores de búsqueda a menos, eso sí, que utilice de una manera malintencionada.

## Refresh

Permite especificar el número de segundos que deben pasar para volver a recargar la página.

```
<meta http-equiv="refresh" content="10; URL=./new-index.html" />
```

Como se puede apreciar en la ilustración anterior, adicionalmente, permite especificar una dirección que es hacia dónde irá transcurridos los segundos descritos inicialmente.

## Robots, MSNbot y Googlebot

Especifica cuál debe ser el comportamiento de rastreo e indexación de los motores de búsqueda ante el documento.

```
<meta name="robots" content="noindex,nofollow">
```

Si el valor del atributo NAME es robots, se aplicará la directiva a todos los motores de búsqueda. No obstante, si el valor del atributo NAME es específico, sólo se aplicará la directiva al motor de búsqueda al que se refiera el identificador.

```
<meta name="googlebot" content="index,follow">
<meta name="msnbot" content="index,nofollow">
<meta name="bingbot" content="noindex">
<meta name="slurp" content="noodp">
```

Si nos fijamos en la ilustración anterior, vemos que, en este caso, se hace referencia a los robots de Google, Microsoft, Bing y Yahoo. Por ejemplo, en el caso del GOOGLEBOT, se indica que se indexe el contenido y se sigan los enlaces, no obstante, en el caso del MSNBOT, se indica que se indexe el contenido, pero que no se sigan los enlaces.

Entre los posibles valores que admite esta directiva tenemos:

Valor	Significado / Descripción
<b>index</b>	Especifica que se puede indexar el documento o página.
<b>noindex</b>	Especifica que no se puede indexar el documento o página.

<b>follow</b>	Especifica que se sigan los enlaces que se encuentren en el documento o página.
<b>nofollow</b>	Especifica que no se sigan los enlaces que se encuentren en el documento o página.
<b>noodp</b>	Evita que los motores de búsqueda utilicen el título y descripción provenientes del directorio público Open Directory Project (ODP o DMOZ), si existe y procede. Este valor sólo es aplicable a Google, Bing y Yahoo.
<b>noarchive</b>	Especifica a los motores de búsqueda que no se cachee el documento o página. Este valor sólo es aplicable a Google, Bing y Yahoo.
<b>nosnippet</b>	Especifica al robot de Google (Googlebot) que el documento no debe ser mostrado en los resultados de búsqueda.
<b>noimageindex</b>	Especifica al robot de Google (Googlebot) que el documento no debe estar referida de un índice de imágenes.
<b>noydir</b>	Evita que el robot de Yahoo utilice el título y descripción provenientes del directorio Yahoo Directory Project, si existe y procede.
<b>nocache</b>	Especifica al robot de Bing no se cachee el documento o página. Es idéntico que NOARCHIVE, pero sólo para Bing.

En referencia a este tema, cabe destacar que, en el año 2007, se introdujo un valor de atributo denominado ROBOTS-NOCCONTENT que le indica al robot de Yahoo que el contenido no debe ser rastreado ni indexado. Esto se realiza a través del atributo CLASS y, aunque no es una metaetiqueta, su declaración hace que funcione como si lo fuese, permitiendo especificar qué contenidos se deben ignorar. A continuación, se muestran un par de ejemplos:

```
<section class="robots-nocontent">
<!-- Contenido a ignorar -->
</section>
<footer class="robots-nocontent">
<!-- Contenido a ignorar -->
</footer>
```

## Viewport

El término VIEWPORT significa ventana y, en el contexto páginas web, se refiere al área útil de la pantalla donde se visualizará el contenido.

En líneas generales, se puede afirmar que el tamaño de una página o documento renderizado no se corresponde con el tamaño del área visible de la pantalla. Por este motivo, cuando no se declara esta directiva, los contenidos pueden mostrarse con las barras de desplazamiento horizontal y vertical.

Pensemos, por ejemplo, en cómo puede verse una página, que está diseñada para ser visualizada en pantallas grandes, en un dispositivo móvil. Si esto sucede e intentamos visualizar un contenido pensado para ser mostrado en resoluciones superiores a 1280 píxeles, en un dispositivo que sólo dispone de 360 píxeles, el resultado será que, el usuario, tendrá que desplazarse tanto vertical, como horizontalmente, para acceder a todo el contenido.

Cierto es que este problema, en parte, podría evitarse no utilizando elementos con ancho fijo ni medidas absolutas como son los píxeles, pero aun así, con todo y con ello, el resultado no sería completamente usable y accesible.

Pues bien, el VIEWPORT o área útil dónde se mostrará el documento, cambia en función de si estamos en un dispositivo de escritorio o en un dispositivo móvil. Si el navegador o herramienta de asistencia se ejecutan en un dispositivo de escritorio, el tamaño de la pantalla coincidirá con el tamaño del área útil para renderizar el documento, no así, cuando se ejecuta en un dispositivo móvil.

Cuando el navegador o herramienta de asistencia se ejecutan en un dispositivo móvil, el VIEWPORT no se corresponde con el tamaño real de la pantalla, sino con el espacio que la aplicación de software está emulando. Por ejemplo, en un dispositivo de Apple, aunque el ancho de la pantalla sea de 320 píxeles, en realidad se pueden estar emulando 980, porque el sistema realiza una serie de extrapolaciones para que se ajuste al espacio visible y se muestre el contenido de la mejor forma posible.

Todo esto, entre otras razones, es porque los dispositivos móviles poseen una densidad en píxeles muy diferente a las pantallas de escritorio. Mientras que en las pantallas de sistemas de escritorio la proporción de densidad en píxeles es de 1:1, en los dispositivos móviles puede llegar a ser muy superior.

## NOTA

La densidad de píxeles es un valor que se calcula a partir de los valores de resolución y tamaño de la pantalla. Este valor especifica el número de píxeles que es posible mostrarse en una pulgada (2,54 cm).

Dicho esto, la directiva VIEWPORT especifica cuál debe ser el tamaño del área útil de pantalla y los valores posibles de escalado. Estos valores de escalado indican el nivel de escalado inicial (zoom) y si se puede o no ampliar o reducir.

```
<meta name="viewport"  
content="width=device-width, initial-scale=1, user-scalable=no">
```

Entre los posibles valores que admite esta directiva tenemos:

Valor	Significado / Descripción
<b>height</b>	Especifica el alto, en píxeles, de la ventana gráfica. Permite definir cualquier valor entero positivo o la palabra clave DEVICE-HEIGHT, que indica que se utilice el alto de la pantalla en píxeles con un nivel de escalado del 100%.
<b>initial-scale</b>	Especifica el nivel de escalado inicial que se debe utilizar. Esto es, indica la proporción que existe entre el ancho o alto del dispositivo y el tamaño de la ventana gráfica. Permite definir cualquier valor, entero o decimal, comprendido entre 0 y 10.
<b>maximum-scale</b>	Especifica el nivel de escalado máximo que se puede utilizar, teniendo que ser, este, igual o superior al determinado por la propiedad MINIMUM-SCALE. Permite definir cualquier valor, entero o decimal, comprendido entre 0 y 10.
<b>minimum-scale</b>	Especifica el nivel de escalado mínimo que se puede utilizar, teniendo que ser, este, igual o menor al determinado por la propiedad MAXIMUM-SCALE. Permite definir cualquier valor, entero o decimal, comprendido entre 0 y 10.

<b>user-scalable</b>	Especifica si el usuario puede o no realizar cambios en el nivel de escalado (puede hacer zoom) en la página. Sus posibles valores son YES y NO. Por defecto el valor es YES.
<b>width</b>	Especifica el ancho, en píxeles, de la ventana gráfica. Permite definir cualquier valor entero positivo o la palabra clave DEVICE-WIDTH, que indica que se utilice el ancho de la pantalla en píxeles con un nivel de escalado del 100%.

Cabe destacar que, si establecemos esta directiva en un documento o página web que no presenta un diseño adaptativo y lo ejecutamos en un dispositivo móvil, el efecto que se producirá será que todo se hará mucho más pequeño para que entre en el área útil o ventana gráfica. Por esta razón, esta directiva debe estar declarada y procesada en combinación con las consultas de medios de CSS (Media Queries) de manera eficiente, para que el contenido sea usable, legible y accesible.

## X-UA-compatible

Permite especificar el modo de compatibilidad del documento. Este valor de atributo sólo es aplicable para Chrome e Internet Explorer.

```
<meta http-equiv="x-ua-compatible" content="IE=11, chrome=1" />
```

En la ilustración anterior, se está indicando que Internet Explorer debe ejecutarse utilizando el modo de compatibilidad establecido en su versión 11 y que el marco de Google Chrome debe comenzar si el usuario lo tiene instalado

En la variable de IE, si se especifica el valor EDGE, lo que se está solicitando es que IE debe ejecutarse en el modo de compatibilidad más actual disponible. Esto es, que se ejecute con la versión más moderna de Internet Explorer.

```
<meta http-equiv="x-ua-compatible" content="IE=Edge, chrome=1" />
```

## DUBLIN CORE

Dublin Core es un modelo de metadatos definido por la Dublin Core Metadata Initiative y compuesto por 15 definiciones semánticas descriptivas. Las normas que sigue este modelo son la ISO-15836, y la norma NISO Z39.85-2012.

Al igual que varios de los metadatos genéricos, los metadatos de la especificación Dublin Core se establecen a través de los atributos NAME y CONTENT. El atributo NAME, debe llevar el prefijo “DC.” antes del nombre de metadato que se va a describir. Además, su declaración es opcional, puede repetirse y aparecer en cualquier orden.

```
<meta name="DC.title"
content="Curso de creación de páginas web" />
```

El modelo Dublin Core presenta dos niveles, el simple y el cualificado. El simple contempla 15 elementos que permiten describir el recurso o documento. El cualificado resulta ser una extensión del simple que hace que puedan definirse nuevos términos o especificaciones de datos sobre cada uno de los

elementos primarios del nivel simple, por lo que son más restrictivos. A continuación, se muestran unos ejemplos de Dublin Core cualificado:

```
<meta name="DCTERMS.created" content="2020-02-14" />
<meta name="DCTERMS.valid" content="2020-02-14" />
<meta name="DCTERMS.issued" content="2020-03-10" />
```

Aunque pueden utilizarse en HTML, como es el ejemplo de la ilustración anterior, el modelo Dublin Core se utiliza generalmente bajo esquemas XML y RDF. La forma de representarlo bajo estos lenguajes de anotación es:

```
<dc:title>Curso de creación de páginas web</dc:title>
<dcterms:created>2020-02-14</dcterms:created>
```

Entre los principales nombres que admite esta especificación se pueden destacar 15 elementos.

## **Creator**

El elemento CREATOR especifica la persona, empresa u organización responsable de la creación del contenido del documento o recurso.

```
<meta name="DC.creator" content="Fernández Casado, Pablo E." />
<meta name="DC.creator"
      content="Universidad Autónoma de Madrid de Telecomunicaciones" />
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto que represente a una entidad creadora o autora. Esto es, si el valor va referido a una persona, debe estar descrito en formato invertido, es decir, primero los apellidos y después el nombre.

Si el valor va referido a una empresa u organización, se debe intentar representar todos los niveles de jerarquía del modo más natural posible y en orden descendente.

## **Contributor**

El elemento CONTRIBUTOR especifica la persona, empresa u organización responsable de la contribución o colaboración de contenido del recurso.

```
<meta name="DC.contributor" content="Fernández Casado, Pablo E." />
<meta name="DC.contributor" content="UAM, Dpto. de telecomunicaciones" />
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto que represente a una entidad supervisora, técnica, recolectora, editora o promotora, pero con cuidado de no confundirlo con una entidad creadora o autora.

Si el valor va referido a una persona, debe estar descrito en formato invertido, es decir, primero los apellidos y después el nombre.

Si el valor va referido a una empresa u organización, se debe intentar representar todos los niveles de jerarquía del modo más natural posible y en orden descendente.

## Coverage

El elemento COVERAGE especifica las características espaciales y temporales del contenido intelectual del documento o recurso. Estas características pueden venir referidas al alcance y/o ámbito del documento o recurso, a la aplicación espacial del documento o recurso o, a la jurisdicción bajo la cual el recurso es relevante.

```
<meta name="DC.coverage" content="2010-2012" />
<meta name="DC.coverage" content="Madrid, ES" />
<meta name="DC.coverage" content="Siglo XX" />
<meta name="DC.coverage" content="Sureste de África" />
```

En lo referente a los posibles valores admitidos puede ser cualquier valor de texto que haga referencia a un lugar mediante el nombre o las coordenadas de latitud y longitud, a una fecha o rango de fechas con nombre o a una jurisdicción como pueda ser una entidad administrativa o lugar geográfico.

No obstante, si el valor es textual, se recomienda seleccionar valores controlados de tesauros estándar como *Thesaurus of Geographic Names* (TGN) o *Geographic Names Information System* (GNIS).

Si el valor va referido, exclusivamente, a las características espaciales del contenido intelectual se puede utilizar el término SPATIAL.

```
<meta name="DCTERMS.spatial"
content="Longitud: 03°42'9.22" Latitud: N40°24'59.4" />
```

Si el valor va referido, exclusivamente, a las características de tiempo del contenido intelectual se puede utilizar el término TEMPORAL.

```
<meta name="DCTERMS.temporal" content="Siglo XXI" />
<meta name="DCTERMS.temporal" content="Era Mesozoica" />
```

## Date

El elemento DATE representa un valor, a veces, ambiguo debido a que especifica la fecha asociada a un documento o recurso. No obstante, esta fecha suele asociarse al evento de creación o producción.

```
<meta name="DC.date" content="2020-02-14" />
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de fecha en formato Big Endian o ISO 8601, es decir, en formato YYYY-MM-DD.

Si la fecha va referida a la creación del documento o recurso, se puede utilizar el término CREATED.

```
<meta name="DCTERMS.created" content="2020-02-14" />
```

Si la fecha va referida al periodo de validez del documento o recurso o únicamente es relevante en una fecha en concreto, se puede utilizar el término VALID.

```
<meta name="DCTERMS.valid" content="2021-03-01" />
```

Si la fecha va referida a cuándo estará disponible el documento o recurso se puede utilizar el término AVAILABLE.

<meta name="DCTERMS.available" content="2020-03-10" />

Si la fecha va referida a cuándo se publicó formalmente el documento o recurso se puede utilizar el término ISSUED.

<meta name="DCTERMS.issued" content="2020-02-15" />

Si la fecha va referida a cuándo se modificó por última vez el documento o recurso se puede utilizar el término MODIFIED.

<meta name="DCTERMS.modified" content="2020-02-15" />

Si la fecha va referida a cuándo se aceptó formalmente el documento o recurso se puede utilizar el término DATEACCEPTED.

<meta name="DCTERMS.dateAccepted" content="2020-02-15" />

Si la fecha va referida a una declaración de copyright asociada al documento o recurso se puede utilizar el término DATECOPYRIGHTED.

<meta name="DCTERMS.dateCopyrighted" content="2020-02-16" />

Si la fecha va referida a cuándo se realizó la entrega del documento o recurso se puede utilizar el término DATESUBMITTED.

## Description

El elemento DESCRIPTION especifica la descripción del contenido del documento o recurso.

<meta name="DC.description" content="Curso de creación de páginas web que describe los principales componentes necesarios y diferentes tecnologías que se pueden utilizar en sistemas y aplicaciones web. Todo ello permitirá a todo aquel que lo desee llevar a cabo un proceso de diseño de forma clara y sencilla" />

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto representativo como un resumen, listado de contenidos, referencia de una representación gráfica, notas explicativas o una descripción de texto libre.

## Format

El elemento FORMAT especifica el formato del fichero digital, el medio físico o las dimensiones del recurso. Esto no incluye el soporte físico original.

<meta name="DC.format" content="application/pdf" />

<meta name="DC.format" content="video/mpeg" />

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto que venga respaldado por los valores de IANA Media Types. Estos valores pueden consultarse en <https://www.iana.org/assignments/media-types/media-types.xhtml>

Si el valor va referido al tamaño o duración del recurso se puede utilizar el término EXTEND.

<meta name="DCTERMS.extent" content="1.43 MB" />

<meta name="DCTERMS.extent" content="14 minutos" />

Si el valor va referido al medio se puede utilizar el término MEDIUM.

```
<meta name="DCTERMS.medium" content="Din A4 80 gr" />
```

## Identifier

El elemento IDENTIFIER especifica una referencia incuestionable e irrefutable hacia un documento o recurso dentro de un contexto determinado.

```
<meta name="DC.identifier"  
content="https://dublincore.org/specifications/dublin-core/" />
```

En lo referente a los posibles valores admitidos, lo frecuente es que se refiera a través de un valor numérico o alfanumérico que se rija por algún sistema formal. Entre los sistemas formales más utilizados están el Identificador de Recursos Uniforme (URI), el Localizador de Recursos Uniforme (URL), el Identificador de Objetos Digitales (DOI) o Identificador Único Standard Internacional de Libros (ISBN).

Ahora bien, si el valor va referido o es una referencia biográfica se puede utilizar el término BIBLIOGRAPHIC CITATION. Este tipo de referenciación se debe intentar establecer de la forma más precisa posible usando algún sistema de identificación único como Autor-Año-Página, Libro-Capítulo-Versículo-Versión, etcétera.

```
<meta name="DCTERMS.bibliographicCitation"  
content="Job 13:2 – Biblia Reina Valera 1960" />  
<meta name="DCTERMS.bibliographicCitation"  
content="Nietzsche, 1873, p56-57" />
```

## Language

El elemento LANGUAGE especifica el idioma o lenguaje utilizado para describir el contenido intelectual del documento o recurso.

```
<meta name="DC.language" content="spa" />
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto de tres caracteres que esté avalado por la norma ISO 639-3 o, en su defecto, se puede utilizar el estándar RFC 3066 en conjunción con la ISO 639-2 que define las etiquetas de tres letras primarias para lenguaje, con subetiquetas opcionales.

Los valores válidos para la ISO 639-3 pueden consultarse a través de la dirección [https://iso639-3.sil.org/code\\_tables/639/data](https://iso639-3.sil.org/code_tables/639/data).

## Publisher

El elemento PUBLISHER especifica la persona, empresa u organización que representa a la figura propietaria intelectual, responsable de publicar el documento.

```
<meta name="DC.publisher" content="Fernández Casado, Pablo E." />  
<meta name="DC.publisher"
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto que represente a una entidad propietaria. Esto es, si el valor va referido a una persona, debe estar descrito en formato invertido, es decir, primero los apellidos y después el nombre.

Si el valor va referido a una empresa u organización, se debe intentar representar todos los niveles de jerarquía del modo más natural posible y en orden descendente.

## **Relation**

El elemento RELATION especifica una referencia relacionada con el documento o recurso.

```
<meta name="DC.relation" content="ISBN: 978-8499647357" />
<meta name="DC.relation"
content="https://es.wikipedia.org/wiki/Metadatos" />
```

En lo referente a los posibles valores admitidos, lo frecuente es que se refiera a través de un valor numérico o alfanumérico que se rija por algún sistema formal. Entre los sistemas formales más utilizados están el Identificador de Recursos Uniforme (URI), el Localizador de Recursos Uniforme (URL), el Identificador de Objetos Digitales (DOI) o Identificador Único Standard Internacional de Libros (ISBN).

Si el valor es una versión de, o tiene como versión a, se puede utilizar el término ISVERSIONOF.

```
<meta name="DCTERMS.isVersionOf" content="El Quijote" />
<meta name="DCTERMS.hasVersion" content="La guerra de los mundos" />
```

Si el valor es reemplazado por o, es un reemplazo de, se puede utilizar el término ISREPLACEDBY o el término REPLACES.

```
<meta name="DCTERMS.isReplacedBy" content="La guerra de los mundos" />
<meta name="DCTERMS.replaces" content="La guerra de los mundos" />
```

Si el valor es parte de, o tiene parte de, se puede utilizar el término ISREQUIREDBY o el término REQUIRES.

```
<meta name="DCTERMS.isPartOf" content="La guerra de los mundos" />
<meta name="DCTERMS.hasPart" content="La guerra de los mundos" />
```

Si el valor es requerido por o, es un requerimiento de, se puede utilizar el término ISREQUIREDBY o el término REQUIRES.

```
<meta name="DCTERMS.isRequiredBy" content="La guerra de los mundos" />
<meta name="DCTERMS.requires" content="La guerra de los mundos" />
```

Si el valor está citado a, o es una cita a, se puede utilizar el término ISREFERENCEDBY o el término REFERENCES.

```
<meta name="DCTERMS.isReferencedBy" content="La guerra de los mundos" />
<meta name="DCTERMS.references" content="La guerra de los mundos" />
```

Si el valor está en formato de, o tiene formato de, se puede utilizar el término ISFORMATOF o el término HASFORMAT.

```
<meta name="DCTERMS.isFormatOf" content="URI" />
<meta name="DCTERMS.hasFormat" content="ISBN" />
Si el valor es conforme a, se puede utilizar el término CONFORMSTO.
<meta name="DCTERMS.conformsTo" content="W3C WAI ARIA" />
<meta name="DCTERMS.conformsTo" content="WCAG 2.1" />
<meta name="DCTERMS.conformsTo" content="IANA Media Types" />
```

## Rights

El elemento RIGHTS especifica los derechos de propiedad intelectual hacia o sobre el documento o recurso.

```
<meta name="DC.rights"
content="Pablo Enrique Fernández Casado, 2018" />
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto que describa la gestión de derechos para el recurso, referencia a un documento interno o externo que provea tal información o una interpretación libre que proporcione, de forma adecuada, la información sobre los derechos legales que afectan a la utilización del documento o recurso.

## Source

El elemento SOURCE especifica una referencia de la cual deriva el documento o recurso descrito.

```
<meta name="DC.source" content="ISBN: 978-8499648712" />
```

En lo referente a los posibles valores admitidos, lo frecuente es que se refiera a través de un valor numérico o alfanumérico que se rija por algún sistema formal. Entre los sistemas formales más utilizados están el Identificador de Recursos Uniforme (URI), el Localizador de Recursos Uniforme (URL), el Identificador de Objetos Digitales (DOI) o Identificador Único Standard Internacional de Libros (ISBN).

## Subject

El elemento SUBJECT especifica el asunto o tema sobre el que va el contenido del documento o recurso.

```
<meta name="DC.subject"
content="Descripción de metadatos de Dublin Core">
```

En lo referente a los posibles valores admitidos, debe seguir las mismas normas que el metadato KEYWORDS, aunque puede ser cualquier valor de texto expresado en forma de palabras clave, tabla de contenidos o códigos de clasificación que describan el tema sobre el que va el documento.

Si lo que se está describiendo es una tabla de contenidos se deben establecer todos los niveles jerárquicos en orden separados por punto y coma.

```
<meta name="DCTERMS.tableofContents"
```

Si lo que se está describiendo es un resumen debe ser un texto formal que describa el asunto o tema sin extenderse demasiado.

```
<meta name="DCTERMS.abstract" content="Curso de creación de páginas web que  
describe los principales componentes necesarios y diferentes tecnologías que se  
pueden utilizar en sistemas y aplicaciones web. Todo ello permitirá a todo aquel  
que lo desee llevar a cabo un proceso de diseño de forma clara y sencilla" />
```

## Type

El elemento TYPE especifica la naturaleza, dominio o género del documento o recurso.

```
<meta name="DC.type" content="Artículo Formativo" />
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto que identifique el tipo o naturaleza del documento o recurso. Por ejemplo, si se está describiendo un estilo musical, lo que se debe especificar es SONIDO y no el género. Ahora bien, Si se está describiendo algo como un libro o un artículo, lo que se puede establecer es su tipo de uso, es decir, si es para uso interno o externo, o su género, es decir, si es de tipo novela, bibliografía, formativo, etcétera.

## Title

El elemento TITLE especifica el nombre del recurso o documento para darlo a conocer.

```
<meta name="DC.title" content="Curso de creación de páginas web" />
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto que respete las normas de ortografía y gramática. Esto es, no se debe capitalizar la descripción del título, a no ser que sea un nombre propio, y se deben respetar los signos de puntuación.

Si el título que se desea describir es alternativo o complementario se puede utilizar el término ALTERNATIVE.

```
<meta name="DCTERMS.alternative"  
content="HTML5, CSS3, usabilidad y accesibilidad web" />
```

Sus valores admitidos son los mismos que para el elemento TITLE, no obstante, sólo debe estar presente si también lo está el elemento TITLE.

## OPEN GRAPH

El protocolo Open Graph son unos fragmentos de código que permiten controlar cómo se mostrarán las URI o URL cuando se compartan en redes sociales.

Por su nombre, puede parecer que no es muy conocido o no está muy extendido, pero gracias a este estándar, cualquier documento o página web puede convertirse en un recurso rico en términos de social media. De hecho, es bastante utilizado en Facebook, LinkedIn o Twitter (cuando no se usan las Twitter

Cards) porque hace que cualquier documento o aplicación web pueda tener la misma funcionalidad que cualquier otro objeto o recurso de su misma índole.

## Description

El elemento DESCRIPTION especifica una breve descripción del documento u objeto.

```
<meta property="og:description" content="Aprende a crear buenas páginas web de  
forma rápida, clara y económica." />
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto que se encuentre en el rango de dos a cuatro líneas. La recomendación es, además, ser creativo y claro para conseguir clics.

## Image

El elemento IMAGE especifica la dirección de la imagen que está asociado al documento u objeto.

```
<meta property="og:image"  
content="https://islavvisual.com/img/urbanjobs.png" />
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto que se corresponda con el formato de URL, no obstante, la recomendación suele ser que, la imagen, esté en formato 1.91:1 y tenga un tamaño mínimo de 705x368 píxeles.

Cabe destacar que, aunque la recomendación es utilizar direcciones web que estén bajo el protocolo de transferencia de hipertexto seguro HTTPS, el protocolo de Open Graph permite la descripción de direcciones no seguras mediante HTTP. Si se desea declarar ambas direcciones se puede utilizar la propiedad SECURE\_URL.

```
<meta property="og:image" content="http://films.com/ET-cover.mp4" />  
<meta property="og:image:secure_url"  
content="https://films.com/ET-cover.mp4" />
```

Si, además, lo que se desea es especificar un valor de ancho, alto o tipo para la imagen es puede utilizar las propiedades WIDTH, HEIGHT y TYPE respectivamente.

```
<meta property="og:image:type" content="image/jpeg" />  
<meta property="og:image:width" content="705" />  
<meta property="og:image:height" content="368" />
```

Los valores permitidos para la propiedad TYPE están en la página de IANA Media Types en la dirección <https://www.iana.org/assignments/media-types/media-types.xhtml>.

## Locale

El elemento LOCALE especifica la ubicación del documento u objeto descrita a partir de la configuración regional.

```
<meta property="og:locale" content="es_ES" />
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor contemplado por las normas RFC 5646 o BCP 47, ambas, utilizando guion bajo en vez de medio. Es opcional puesto que, si no se especifica, su valor por defecto será “en\_US”.

Existe una variación de este elemento que es LOCALE:ALTERNATE y que describe la lista con las ubicaciones alternativas en otros idiomas dónde se encuentra disponible el documento u objeto.

```
<meta property="og:locale:alternate" content="en_US" />
<meta property="og:locale:alternate" content="en_GB" />
<meta property="og:locale:alternate" content="fr_FR" />
```

## Title

El elemento TITLE especifica el título del documento u objeto.

```
<meta property="og:title"
content="Creación de páginas web desde cero" />
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto que identifique con precisión el contenido al que hace referencia utilizando, para ello, entre 40 y 60 caracteres.

## Type

El elemento TYPE especifica el tipo o medio del documento u objeto que se va a compartir. Es decir, si es un sitio web, un artículo, una infografía, etcétera.

```
<meta property="og:type" content="article" />
```

En lo referente a los posibles valores admitidos, se debe especificar el valor adecuado según la lista oficial de tipos descrita en la url <https://ogp.me/#types>, pero los valores más frecuentes son ARTICLE para indicar que es un artículo y WEBSITE para indicar que es un sitio web.

## URL

El elemento URL especifica la dirección dónde se encuentra el contenido.

```
<meta property="og:url"
content="https://www.w3schools.com/tags/tag_meta.asp" />
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto que se rija por las normas de URL canónica, es decir, no deben contener propiedades extra ni parámetros de ninguna clase. Para que se entienda mejor este concepto, veamos el siguiente ejemplo:

URL 1: <https://www.dominio.com/XXX-0123-Maletín-universal?id=xj23145af&s=2>

URL 2: <https://www.dominio.com/XXX-0123-Maletín-universal>

Si nos fijamos en la ilustración anterior, la primera dirección utiliza dos parámetros que, seguramente, son para uso interno del sitio mientras que, la segunda, no lleva ningún parámetro. Pues bien, esta segunda dirección sin parámetros es la que los motores de búsqueda consideran la dirección canónica original y, la primera, la que consideran una copia.

## Video

El elemento VIDEO es una etiqueta especial que permite la declaración de todos los tipos de contenido multimedia y especifica la dirección del video que está asociado al documento u objeto.

```
<meta property="og:video"  
content="https://films.com/la-guerra-de-los-mundos.mp4" />
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto que se corresponda con el formato de URL, no obstante, aunque la recomendación es utilizar direcciones web que estén bajo el protocolo de transferencia de hipertexto seguro HTTPS, el protocolo de Open Graph permite la descripción de direcciones no seguras mediante HTTP. Si se desea declarar ambas direcciones se puede utilizar la propiedad SECURE\_URL.

```
<meta property="og:video" content="http://films.com/ET.mp4" />  
<meta property="og:video:secure_url" content="https://films.com/ET.mp4" />
```

El elemento VIDEO tiene como requerimiento la especificación de las propiedades de ancho, alto y tipo de contenido multimedia. Para ello se debe utilizar las propiedades WIDTH, HEIGHT y TYPE respectivamente.

```
<meta property="og:video:type" content="video/mp4" />  
<meta property="og:video:width" content="705" />  
<meta property="og:video:height" content="368" />
```

Los valores permitidos para la propiedad TYPE están en la página de IANA Media Types en la dirección <https://www.iana.org/assignments/media-types/media-types.xhtml>.

## TWITTER CARDS

El protocolo Twitter Cards o tarjetas de Twitter es un protocolo similar a Open Graphs, aunque sólo enfocado a Twitter. Es un lenguaje de marcado que permite controlar cómo se mostrará el contenido a los usuarios y provee la posibilidad de anexar información de Twitter en los sitios web y/o aplicaciones móviles.

Entre sus posibilidades tenemos:

### **twitter:card**

El elemento TWITTER:CARD especifica el tipo de tarjeta que se desea para el tweet. Es un metadato opcional, no obstante, si se definen los metadatos OG:TYPE, OG:TITLE y OG:DESCRIPTION, y TWITTER:CARD no está presente, se tomará como una tarjeta resumen.

```
<meta property="twitter:card" content="summary" />
```

Entre los posibles valores que admite este metadato tenemos:

Valor	Significado / Descripción
<b>summary</b>	Puede utilizarse en blogs, sitios de noticias, comercios electrónicos, etcétera. Su objetivo, es

	ofrecer una vista previa del contenido a anexar o enlazar en la cual existe, al menos, un título (TWITTER:TITLE), una descripción (TWITTER:DESCRIPTION) y una imagen (TWITTER:IMAGE) en miniatura.
<b>summary_large_image</b>	Es exactamente lo mismo que SUMMARY, si exceptuamos que la imagen no es una miniatura.
<b>app</b>	Suele utilizarse para promocionar aplicaciones móviles en Twitter, lo que suele conllevar un aumento de las instalaciones. En general, requiere de un nombre (TWITTER:APP:NAME), una descripción (TWITTER:APP:DESCRIPTION), una imagen o icono (TWITTER:APP:IMAGE), una calificación (TWITTER:APP:CALIFICATION) y un precio (TWITTER:APP:PRICE), no obstante, es una buena idea establecer el país de procedencia a través de la propiedad TWITTER:APP:COUNTRY.
<b>player</b>	Suele utilizarse para indicar que el contenido del tweet es un video o audio.
<b>lead_generation</b>	Suele usarse para recopilar información adicional sobre posibles clientes potenciales. Esto evita que se tengan que llenar grandes formularios ya que, con su nombre, nombre de usuario y correo electrónico es suficiente.

## twitter:site

El elemento TWITTER:SITE especifica el nombre del usuario del sitio web. Es aplicable cuando el tipo de tarjeta es SUMMARY, SUMMARY\_LARGE\_IMAGE o PLAYER y puede ser sustituido por TWITTER:SITE:ID. En general, se suelen establecer los dos.

```
<meta property="twitter:site" content="@UniversES" />
<meta property="twitter:site:id" content="12345678" />
```

## twitter:site:id

El elemento TWITTER:SITE:ID especifica el ID del usuario del sitio web. Es aplicable cuando el tipo de tarjeta es SUMMARY, SUMMARY\_LARGE\_IMAGE o PLAYER y puede ser sustituido por TWITTER:SITE. En general, se suelen establecer los dos.

```
<meta property="twitter:site" content="@UniversES" />
<meta property="twitter:site:id" content="12345678" />
```

## twitter:creator

El elemento TWITTER:CREATOR especifica el nombre del usuario que creó el contenido. Es aplicable cuando el tipo de tarjeta es SUMMARY\_LARGE\_IMAGE.

```
<meta property="twitter:creator" content="@universES" />
```

## twitter:creator:id

El elemento TWITTER:CREATOR especifica el ID del usuario que creó el contenido. Al igual que su antecesor, es opcional y es aplicable cuando el tipo de tarjeta es SUMMARY\_LARGE\_IMAGE.

```
<meta property="twitter:creator:id" content="12345678" />
```

## **twitter:description**

El elemento TWITTER:DESCRIPTION especifica una breve descripción del contenido, la cual, no puede exceder los 200 caracteres. Es aplicable cuando el tipo de tarjeta es SUMMARY, SUMMARY\_LARGE\_IMAGE o PLAYER.

```
<meta property="twitter:description" content="Noticias sobre la NAS..." />
```

## **twitter:title**

El elemento TWITTER:TITLE especifica el título del contenido, el cual, no puede exceder los 70 caracteres. Es aplicable cuando el tipo de tarjeta es SUMMARY, SUMMARY\_LARGE\_IMAGE o PLAYER.

```
<meta property="twitter:creator" content="UniversES – Últimas noticias" />
```

## **twitter:image**

El elemento TWITTER:IMAGE especifica la imagen que se utilizará en el contenido, la cual, no puede exceder los 5 MB. Es aplicable cuando el tipo de tarjeta es SUMMARY, SUMMARY\_LARGE\_IMAGE o PLAYER y, sus formatos admitidos son PNG, JPEG, WEBP y GIF. Sin embargo, si la imagen es de tipo GIF, sólo mostrará el primer frame del mismo.

```
<meta property="twitter:image"  
content="https://universes.es/images/logo.png" />
```

## **twitter:image:alt**

El elemento TWITTER:IMAGE:ALT especifica el texto alternativo para la imagen que se utilizará en el contenido, el cual, no puede exceder los 420 caracteres. Es aplicable cuando el tipo de tarjeta es SUMMARY, SUMMARY\_LARGE\_IMAGE o PLAYER.

```
<meta property="twitter:image_alt" content="Logotipo de universES" />
```

## **twitter:player**

El elemento TWITTER:PLAYER especifica la URL del marco (IFRAME) del reproductor. En general, suele llevar asociados los metadatos TWITTER:PLAYER:WIDTH, que especifica el ancho en píxeles, TWITTER:PLAYER:HEIGHT, que especifica el alto en píxeles y, TWITTER:PLAYER:STREAM, que indica que el medio está sin procesar.

Estos metadatos sólo son aplicables cuando el tipo de tarjeta es PLAYER.

```
<meta property="twitter:player"  
content="https://universes.es/videos/saturn-rings.avi" />
```

## **twitter:app**

El elemento TWITTER:APP especifica datos sobre el tipo de aplicación (si es para IPhone, Android o IPad), su nombre, dirección o URL y su ID, si procede. No obstante, en general, suele llevar asociado también el país de procedencia.

Estos metadatos sólo son aplicables cuando el tipo de tarjeta es APP.

```
<meta name="twitter:app:country" content="ES" />
<meta name="twitter:app:name:iphone" content="UniversES" />
<meta name="twitter:app:id:iphone" content="123456789" />
<meta name="twitter:app:url:iphone"
content="http://universes/action/4485e249560f9e600a1234cd" />
<meta name="twitter:app:name:ipad" content="UniversES" />
<meta name="twitter:app:id:ipad" content="123456789" />
<meta name="twitter:app:url:ipad"
content="http://universes/action/4485e249560f9e600a1234cd" />
<meta name="twitter:app:name:googleplay" content="UniversES" />
<meta name="twitter:app:id:googleplay" content="com.ejemplo.universes" />
<meta name="twitter:app:url:googleplay"
content="http://universes/action/4485e249560f9e600a1234cd" />
```

## **GOOGLE**

Google admite muchos más tipos de metadatos de lo que uno podría pensar. De hecho, existen algunos metadatos que son bastante desconocidos y pueden influenciar positivamente en el Posicionamiento SEO de la página consiguiendo un aumento del número de visitantes.

### **nositelinkssearchbox**

Permite especificar que no se muestre el cuadro de búsqueda de enlaces de sitio. Esto es, permite indicar si se permite que puedan buscar en tu sitio web directamente desde la página de resultados de Google por lo que, si este metadato NO está establecido, se tomará como acción permitida.

```
<meta name="google" content="nositelinkssearchbox" />
```

### **notranslate**

Cuando los usuarios buscan algo en Google, puede darse la posibilidad de que el idioma del usuario no coincida con el idioma de la página y, en ese caso, se ofrece un enlace que permite ver la página traducida de forma directa.

Esta directiva permite eso, indicar que NO se desea que se ofrezca la posibilidad de traducción desde la página de resultados de Google.

```
<meta name="google" content="notranslate" />
```

## **nopagereadaloud**

Hay ocasiones en que los usuarios indican al asistente de Google que lea en voz alta la página que está visitando o desean visitar.

Pues bien, esta directiva permite eso, indicar que NO se desea que se ofrezca la posibilidad de leer en voz alta la página cuando esta directriz está habilitada.

```
<meta name="google" content="nopagereadaloud" />
```

## **google-site-verification**

Permite especificar que la página que se está visitando está subida y verificada por el autor del sitio web. Esto es posible gracias a un identificador único que provee la Consola de Búsqueda de Google, accesible desde la cuenta de Google Webmasters Tools.

```
<meta name="google-site-verification"
content="+giHDUJoQpAZ123Bsjdi1o2tLZV21Alh5d1Nl45678vVuFHS9o=" />
```

## **PRÁCTICA 11: ASIGNACIÓN DE METADATOS**

### **Código del ejemplo**

<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-11>



### **Objetivo de la práctica**

Asignar los metadatos necesarios a nuestra página de inicio de UniversES.

### **Resolución**

Si recordamos, en la práctica 1, definimos la estructura de la página inicial de nuestro sitio UniversES. En ella, declaramos los siguientes metadatos:

```
<title>UniversES</title>
<meta charset="UTF-8">
<meta name="title" content="UniversES - Home">
<meta name="description" content="Todo sobre nuestro universo" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

Si nos fijamos en la etiqueta TITLE, únicamente se establece el nombre del sitio web, lo cual no aporta ninguna información ni resulta de gran ayuda. Por tanto, si queremos que la etiqueta se vuelva más

usable, legible y accesible, lo que debemos hacer es establecer un valor descriptivo y claro, como, por ejemplo:

```
<title>UniversES – Página de Inicio / Últimas noticias</title>
```

La segunda cosa que se puede observar en la definición de metadatos anterior es que, la etiqueta TITLE y el metadato TITLE no tienen el mismo contenido. Esto no tiene por qué ser un error, o un problema, puesto que la etiqueta TITLE está actualmente dedicada a mostrar el título de la página en los agentes de usuario y, el metadato TTILE está más orientado a los crawlers. Sin embargo, si se establecen ambas, lo suyo es que tengan el mismo contenido.

Otra cosa que se puede observar si se ha prestado atención a este capítulo es que falta el metadato KEYWORDS y que, el metadato DESCRIPTION, resulta poco descriptivo. Para solucionar esto, lo que se puede hacer es lo siguiente:

```
<meta name="description" content="UniversES te ofrece las últimas noticias, fotos y videos sobre el universo recopiladas de fuentes como la NASA, HubbleSite, N2YO, Wikipedia o Sky Map" />
<meta name="keywords" content="Universo, planetas, satélites, blog noticias universo, blog noticias espacio, lunas, iis, sistema solar" />
```

Y, por último, pero no menos importante, puede que caigamos en la cuenta de que faltan metadatos importantes como los de Dublin Core, Twitter Cards, Open Graph y Google:

```
<!-- GENERALES -->
<meta name="generator" content="Drupal 8 (http://drupal.org)" />
<!-- GOOGLE -->
<meta name="google-site-verification"
content=" +giHDUJoQpAZ123Bsjdi1o2tLZV21AIh5d1Nl45678vVuFHs9o=" />
<!-- DUBLIN CORE -->
<meta name="dcterms.identifier" content="/" />
<meta name="dcterms.title" content="UniversES" />
<meta name="dcterms.format" content="text/html" />
<!-- TWITTER CARDS -->
<meta name="twitter:app:country" content="ES" />
<meta name="twitter:app:name:iphone" content="UniversES" />
<meta name="twitter:app:id:iphone" content="123456789" />
<meta name="twitter:app:url:iphone"
content="http://universes/action/4485e249560f9e600a1234cd" />
<meta name="twitter:app:name:ipad" content="UniversES" />
<meta name="twitter:app:id:ipad" content="123456789" />
<meta name="twitter:app:url:ipad"
content="http://universes/action/4485e249560f9e600a1234cd" />
<meta name="twitter:app:name:googleplay" content="UniversES" />
<meta name="twitter:app:id:googleplay" content="com.ejemplo.universes" />
<meta name="twitter:app:url:googleplay"
content="http://universes/action/4485e249560f9e600a1234cd" />
<!-- OPEN GRAPH -->
<meta property="og:url" content="http://www.universes.es/" />
<meta property="og:image" content="http://www.universes/images/logo.png" />
<meta property="og:type" content="website" />
<meta property="og:description"
content="UniversES te ofrece las últimas noticias, fotos y videos
sobre el universo recopiladas de fuentes como la NASA,
```

# 10

## SVG

### INTRODUCCIÓN

Los Gráficos Vectoriales Redimensionables o Escalables (Scalable Vector Graphics) o SVG es una especificación que describe cómo diseñar gráficos vectoriales bidimensionales, tanto estáticos como animados a partir de en lenguaje XML. Dicho con otras palabras, SVG es un formato que permite diseñar gráficos definidos como texto basados en la especificación del lenguaje XML. Su desarrollo y evolución están a cargo de la W3C (World Wide Web Consortium).

Al margen de su naturaleza tecnológica, el formato SVG cuenta con numerosas ventajas que deberíamos tener en cuenta. En primer lugar, es un formato estándar Open Source, es decir, que se puede utilizar sin licencias ni permisos especiales.

En segundo lugar, SVG es el medio ideal para diseñar gráficos con un peso muy aceptable. No es un formato fácil de manejar, pero sus posibilidades son mucho más espectaculares de lo que uno, a simple vista, puede llegar a imaginar.

Entre las principales ventajas que posee el lenguaje SVG podemos destacar que permiten un renderizado de gráficos sin solapamientos ni pérdidas de información, el diseño de patrones de relleno, gradientes, filtros, animaciones y efectos avanzados sin invertir mucho tiempo en los desarrollos, la compresión a través de GZIP, la fácil indexación gracias a contenido XML textual y una total integración con lenguajes como CSS, XML, HTML o XHTML.

En lo referente a su soporte nativo, hoy en día lo soportan todos los navegadores, tanto en su forma interna, es decir, la incrustación en otro tipo de documentos, como en su forma externa, es decir, a través de llamadas externas a través de elementos como PICTURE o IMG.

De hecho, la especificación SVG permite su uso a modo de documento único, como un objeto embebido por referencia, como un objeto embebido en línea, a través de links externos, de CSS o

de XSL.

## PRIMITIVAS BÁSICAS DE SVG

### Primitiva svg

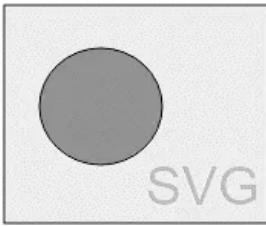
Para empezar a definir o crear un gráfico en Formato SVG, lo primero, es poner la cabecera indicador de que se va a interpretar como un gráfico. Esto es:

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
</svg>
```

Dentro de la estructura podremos ir insertando gráficos individuales, como círculos, rectángulos, rutas, degradados, o texto, entre otras posibilidades. A continuación, se muestra un ejemplo.

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
      xmlns:xlink="http://www.w3.org/1999/xlink"
      width="200"
      height="200"
      version="1.1">
  <rect x="13"
        y="42"
        width="172"
        height="142"
        style="fill:rgb(255,244,193);
               stroke:rgb(0,0,0);
               stroke-width:1" />
  <text x="105"
        y="172"
        fill="rgba(0,0,250,0.5)"
        font-size="35"
        font-family="Arial">SVG</text>
  <ellipse cx="76"
           cy="108"
           rx="40"
           ry="38"
           fill="red"
           stroke="black"
           stroke-width="1" />
</svg>
```

Su resultado será algo similar a:



Como uno se podrá imaginar, cuando se utilizan primitivas lineales como RECT, las coordenadas (x, y) dónde empezar a dibujar se definen a través de los atributos X e Y y, su ancho y alto.

Sin embargo, cuando se utilizan primitivas radiales ELLIPSE, las coordenadas (x, y) dónde empezar a dibujar se expresan a través de los atributos CX y CY.

Los atributos de FILL y STROKE son para definir el relleno y bordes respectivamente. Las propiedades de FONT-SIZE y FONT-FAMILY establecen el tamaño y tipo de letra en primitivas de gestión de textos.

## ATRIBUTO VIEWBOX

El atributo VIEWBOX de SVG resulta ser algo muy similar al metadato VIEWPORT y está igualmente asociado al concepto de tamaño de ventana gráfica. Si se piensa en un documento como un lienzo, el VIEWBOX (o área de visión) es la parte del lienzo que deseamos que vea el espectador. Es decir, aunque la página puede cubrir la pantalla completa, la figura o diseño que se va a realizar sólo puede existir dentro de un área de visión concreta.

Cuando el atributo VIEWBOX se utiliza correctamente, indica al analizador en qué parte del documento se debe hacer enfocar, eliminando el espacio en blanco extra de alrededor. Dicho de otro modo, actúa como una lente que disminuye o aumenta el área o zona enfocada.

La mejor forma de usar este atributo es pensar en él como si fuese el recorte de una imagen, es decir, enfocándose en los puntos dónde se harán los cortes. Para ello, se tendrán que definir un valor de X inicial, el cual marcará el comienzo en el eje de las abscisas (eje X) y un valor inicial de Y, que marcará el principio dentro del eje de las ordenadas (eje Y). A partir de este punto en el plano, marcaremos un ancho y alto para visualizar el contenido.

Cabe destacar que, la anchura y la altura del área de visión no tiene por qué ser la anchura y la altura definida para el documento SVG puesto que, esta área puede cubrir todo el lienzo o sólo una parte.

Supongamos el siguiente código:

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="100"  
height="100"  
viewBox="0 0 100 100">  
<rect x="1"  
y="1"
```

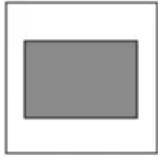
```

width="98"
height="98"
fill="none"
stroke="black"
stroke-width="1" />
<rect x="13"
y="26"
width="73"
height="50"
fill="gray"
stroke="black"
stroke-width="1" />
</svg>

```

Si quisiéramos que se viese todo el lienzo, lo que se debería establecer es el siguiente VIEWBOX:

```
<svg width="100" height="100" viewBox="0 0 100 100">
```



Ahora, si quisiéramos que se viese al doble de su tamaño, es decir, como si tuviese aplicado un efecto de ampliación o aumento de un 200%, lo que se debería establecer es:

```
<svg width="800" height="400" viewBox="0 0 50 50">
```



Sin embargo, si quisiéramos que se viese a la mitad de su tamaño, es decir, como si tuviese aplicado un efecto de reducción a un 50%, lo que se debería establecer es:

```
<svg width="800" height="400" viewBox="0 0 200 200">
```



## Primitiva **defs**

La primitiva DEFS especifica una serie de “directrices” que sirven como definiciones identificables en el documento. Estas definiciones son reutilizables a través de identificadores únicos, pero no se aplicarán hasta que, al menos, se realice una llamada al recurso. Según como se mire, podría considerarse como un modelo que establece un único esquema para todo el documento y que puede utilizarse para construir otras estructuras múltiples.

Para que la primitiva DEFS pueda utilizarse, debe declararse como primer descendiente directo de la primitiva raíz SVG.

```
<svg xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  width="100%" height="100%">
<defs> ... </defs>
</svg>
```

Esto es así porque, cuando el analizador de código ve la primitiva DEFS sabe que no debe mostrar su contenido y que se deben almacenar los nombres de identificación para un uso futuro.

El uso de esta primitiva, no sólo es beneficioso a nivel de tamaño de archivos o compresión, sino que también resulta francamente útil en lo que se refiere a Accesibilidad Web. En él, normalmente, se pueden definir desde gradientes, hasta patrones o filtros.

```
<svg xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  width="100%" height="100%">
<defs>
<circle id="circuloRadio10" cx="0" cy="0" r="10" />
</defs>
</svg>
```

## Primitiva rect

La primitiva RECT especifica que la forma que se desea realizar es una forma rectangular con o sin bordes redondeados. El que una forma rectangular tenga un borde redondeado significará que, en ocasiones, se confunda con un círculo o elipse.

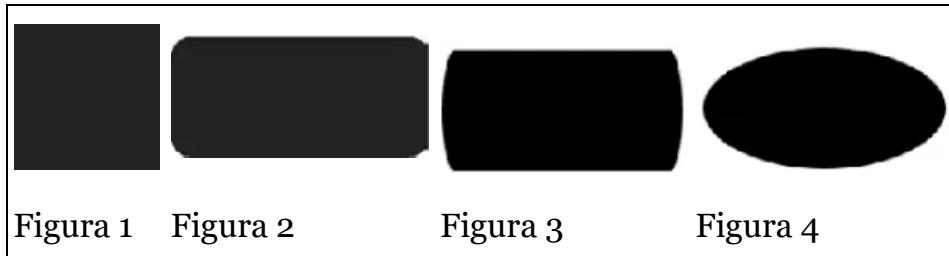
Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>x</b>	Indica el comienzo en el eje de las abscisas o eje de las X.
<b>y</b>	Indica el comienzo en el eje de las ordenadas o eje de las Y.
<b>width</b>	Indica el ancho de la forma.
<b>height</b>	Indica el alto de la forma.
<b>rx</b>	Indica el radio que se debe definir al eje X. Si se define un valor de RX, pero no de RY, se considera que el valor omitido es el mismo. Si el valor de radio es superior a la mitad de ancho de la forma, el valor de radio que se aplicará será el correspondiente a la mitad del ancho.
<b>ry</b>	Indica el radio que se debe definir al eje Y. Si se define un valor de RY, pero no de RX, se considera que el valor omitido es el mismo. Si el valor de radio es superior a la mitad de ancho de la forma, el valor de radio que se aplicará será el correspondiente a la mitad del ancho.

## Ejemplo:

```
<svg viewBox="0 0 500 200" xmlns="http://www.w3.org/2000/svg">
<!-- Figura 1: cuadrado -->
<rect x="1" y="1" width="50" height="50"></rect>
<!-- Figura 2: Rectángulo con bordes redondeados -->
<rect x="101" y="1" width="100" height="50" rx="10"></rect>
<!-- Figura 3: Rectángulo con bordes redondeados -->
<rect x="251" y="1" width="100" height="50" rx="10" ry="100"></rect>
<!-- Figura 4: Rectángulo con bordes redondeados -->
<rect x="400" y="1" width="100" height="50" rx="200" ry="100"></rect>
</svg>
```

El resultado debería ser algo similar a:



## Primitiva circle

La primitiva CIRCLE especifica que la forma a representar es un círculo.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>cx</b>	Indica el punto central en el eje de las abscisas (eje X) donde se definirá la forma con el radio especificado por R.
<b>cy</b>	Indica el punto central en el eje de las ordenadas (eje Y) donde se definirá la forma con el radio especificado por R.
<b>r</b>	Indica el radio que se debe definir a la forma circular.

## Ejemplo:

```
<svg viewBox="0 0 200 100" xmlns="http://www.w3.org/2000/svg">
<circle cx="50" cy="50" r="50"></circle>
</svg>
```

## Primitiva ellipse

La primitiva ELLIPSE especifica que la forma a representar es una elipse.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
----------	-------------

<b>cx</b>	Indica el punto central en el eje de las abscisas (eje X) donde se definirá la forma con el radio especificado por R.
<b>cy</b>	Indica el punto central en el eje de las ordenadas (eje Y) donde se definirá la forma con el radio especificado por R.
<b>rx</b>	Indica el radio en el eje X que se debe definir a la forma circular.
<b>ry</b>	Indica el radio en el eje Y que se debe definir a la forma circular.

### Ejemplo:

```
<svg viewBox="0 0 200 100" xmlns="http://www.w3.org/2000/svg">
<ellipse cx="50" cy="50" rx="50" ry="20"></ellipse>
</svg>
```

## Primitiva line

La primitiva LINE especifica que la forma a representar es una línea.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>x1</b>	Es el valor inicial en el eje de las abscisas (eje X). Junto con el valor de Y1, define la coordenada superior izquierda.
<b>y1</b>	Es el valor inicial en el eje de las ordenadas (eje Y). Junto con el valor de X1, define la coordenada superior izquierda.
<b>x2</b>	Es el valor final en el eje de las abscisas (eje X). Junto con el valor de Y1, define la coordenada inferior derecha.
<b>y2</b>	Es el valor final en el eje de las ordenadas (eje Y). Junto con el valor de X1, define la coordenada inferior derecha.
<b>stroke</b>	Indica el color de la línea. Admite todos los formatos compatibles con CSS.

### Ejemplo:

```
<svg viewBox="0 0 200 100" xmlns="http://www.w3.org/2000/svg">
<line x1="10" y1="10" x2="90" y2="90" stroke="black"></line>
</svg>
```

## Primitiva polyline

La primitiva POLYLINE especifica que la forma a representar es un conjunto de segmentos o líneas que definen una forma abierta, aunque también puede ser cerrada.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
----------	-------------

<b>points</b>	Es el conjunto de coordenadas. El conjunto se define a través de pares de valores separados por comas y diferenciadas a través de espacios. Si bien, el primer punto establece el punto inicial, el segundo dibuja una línea entre el primero y el segundo. El tercer punto o coordenada, dibuja una línea con respecto a su predecesor, y así sucesivamente.
<b>stroke</b>	Indica el color de la línea. Admite todos los formatos compatibles con CSS.
<b>fill</b>	Indica el color de relleno. Admite todos los formatos compatibles con CSS.

### Ejemplo:

```
<svg viewBox="0 0 500 100" xmlns="http://www.w3.org/2000/svg">
<polyline points="10,10 50,10 50,90 190,90 190,10 450,10 450,90 500,90"
fill="none" stroke="black"></polyline>
</svg>
```

El resultado debería ser algo similar a:



### Primitiva polygon

La primitiva POLYGON especifica que la forma que se desea realizar es un conjunto de segmentos o líneas que definen una forma cerrada, es decir, especifica que la forma que se desea realizar es un polígono.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>points</b>	Es el conjunto de coordenadas. El conjunto se define a través de pares de valores separados por comas y diferenciadas a través de espacios. Si bien, el primer punto establece el punto inicial, el segundo dibuja una línea entre el primero y el segundo. El tercer punto o coordenada, dibuja una línea con respecto a su predecesor, y así sucesivamente.
<b>stroke</b>	Indica el color de la línea. Admite todos los formatos compatibles con CSS.
<b>fill</b>	Indica el color de relleno. Admite todos los formatos compatibles con CSS.

### Ejemplo:

```
<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg">
<polygon points="10,10 50,50 90,50 90,10"></polygon>
</svg>
```

## Primitiva a

La primitiva A especifica que el contenido asociado representa un vínculo o enlace, idéntico en comportamiento al de HTML.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>download</b>	Especifica que el contenido al que apunta el enlace debe ser descargado. Aunque casi todos los navegadores lo soportan, no es funcional con ningún navegador de Microsoft hasta la versión 18 de Microsoft Edge.
<b>href</b>	Especifica el destino hacia dónde se irá cuando se pulse en el enlace. Si este valor empieza por el símbolo almohadilla, indicará que se desea ir a otra sección del documento actual. De no ser así, indicará la dirección hacia otro documento diferente.
<b>hreflang</b>	Especifica el idioma del documento vinculado.
<b>media</b>	Especifica para qué medios y/o dispositivos está optimizado el documento vinculado.
<b>referrerpolicy</b>	Especifica la cabecera de HTTP que debe incluirse en la petición de envío cuando se haga clic. No es frecuente verlo en los enlaces de HTML, aunque sí en peticiones Ajax.
<b>rel</b>	Especifica la relación existente entre el documento actual y el vinculado. Entre los posibles valores que puede tomar, los más frecuentes son NOREFERRER, para indicar que no se envíe ningún encabezado, NOFOLLOW, para indicar que el enlace no sea rastreado por los crawlers y SEARCH, para indicar que el documento es una página de búsqueda.
<b>target</b>	Especifica dónde se abrirá el vínculo. Entre los posibles valores que puede tomar, los más frecuentes son _BLANK, para indicar que se abra en una nueva pestaña, _SELF, para indicar que se abra en la misma pestaña y _TOP, para que se abra en el primer elemento BODY de la ventana.
<b>type</b>	Especifica el tipo de medio, antes conocido como MIME type, del documento vinculado. Los posibles valores que puede tomar se encuentran en la URL <a href="http://www.iana.org/assignments/media-types/">http://www.iana.org/assignments/media-types/</a> .

### Ejemplo:

```
<svg viewBox="0 0 200 100" xmlns="http://www.w3.org/2000/svg">
<text text-anchor="middle" x="100" y="50">
Pelusa es un <a href="https://dle.rae.es/animal">animal</a>
</text>
</svg>
```

## Primitiva path

La primitiva PATH especifica que la forma que se desea realizar es un conjunto de líneas o curvas. En general, cualquier forma básica puede realizarse a través de la primitiva PATH, sin embargo, se suele utilizar para crear formas no lineales o complejas.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>d</b>	Especifica la trayectoria indicando, mediante los comandos introducidos, si se dibuja o no y el qué.
<b>pathLength</b>	Especifica una longitud total para la ruta en unidades de usuario. En general, se utiliza para escalar las distancias de las rutas definidas por el atributo D usando como valor de relación el dispuesto por este atributo.

Cuando se habla de comandos, se hace referencia a una anotación que se expresa a través de una letra (mayúscula o minúscula) seguido de unos parámetros.

Si la letra es mayúscula, lo que se está especificando son coordenadas absolutas en la página, mientras que, si la letra es minúscula, lo que se está especificando son coordenadas relativas al último comando.

Cabe destacar que, los valores del atributo D siempre van sin unidades asociadas y, por lo tanto, están basadas en el sistema de coordenadas del usuario.

### **COMANDO MOVER (M)**

Mueve el cursor hasta el punto indicado (x,y). Si especificamos el parámetro en minúsculas el cursor se desplazará (x,y) puntos con respecto a la ubicación actual.

#### **Ejemplo:**

M10 10

### **COMANDO LÍNEA (L)**

Dibuja una línea desde el punto actual hasta (x,y). Si especificamos el parámetro en minúsculas la línea será dibujada tomando como referencia el punto anterior o ubicación actual.

#### **Ejemplo:**

L10 10

### **COMANDO HORIZONTAL (H)**

Traza una línea horizontal hasta el punto actual hasta (x,y). Si especificamos el parámetro en minúsculas la línea será dibujada tomando como referencia el punto anterior o ubicación actual.

#### **Ejemplo:**

H100

### **COMANDO VERTICAL (V)**

Traza una línea vertical hasta el punto actual hasta (x,y). Si especificamos el parámetro en minúsculas la línea será dibujada tomando como referencia el punto anterior o ubicación actual.

## **Ejemplo:**

V100

## **COMANDO CERRAR (Z)**

Cierra el camino, trazando una línea desde el punto actual hasta el primer punto que se estableció en PATH actual.

## **Ejemplo:**

Mo o H50 V100 Z

Si especificamos el parámetro en minúsculas el cursor se desplazará (x,y) puntos con respecto a la ubicación actual.

## **COMANDO DE CURVA CÚBICA (C)**

Dibuja una curva cúbica de Bézier desde el punto actual (x,y). Los valores de (x<sub>1</sub>, y<sub>1</sub>) y (x<sub>2</sub>, y<sub>2</sub>) son los puntos de inicio y final de la curva de control, que controlan cómo se dobla. Si especificamos el parámetro en minúsculas las coordenadas se interpretan de forma relativa al punto actual.

## **Ejemplo:**

C20 00, 50 20, 50 10

## **COMANDO DE CURVA CÚBICA SUAVE (S)**

Dibuja una curva cúbica suave de Bézier desde el punto actual (x,y). x<sub>2</sub>, y<sub>2</sub> es el punto de control final. El punto de control de inicio se supone que es el mismo que el punto de control final de la curva anterior. Si especificamos el parámetro en minúsculas las coordenadas se interpretan de forma relativa al punto actual.

## **Ejemplo:**

S20 0, 50 20

## **COMANDO DE CURVA CUADRÁTICA (Q)**

Dibuja una curva cuadrática de Bézier desde el punto actual (x,y). x<sub>1</sub>, y<sub>1</sub> es el punto de control que controla cómo se dobla la curva. Si especificamos el parámetro en minúsculas las coordenadas se interpretan de forma relativa al punto actual.

## **Ejemplo:**

Q20 0, 50 20

## **COMANDO DE CURVA CUADRÁTICA SUAVE (T)**

Dibuja una curva cuadrática suave de Bézier desde el punto actual (x, y). El punto de control se supone que es el mismo que el último punto de control utilizado. Si especificamos el parámetro en

minúsculas las coordenadas se interpretan de forma relativa al punto actual.

### Ejemplo:

Q20 00, 50 20 T100 10

## COMANDO ARCO

Dibuja un arco elíptico desde el punto actual hasta el punto (x, y). Este es uno de los pocos comandos que utiliza varios parámetros, algunos bastante complejos de entender.

Su sintaxis es:

Arx,ry x-axis-rotation large-arch-flag sweepflag x,y

Los parámetros RX y RY son el radio de la elipse en la dirección X e Y. El parámetro X-AXIS-ROTATION determina la cantidad de arco se va a girar alrededor del eje x. Sólo parece tener un efecto cuando rx y ry tienen valores diferentes. El parámetro LARGE-ARCH-FLAG puede tomar los valores 0 o 1 aunque no siempre hace efecto. El parámetro SWEEPFLAG determina la dirección para dibujar el arco.

Si especificamos el parámetro en minúsculas las coordenadas se interpretan de forma relativa al punto actual.

### Ejemplo:

A 30 20 50 0 1 180 225

## COMENTARIOS Y EJEMPLOS

Con respecto a escribir los comandos en mayúsculas o minúsculas, como se ha indicado antes, básicamente, la diferencia estriba en el posicionamiento. Si escribimos con letras mayúsculas significa una posición absoluta, mientras que en minúsculas significa una posición relativa.

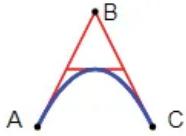


El ejemplo del triángulo que acabamos de mostrar, define un PATH que comienza en la posición 75,0 con una línea en la posición 25,100. A continuación, a partir de ahí, una línea a 125,100 y, que finalmente cierra el camino de regreso hasta el punto 75,0.

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
<path d="M75,0 L25,100 L125,100 Z" fill="black" />
</svg>
```

Las curvas de Bézier se utilizan para modelar curvas suaves que se pueden ampliar de forma indefinida. Generalmente, el usuario selecciona dos puntos extremos y uno o dos puntos de control. Una curva de Bézier con un punto de control se denomina curva Bézier cuadrática y al tipo con los dos puntos de control se denomina curva Bézier cúbica.

A continuación, se muestra una curva Bézier cuadrática, donde A y C son los puntos inicial y final, y B es el punto de control.



```
<svg viewBox="0 0 400 400" xmlns="http://www.w3.org/2000/svg">
<path id="lineAB"
d="M 50 175 l 75 -150"
stroke="red"
stroke-width="3"
fill="none" />
<path id="lineBC"
d="M 125 25 l 75 150"
stroke="red"
stroke-width="3"
fill="none" />
<path id="lineBC"
d="M 87.5 100 l 75 0"
stroke="red"
stroke-width="3"
fill="none" />
<path id="quadcurveABC"
d="M 50 175 q 75 -150 150 0"
stroke="blue"
stroke-width="5"
fill="none" />
<!-- Mark relevant points --&gt;
&lt;g stroke="black" stroke-width="3" fill="black"&gt;
&lt;circle id="pointA" cx="50" cy="175" r="3"&gt;&lt;/circle&gt;
&lt;circle id="pointB" cx="125" cy="25" r="3"&gt;&lt;/circle&gt;
&lt;circle id="pointC" cx="200" cy="175" r="3"&gt;&lt;/circle&gt;
&lt;/g&gt;
<!-- Label the points --&gt;
&lt;g font-size="30" fill="black" stroke="none" text-anchor="middle"&gt;
&lt;text x="50" y="175" dx="-30"&gt;A&lt;/text&gt;
&lt;text x="145" y="45" dy="-10"&gt;B&lt;/text&gt;
&lt;text x="200" y="175" dx="30"&gt;C&lt;/text&gt;
&lt;/g&gt;
&lt;/svg&gt;</pre>
```

Debido a que la creación de trayectorias y formas a través de la primitiva PATH resulta bastante compleja, suele ser recomendable utilizar un editor de SVG para crear gráficos complejos como inkscape, Corel Draw o Adobe Illustrator.

## Primitiva image

La primitiva IMAGE permite renderizar imágenes rasterizadas dentro de la declaración de un objeto SVG.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>x</b>	Indica el comienzo en el eje de las abscisas o eje de las X. Su valor por defecto es 0.
<b>y</b>	Indica el comienzo en el eje de las ordenadas o eje de las Y. Su valor por defecto es 0.
<b>width</b>	Indica el ancho de la forma. Su valor por defecto es 0.
<b>height</b>	Indica el alto de la forma. Su valor por defecto es 0.
<b>href</b>	Especifica la URL dónde está localizada o ubicada la imagen.
<b>preserveAspectRatio</b>	Especifica que se debe forzar una escalabilidad uniforme y, en cuyo caso, el método de alineación que se debe usar para en la ventana gráfica o VIEWBOX. Entre los posibles valores podemos encontrar todas las variaciones en MAX, MIN y MED para cada valor de X o Y. Por ejemplo, la especificación XMINYMED establece que se alinee con el valor más pequeño del eje X y con el valor o punto medio del eje Y. Admite un parámetro opcional que indica si se debe permitir que la imagen sea cortada (SLICE) o debe ser mostrada de forma íntegra (MEET).

## Ejemplo:

```
<svg viewBox="0 0 200 100" xmlns="http://www.w3.org/2000/svg">
<image preserveAspectRatio="xMinYMed meet"
href="./logo.png"
width="128"
height="64" />
</svg>
```

## TEXTOS

## Primitiva text

Una de las necesidades que a veces se da a la hora de diseñar es la creación de textos con perspectiva. En realidad, no vamos a descubrir nada, puesto que todo lo que se puede realizar en SVG se puede hacer en CSS, sin embargo, es una manera más para conseguir nuestro objetivo.

La primitiva TEXT especifica que la forma que se desea realizar o crear es una caja de texto. Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>x</b>	Es el valor inicial en el eje de las abscisas (eje X). Junto con el valor de Y, define la coordenada superior izquierda.
<b>y</b>	Es el valor inicial en el eje de las ordenadas (eje Y). Junto con el valor de X, define la coordenada superior izquierda.
<b>dx</b>	Indica el desplazamiento a realizar en el eje de las abscisas antes de empezar a pintar.
<b>dy</b>	Indica el desplazamiento a realizar en el eje de las ordenadas antes de empezar a pintar.
<b>lengthAdjust</b>	Especifica como debe ser el ajuste del texto, si sólo mediante el espacio entre caracteres, o también mediante el tamaño del glifo. Sus posibles valores son SPACING y SPACINGANDGLYPHS. Por defecto está asignado a SPACING.
<b>text-anchor</b>	Indica cómo se debe realizar la alineación del texto, si por el principio, centrada o por el final.
<b>textLength</b>	Especifica el ancho disponible para pintar o dibujar. Con esta opción, el navegador se asegurará de que el texto no se desborde fuera de los límites marcados por esta propiedad.

### Ejemplo:

```
<svg viewBox="0 0 200 100" xmlns="http://www.w3.org/2000/svg">
<text text-anchor="middle" x="100" y="50">Pelusa es un animal</text>
</svg>
```

## Primitiva tspan

La primitiva TSPAN especifica una zona interna al texto en la que se puede asignar un estilo diferente al que se estableció en la etiqueta padre. Además, puede establecer más primitivas de texto, como si de líneas de un párrafo se tratase. Con esta primitiva, es posible agregar negrita dentro del texto o, incluso, agregar texto con otras coordenadas dentro de la primitiva TEXT.

Entre los atributos que admite en su configuración, no hay ninguno que se deba destacar, si exceptuamos los ya descritos por la primitiva TEXT.

### Ejemplo:

```
<svg viewBox="0 0 200 100" xmlns="http://www.w3.org/2000/svg">
<style>
text { font-size: 14px; }
tspan { font-weight: bold; }
</style>
<text text-anchor="middle" x="100" y="50">
<tspan>Pelusa</tspan> es un animal
</text>
</svg>
```

## Primitiva textPath

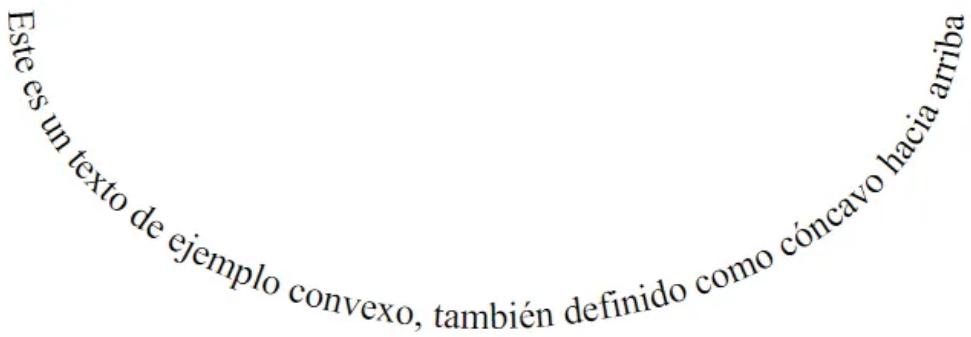
La primitiva TEXTPATH especifica una trayectoria para una primitiva de texto, es decir, especifica el camino por el que se distribuirá el texto. Dicha trayectoria se establece a través de la primitiva PATH.

Entre los atributos que admite en su configuración, sólo cabe destacar HREF, que indica la referencia a la primitiva PATH que define la trayectoria.

### Ejemplo:

```
<svg viewBox="0 0 500 100" xmlns="http://www.w3.org/2000/svg">
<defs>
<path id="path1" d="M0,20 a3,2 0 0,0 200,0" />
</defs>
<text x="0" y="100">
<textPath href="#path1">
Este es un texto de ejemplo convexo, también definido como cóncavo hacia arriba
</textPath>
</text>
</svg>
```

El resultado debería ser algo similar a:



## GRADIENTES

### Primitiva lineargradient

En los gráficos vectoriales escalables (SVG), vamos escribiendo las instrucciones para que, el analizador de SVG, cree una imagen interpretando dichas instrucciones. Los gradientes lineales, también conocidos como degradados lineales, lo que hacen es, fundir un color con otro de forma secuencial proporcionando una cierta profundidad a los diseños.

Aunque no es la única posibilidad, cuando se trabaja con degradados, lo adecuado es definirlo dentro de una primitiva DEFS. Como ya se explicó anteriormente, la etiqueta DEFS funciona como un disparador que declara que se va a definir algo para un posterior uso. En este caso, un degradado.

La primitiva LINEARGRADIENT especifica un fundido de color en una dirección o sentido cualquiera. Eso sí, para conseguir esta característica se debe recurrir al atributo GRADIENTTRANSFORM.

Para poder utilizar el degradado definido, dentro de cada primitiva se debe proporcionar un atributo FILL con el ID del gradiente para identificar la estructura o esquema y poder usarlo mediante una dirección de referencia.

Cabe destacar que, la primitiva LINEARGRADIENT, utiliza las coordenadas (x, y) para definir el punto de cambio del color. Por ejemplo, al cambiar la dirección con respecto al eje X se puede observar que el color se fusiona de izquierda a derecha mientras que, si se cambia con respecto al eje Y, se observa que el color se fusiona de arriba hacia abajo.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

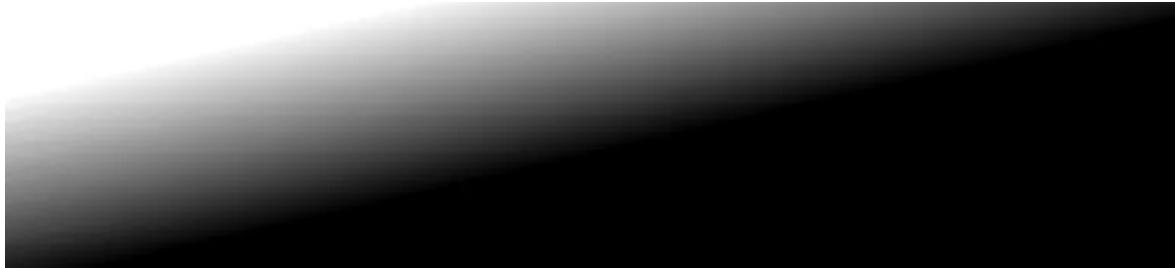
Atributo	Descripción
<b>x1</b>	Es el valor inicial en el eje de las abscisas (eje X). Junto con el valor de Y1, define la coordenada superior izquierda.
<b>y1</b>	Es el valor inicial en el eje de las ordenadas (eje Y). Junto con el valor de X1, define la coordenada superior izquierda.
<b>x2</b>	Es el valor final en el eje de las abscisas (eje X). Junto con el valor de Y1, define la coordenada inferior derecha.
<b>y2</b>	Es el valor final en el eje de las ordenadas (eje Y). Junto con el valor de X1, define la coordenada inferior derecha.
<b>href</b>	Especifica una referencia a otro degradado lineal que será utilizado como plantilla.
<b>spreadMethod</b>	Especifica si el degradado debe empezar o terminar en los límites de la forma que utiliza o aplica el degradado.
<b>gradientTransform</b>	Especifica la transformación a realizar sobre el degradado. Los funciones y valores válidos son los mismos que los aplicables en CSS.
<b>gradientUnits</b>	Especifica las unidades de medida en las que se deben de basar los ejes de coordenadas donde se aplica el degradado.

### Ejemplo:

```
<svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%">
<defs>
<linearGradient id="grad45"
gradientUnits="userSpaceOnUse"
gradientTransform="rotate(45)">
<stop offset="25%" stop-color="white" />
<stop offset="75%" stop-color="black" />
</linearGradient>
</defs>
```

```
<!-- Para utilizar el degradado -->
<rect x="1" y="1" width="100%" height="100%" fill="url(#grad45)" />
</svg>
```

El resultado debería ser algo similar a:



## Primitiva stop

Como se acaba de ver en el ejemplo anterior LINEARGRADIENT gestiona el flujo de color y STOP define el punto de inicio. Por este motivo, si se desea realizar un degradado, al menos, se necesitan dos primitivas STOP.

Al igual que sucede con todos las primitivas, propiedades y atributos de comportamiento similar, la primitiva STOP permite la definición de colores por nombre, RGB, HSL y hexadecimal.

Para entender mejor cómo funcionan los gradientes y la primitiva STOP, analicemos el ejemplo anterior de LINEARGRADIENT. El primer conjunto de coordenadas (primera primitiva STOP) especifica el punto de inicio del degradado y, el segundo conjunto, identifica la dirección del gradiente.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>offset</b>	Especifica dónde se iniciará la fusión al color indicado por el atributo STOP-COLOR.
<b>stop-color</b>	Especifica el color de parada en el degradado. También es posible utilizarlo como una propiedad de CSS.
<b>stop-opacity</b>	Especifica el grado de transparencia que se desea aplicar al color definido por el atributo STOP-COLOR. Sus posibles valores van de 0 a 1. Su valor por defecto es 1.

## Primitiva radialgradient

La primitiva RADIALGRADIENT especifica un fundido de color que comienza en un punto central y se va aplicando de manera radial.

No me cansaré de repetirlo. Cuando se trabaja con degradados, sean lineales o radiales, lo adecuado es definirlo dentro de una primitiva DEFS porque, como ya se explicó anteriormente, la etiqueta DEFS funciona como un disparador que declara que se va a definir algo para un posterior uso.

También, al igual que sucede con los degradados lineales, para poder definirlos a través de la primitiva DEFS, dentro de cada elemento se debe proporcionar un atributo FILL con el ID del gradiente para identificar la estructura o esquema.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>cx</b>	Especifica el punto final del círculo en el eje de abscisas (eje x) para el gradiente radial. Su valor por defecto es el punto central del espacio asignado para la forma dónde se aplica.
<b>cy</b>	Especifica el punto final del círculo en el eje de ordenadas (eje y) para el gradiente radial. Su valor por defecto es el punto central del espacio asignado para la forma dónde se aplica.
<b>r</b>	Especifica el radio de aplicabilidad del gradiente de forma que, el 100% se asigne al perímetro final.
<b>fx</b>	Especifica dónde se situará el punto inicial en el eje de abscisas (eje X) para aplicar el degradado radial.
<bfy< b=""></bfy<>	Especifica dónde se situará el punto inicial en el eje de ordenadas (eje Y) para aplicar el degradado radial.
<b>fr</b>	Especifica el radio del círculo inicial del degradado de forma que el 0% se asigne al perímetro del círculo de inicio.
<b>href</b>	Especifica una referencia a otro degradado lineal que será utilizado como plantilla.
<b>spreadMethod</b>	Especifica si el degradado debe empezar o terminar en los límites de la forma que utiliza o aplica el degradado.
<b>gradientTransform</b>	Especifica la transformación a realizar sobre el degradado. Los funciones y valores válidos son los mismos que los aplicables en CSS.
<b>gradientUnits</b>	Especifica las unidades de medida en las que se deben de basar los ejes de coordenadas donde se aplica el degradado.

### Ejemplo:

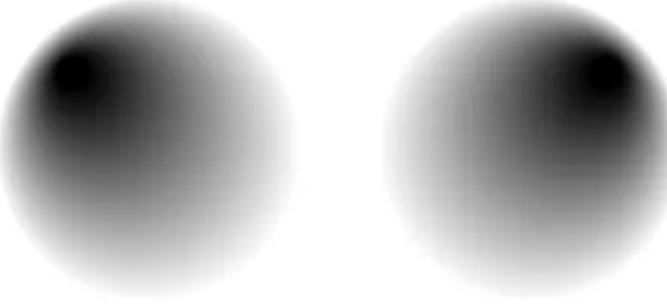
```
<svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%">
<defs>
<radialGradient id="grad45L" fx="0.25" fy="0.25" fr="5%">
<stop offset="0%" stop-color="black"/>
<stop offset="100%" stop-color="white"/>
</radialGradient>
<radialGradient id="grad45R" fx="0.75" fy="0.25" fr="5%">
<stop offset="0%" stop-color="black"/>
```

```

<stop offset="100%" stop-color="white"/>
</radialGradient>
</defs>
<rect x="1" y="1" width="200" height="200" fill="url(#grad45L)" />
<rect x="250" y="1" width="200" height="200" fill="url(#grad45R)" />
</svg>

```

El resultado debería ser algo similar a:



## PATRONES

Un patrón podría definirse como un objeto que se repite de forma secuencial a intervalos regulares a modo de mosaico. Los patrones pueden ofrecer una de las formas más atractivas de generar rellenos en SVG, pero, por la razón que sea, también resultan ser una de las más confusas, por ello, es bueno tratarlos de forma individual e ir construyéndolos capa a capa.

La creación de un patrón puede realizarse mediante trayectorias, superposición de imágenes o, incluso, a través de formas vectoriales y atributos como FILL y STROKE.

Por defecto, los patrones se comportan como la propiedad OVERFLOW con valor HIDDEN de CSS, es decir, si el patrón no puede ser renderizado de manera íntegra, será recortado.

La primitiva PATTERN es la encargada de crear patrones en SVG y, al igual que sucede con los degradados o gradientes, los patrones deben definirse dentro de la primitiva DEFS para poder utilizarlo.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

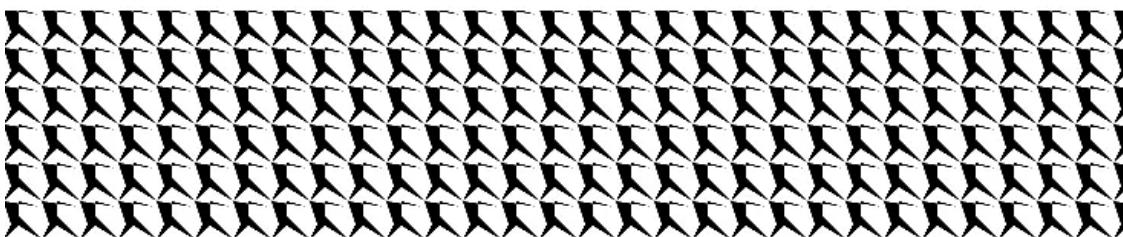
Atributo	Descripción
<b>id</b>	Especifica el nombre del identificador que se utilizará para hacer referencia al patrón.
<b>patternUnits</b>	Especifica sistema de coordenadas se debe utilizar, si al sistema definido por el usuario o al sistema del cuadro delimitador. Los posibles valores son USERSPACEONUSE para establecer que todas las coordenadas utilizadas se refieren al sistema de coordenadas definido por el usuario dentro del patrón y OBJECTBOUNDINGBOX que establece que todas las coordenadas utilizadas se refieren a fracciones o porciones del cuadro delimitador.

	Este último caso hace que las unidades de posicionamiento y tamaño (X, Y, WIDTH y HEIGHT) tengan que venir establecidas en formato decimal (0.00 a 1.00) o en porcentaje.
<b>patterContentUnits</b>	Es idéntico a PATTERNUNITS, con la diferencia de que, si el atributo VIEWBOX está definido, este será ignorado.
<b>patternTransform</b>	Permite que el objeto pueda ser “transformado” de forma idéntica a como lo realiza CSS. Es decir, permite los mismos posibles valores y funciones que permite la propiedad TRANSFORM de CSS.
<b>x</b>	Especifica el desplazamiento desde el borde para el eje de las abscisas.
<b>y</b>	Especifica el desplazamiento desde el borde para el eje de las ordenadas.
<b>width</b>	Especifica el ancho del patrón.
<b>height</b>	Especifica el alto del patrón.
<b>viewbox</b>	Especifica la zona de visión del patrón. Funciona de la misma manera que el atributo VIEWBOX de la primitiva SVG.
<b>href</b>	Especifica o hace referencia a otro modelo cuyos atributos se utilizan como valores por defecto y hereda sus hijos o descendientes. Puede ser recursivo.

### Ejemplo con trayectorias:

```
<svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%">
<defs>
<pattern id="pattern1"
width="25"
height="25"
patternUnits="userSpaceOnUse">
<polygon points="0,0 5,15 0,25 10,16 25,25 10,10 10,0 25,4" />
</pattern>
</defs>
<rect x="1" y="1" width="100%" height="100%" fill="url('#pattern1')"/>
</svg>
```

El resultado debería ser algo similar a:



### Ejemplo con trayectorias 2:

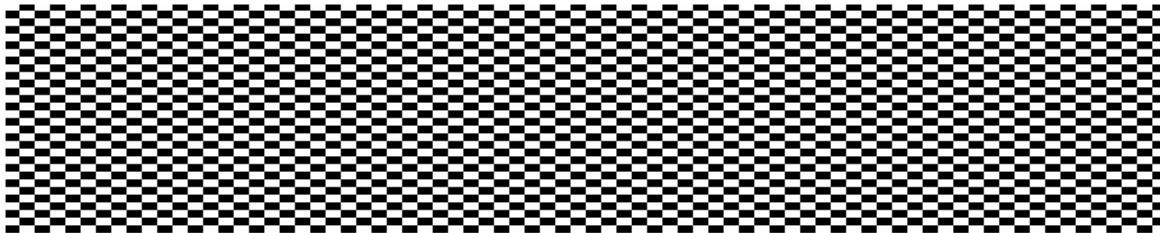
```
<svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%">
<defs>
<pattern id="pattern2"
width="20"
height="10">
```

```

patternUnits="userSpaceOnUse">
<path d="M0,0 L10,0 L10,5 Lo,5z"
fill="#ffffff"
stroke="none" />
<path d="M0,5 L10,5 10,10 Lo,10z"
fill="#oooooooo"
stroke="none" />
<path d="M10,0 L20,0 L20,5 L10,5z"
fill="#oooooooo"
stroke="none" />
<path d="M10,5 L20,5 L20,10 L10,10z"
fill="#ffffff"
stroke="none" />
</pattern>
</defs>
<rect x="1" y="1" width="100%" height="100%" fill="url('#pattern2')"/>
</svg>

```

El resultado debería ser algo similar a:



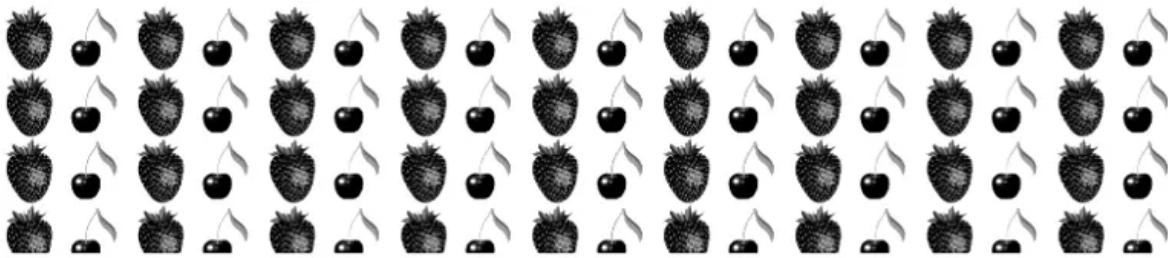
### Ejemplo con imágenes:

```

<svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%">
<defs>
<pattern id="pattern3"
width="80"
height="40"
patternUnits="userSpaceOnUse">
<image x="0"
y="0"
width="40"
height="40"
href="fresa.png" />
<image x="40"
y="0"
width="40"
height="40"
href="cereza.png" />
</pattern>
</defs>
<rect x="1" y="1" width="100%" height="100%" fill="url('#pattern3')"/>
</svg>

```

El resultado debería ser algo similar a:



## ANIMACIONES

SVG nos proporciona una manera sencilla de realizar animaciones, sin embargo, antes de adentrarnos en esta sección es crucial entender la diferencia que existe entre acción y movimiento.

Mientras que un movimiento conlleva un desplazamiento o evolución, una acción, sólo implica un cambio, transformación o transición. Por ejemplo, a la modificación del color en una forma después de haberse cargado y renderizado se le denomina acción ya que, dicha forma se mantiene en una posición fija. Tanto las acciones, como los movimientos pueden aplicarse independientemente o a través de eventos concretos como puedan ser click, mouseover, hover, etcétera.

Como característica añadida, SVG disponen de eventos específicos, los cuales puede utilizarse para enriquecer las animaciones y transiciones. Los eventos más recurrentes quizás sean ONBEGIN, ONEND y ONREPEAT, sin embargo, SVG presenta multitud de eventos que no son utilizados con mucha frecuencia, pero que nos pueden ayudar a crear buenas presentaciones.

En lo referente a los elementos disponibles, presenta cinco elementos para crear y manipular animaciones. No obstante, algunos de ellos no son totalmente compatibles con todos los navegadores y, alguno de ellos (como ANIMATECOLOR), se encuentra obsoleto.

### Attributos comunes a todos los elementos de animación

A continuación, se presentan los atributos que pueden utilizarse en los elementos de ANIMATE, ANIMATECOLOR, ANIMATEMOTION, ANIMATETRANSFORM y SET. Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>accumulate</b>	Especifica si la animación es o no acumulativa. Esto es útil si lo que se desea realizar es una animación que se construye en base a los valores predecesores aumentando en cada iteración. Entre sus posibles valores podemos encontrar SUM para indicar que la animación se utilizará el último valor conocido de la iteración anterior y, NONE para indicar que la animación no es acumulativa.

	Este atributo será ignorado si el atributo FROM está exento en la definición de la animación. Por defecto, su valor es NONE.
<b>additive</b>	Especifica si la animación es o no aditiva. Esto es útil si lo que se desea realizar es una animación que utiliza valores añadidos o desplazamientos a través de valor delta constante. Entre sus posibles valores podemos encontrar SUM para indicar que la animación irá añadiendo un valor subyacente y otras animaciones de menor relevancia o prioridad y, REPLACE para indicar que la animación irá anulando el valor subyacente y demás animaciones de menor relevancia o prioridad. Este atributo puede tener comportamientos inesperados si se producen animaciones concatenadas dependientes unas de otras. Por defecto, su valor es REPLACE.
<b>attributeName</b>	Especifica el nombre de la propiedad CSS que será utilizada en la animación, transición o transformación.
<b>begin</b>	Especifica en qué momento debe empezar la animación Sus posibles valores son un valor en segundos o la palabra reservada INDEFINITE. Por defecto, su valor es 0.
<b>calcMode</b>	Especifica el modo en el que interpolarán los píxeles para generar la animación. Entre sus posibles valores podemos encontrar DISCRETE, para indicar que la animación se realizará sin realizar ninguna interpolación, LINEAR para indicar que la animación se realizará mediante interpolaciones simples, PACED para indicar que la animación se realizará de manera uniforme y, SPLINE para indicar que la animación se realizará utilizando una función de tiempo definida por una curva cúbica de Bézier. Por defecto, su valor es LINEAR.
<b>dur</b>	Especifica la duración de la animación. Entre sus posibles valores, se puede establecer cualquier valor de tiempo solos o combinados (horas, minutos, segundos o milisegundos) o las palabras clave MEDIA y INDEFINITE. La palabra clave MEDIA sólo es válida en aquellos elementos que definen los medios.
<b>end</b>	Especifica en qué momento debe terminar la animación. Sus posibles valores se suelen expresar en segundos y, por defecto, no tiene ningún valor establecido.
<b>fill</b>	Especifica el estado final en el que debe permanecer la animación. Entre sus posibles valores podemos encontrar FREEZE que provoca que se mantenga en su estado o valor final de la animación y, REMOVE que provoca que la animación vuelva a su estado inicial.
<b>from</b>	Especifica el valor inicial del atributo que se utilizará en la animación. Lo normal es usarlo de manera conjunta con el atributo TO y sólo tiene sentido su uso cuando el atributo ATTRIBUTENAME está establecido.
<b>keyTimes</b>	Especifica una lista de valores de tiempo que serán utilizados para controlar el ritmo de la animación. Los posibles valores para este atributo vienen establecidos en tanto por uno, siendo 1, el desplazamiento proporcional a duración de la animación.
<b>keySplines</b>	Especifica un conjunto de valores de control aplicables a la curva de Bézier que está asociada con el atributo KEYTIMES, y que define la función Bézier cúbica que controla el ritmo de intervalo. Este atributo es ignorado a menos que el atributo CALCMODE esté establecido a SPLINE.
<b>max</b>	Especifica el tiempo mínimo que debe durar la animación.
<b>min</b>	Especifica el tiempo máximo que debe durar la animación.

<b>restart</b>	Especifica si una animación puede reiniciarse o no. Sus posibles valores son ALWAYS para indicar que se puede reiniciar en cualquier momento, WHENNOTACTIVE para indicar que sólo se puede reiniciar cuando la animación esté inactiva y NEVER para indicar que no es posible el reinicio.
<b>repeatcount</b>	Especifica el número de veces que repetirá la animación.
<b>repeatDur</b>	Especifica la duración durante la cual la animación podrá repetirse. Entre sus posibles valores, se puede establecer cualquier valor de tiempo solos o combinados (horas, minutos, segundos o milisegundos) o la palabra reservada INDEFINITE. No presenta ningún valor por defecto.
<b>to</b>	Especifica el valor final del atributo que se utilizará en la animación. Lo normal es usarlo de manera conjunta con el atributo BEGIN y sólo tiene sentido su uso cuando el atributo ATTRIBUTERENAME está establecido.
<b>values</b>	Especifica los valores que se tomarán durante el transcurso de la animación.

## Primitiva animate

La primitiva ANIMATE permite realizar una animación que requiere de una o varias acciones, pero que no implican ningún movimiento forzosamente. En otras palabras, permite animar uno o varios atributos a lo largo del tiempo.

En general, se suele utilizar para realizar transiciones o transformaciones sobre alguno de los atributos que definen la forma u objeto.

### Ejemplo de transición sencilla:

```
<svg viewBox="0 0 200 200" width="200" height="200">
<circle cx="100" cy="100" r="0" fill="black">
<animate attributeName="r"
values="0;50;0"
dur="5s"
repeatCount="indefinite"/>
</circle>
</svg>
```

El resultado debería ser un círculo negro situado en el centro del tapiz o de la ventana que cambia de tamaño de manera constante e indefinida, desde 0 hasta 50 píxeles y, una vez en su tamaño máximo, vuelve a 0.

### Ejemplo de desplazamiento sencillo:

```
<svg viewBox="0 0 200 200" width="200" height="200">
<circle cx="-25" cy="100" r="25" fill="black">
<animate attributeName="cx"
from="-25"
to="225"
dur="2s"
repeatCount="indefinite"/>
```

```
</circle>
</svg>
```

El resultado debería ser el desplazamiento de un círculo negro que se desplaza de manera indefinida desde fuera de la ventana en el extremo izquierdo, hasta que desaparece por completo en el extremo derecho.

### Ejemplo de desplazamiento con rebote:

```
<svg viewBox="0 0 200 200" width="200" height="200">
<circle cx="100" cy="25" r="25" fill="black">
<animate attributeName="cy"
from="-25"
to="225"
dur="0.5s"
begin="click"
values="50; 175; 135; 175"
keyTimes="0; 0.60; 0.80; 1"
fill="freeze" />
</circle>
</svg>
```

El resultado debería ser el desplazamiento de un círculo negro que cae de arriba hacia abajo con un rebote al final. La animación puede repetirse las veces que se deseé puesto que lleva declarado un evento click que hace que se repita la animación cuando se pulsa en la figura.

### Ejemplo de transición en el borde y fondo de una figura:

```
<svg viewBox="0 0 200 200" width="200" height="200">
<rect x="50"
y="50"
width="100"
height="100"
rx="5"
ry="5"
style="fill:darkgreen; stroke: darkgreen;
stroke-width: 3; stroke-dasharray: 10, 5;">
<animate attributeName="fill"
dur="5s"
values="black:white:black"
repeatCount="indefinite"/>
<animate attributeName="stroke-dasharray"
dur="5s"
values="5,5; 6,5; 7,5; 6,5; 5,5"
repeatCount="indefinite"></animate>
</rect>
</svg>
```

El resultado debería ser una figura cuadrada que presenta un fondo que se degrada de forma permanente desde blanco a negro y viceversa. Además, el borde de tipo matriz de guiones va

aumentado y disminuyendo en función del tiempo.

### Ejemplo de cambio de forma:

```
<svg viewBox="0 0 200 200" width="200" height="200">
<rect x="50"
y="50"
width="100"
height="100"
rx="5"
ry="5">
<animate attributeName="rx"
dur="2.5s"
values="0; 150; 0"
begin="mouseover"/>
<animate attributeName="ry"
dur="2.5s"
values="0; 150; 0"
begin="mouseover"/>
</rect>
</svg>
```

El resultado debería ser una figura cuadrada de 100x100 situada en el centro del tapiz que cambia de forma de manera suave durante los dos segundos y medio que dura la animación.

### Primitiva set

La primitiva SET establece o modifica el valor de un atributo una vez haya pasado un tiempo específico. Esta primitiva permite realizar cambios de forma similar a como lo hace ANIMATE, con la diferencia de que no es aditiva, lo que significa que los cambios puede que se realicen de manera suave y continuada.

Po ende, el elemento SET no tiene los atributos ADDITIVE y ACCUMULATE disponibles para configurar las posibles animaciones.

Para entender mejor las diferencias entre ANIMATE y SET, veamos el último ejemplo expuesto de ANIMATE, pero ejecutando los cambios a través de SET.

### Ejemplo de cambio de forma:

```
<svg viewBox="0 0 200 200" width="200" height="200">
<rect x="50"
y="50"
width="100"
height="100"
rx="5"
ry="5">
<set attributeName="rx"
dur="2.5s"
to="150"
```

```

begin="mouseover"/>
<set attributeName="ry"
dur="2.5s"
to="150"
begin="mouseover"/>
</rect>
</svg>

```

El resultado debería ser una figura cuadrada de 100x100 situada en el centro del tapiz que pasa a ser un círculo de repente cuando pasamos el ratón por encima de la figura.

## **Primitiva animatemotion**

La primitiva ANIMATEMOTION establece una manera de crear un movimiento a través de una ruta definida por el atributo PATH.

También es posible realizar el movimiento a través de la primitiva MPATH, la cual permite reutilizar una ruta definida anteriormente mediante PATH.

### **Ejemplo de movimiento a través de una figura:**

```

<svg viewBox="0 0 200 200" width="200" height="200">
<rect x="50"
y="50"
width="100"
height="100"
rx="5"
ry="5" />
<circle r="5" fill="orange">
<animateMotion dur="10s"
repeatCount="indefinite"
path="M50,50 T100,50 150,50 150,150 M150,150 T50,150 M50,150 T50,50" />
</circle>
</svg>

```

El resultado debería ser una figura cuadrada redondeada de 100x100 píxeles situada en el centro del tapiz que presenta círculo naranja moviéndose alrededor de sus límites fronterizos.

Cabe destacar que esta primitiva no es compatible con Internet Explorer.

## **Primitiva animatetransform**

La primitiva ANIMATEMOTION establece una manera de crear un movimiento a través de una ruta definida por el atributo PATH.

También es posible realizar el movimiento a través de la primitiva MPATH, la cual permite reutilizar una ruta definida anteriormente mediante PATH.

### **Ejemplo de giro:**

```

<svg viewBox="0 0 200 200" width="200" height="200">
<g transform="translate(100, 100)">
<rect x="-50"
y="-50"
width="100"
height="100"
rx="5"
ry="5">
<animateTransform attributeName="transform"
type="rotate"
from="0"
to="180"
begin="0"
dur="5s"
repeatCount="indefinite" />
</rect>
</g>
<circle r="5" fill="orange" cx="100" cy="100">
</svg>

```

El resultado debería ser una figura cuadrada centrada en el tapiz que gira de manera constante por debajo de un círculo naranja.

Cabe destacar el atributo TYPE puede contener los posibles valores de la propiedad TRANSFORM de CSS y que, esta primitiva, no es compatible con Internet Explorer.

## FILTROS

### Antes de nada

Siempre que se trabaja con filtros SVG se debe indicar el origen para tratarlo, no obstante, a veces también se requiere el destino. Estos atributos tienen asignados los nombres IN e IN2 y sus posibles valores son:

<i>Valor atributo IN / IN2</i>	<i>Descripción</i>
<b>SourceGraphic</b>	Representa al elemento gráfico origen. En general, tanto la propiedad IN, como la propiedad IN2, tendrán el valor por defecto establecido a SOURCEGRAPHIC si es el primer o único filtro aplicable. En cualquier otro caso, normalmente será el resultado del filtro anterior.
<b>SourceAlpha</b>	Representa al elemento gráfico origen. Es idéntica a SOURCEGRAPHIC, pero sólo utilizando el canal alfa.
<b>BackgroundImage</b>	Representa al elemento gráfico destino.

<b>BackgroundAlpha</b>	Representa al elemento gráfico destino. Es idéntica a BACKGROUNDIMAGE, pero sólo utilizando el canal alfa.
<b>FillPaint</b>	Representa al valor de la propiedad FILL. No suele tener efecto si el resultado de la propiedad FILL no dispone de transparencias.
<b>StrokePaint</b>	Representa al valor de la propiedad STROKE. No suele tener efecto si el resultado de la propiedad STROKE no dispone de transparencias.
<b>ID personalizado</b>	Representa al objeto que tenga como identificador indicado en su atributo RESULT. Esto es útil cuando se realizan composiciones o combinaciones de filtros.

## Compensaciones: primitiva feIOffset

La primitiva FEOFFSET permite trasladar o mover las imágenes a través de la ventana del objeto, normalmente, con el objetivo de reorganizar su contenido.

Aunque puede aplicarse a cualquier entrada, lo habitual es que esté asociado a las entradas referenciadas por SOURCEGRAPHIC y BACKGROUNDIMAGE.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>in</b>	Especifica el objeto al que aplicar el filtro.
<b>dx</b>	Especifica el desplazamiento horizontal o a lo largo del eje X expresado en el sistema de coordenadas establecido por el atributo PRIMITIVEUNITS de la primitiva FILTER. Por defecto, su valor es 0.
<b>dy</b>	Especifica el desplazamiento vertical o a lo largo del eje Y expresado en el sistema de coordenadas establecido por el atributo PRIMITIVEUNITS de la primitiva FILTER. Por defecto, su valor es 0.

### Ejemplo:

```
<svg width="400" height="300" viewBox="0 0 400 300">
<defs>
<filter id="offset">
<feOffset dx="5" dy="5" result="offsetBlur" />
</filter>
</defs>
<rect x="10"
y="10"
width="200"
height="150"
fill="#888"
filter="url(#offset)" />
</svg>
```

El resultado de aplicar este filtro es, simplemente, un desplazamiento de 5 píxeles tanto en la horizontal, como en la vertical de su posicionamiento original.

## Inundar: primitiva feFlood

La primitiva FEFLOOD permite inundar o colorear una zona concreta del tapiz o pantalla y, entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>x</b>	Especifica el comienzo en el eje de las abscisas o eje de las X.
<b>y</b>	Especifica el comienzo en el eje de las ordenadas o eje de las Y.
<b>width</b>	Especifica el ancho de la forma.
<b>height</b>	Especifica el alto de la forma.
<b>flood-color</b>	Especifica el color de relleno.
<b>flood-opacity</b>	Especifica el grado de transparencia del color. Por defecto es 1.
<b>result</b>	Especifica el identificador para el resultado. Es útil cuando se desean combinar diferentes resultados de efectos o filtros.

### Ejemplo:

```
<svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%">
<defs>
<filter id="floodFilter" filterUnits="userSpaceOnUse">
<feFlood x="50"
y="50"
width="100"
height="100"
flood-color="#oooooooo"
flood-opacity="1" />
</filter>
</defs>
<use style="filter: url(#floodFilter);"></use>
</svg>
```

## Imágenes: primitiva feImage

La primitiva de filtro FEIMAGE aplica un gráfico externo al elemento de filtro que es renderizado y que, finalmente, se convierte en el resultado de la primitiva de filtro.

Puede hacer referencia a una imagen externa o puede ser una referencia interna o externa de otra pieza de SVG. En general, se produce una imagen similar a la incorporada en el SOURCEGRAPHIC de la imagen de entrada, excepto que el gráfico proviene de una fuente externa.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>x</b>	Indica el comienzo en el eje de las abscisas o eje de las X. Su valor por defecto es 0.
<b>y</b>	Indica el comienzo en el eje de las ordenadas o eje de las Y. Su valor por defecto es 0.
<b>width</b>	Indica el ancho de la forma. Su valor por defecto es 0.
<b>height</b>	Indica el alto de la forma. Su valor por defecto es 0.
<b>href</b>	Especifica la URL dónde está localizada o ubicada la imagen.
<b>preserveAspectRatio</b>	Especifica que se debe forzar una escalabilidad uniforme y, en cuyo caso, el método de alineación que se debe usar para en la ventana gráfica o VIEWBOX. Entre los posibles valores podemos encontrar todas las variaciones en MAX, MIN y MED para cada valor de X o Y. Por ejemplo, la especificación XMINYMIN establece que se alinee con el valor más pequeño del eje X y con el valor 0 punto medio del eje Y. Admite un parámetro opcional que indica si se debe permitir que la imagen sea cortada (SLICE) o debe ser mostrada de forma íntegra (MEET).
<b>result</b>	Especifica el identificador para el resultado. Es útil cuando se desean combinar diferentes resultados de efectos o filtros.

### Ejemplo:

```

<svg viewBox="0 0 200 200"
      xmlns="http://www.w3.org/2000/svg">
  <defs>
    <filter filterUnits="userSpaceOnUse"
           x="10" y="10"
           width="190" height="190">
      <feImage result="pict2"
              x="0" y="0"
              width="100%" height="100%"
              href='./img/imagen-2.png' />
    </filter>
  </defs>
  <image x="0" y="0"
         width="100%" height="100%"
         href='./img/imagen-1.png'
         filter="url(#feImage1)" />
</svg>
```

El resultado de ejecutar este código es que, aunque se declara la “imagen-1”, sólo se mostrará la “imagen-2”. En otras palabras, es como si se superpusiera la “imagen-2” encima de la “imagen-1”.

### Azulejos: primitiva feTile

De la misma manera en que la primitiva FEFLOOD aplica a elementos gráficos básicos como RECT o CIRCLE, y la primitiva FEIMAGE aplica a imágenes, SVG dispone de una primitiva para gestionar patrones.

La utilidad de este filtro es que nos permite reutilizar los patrones repetitivos generados a través de la primitiva **PATTERN** para generar efectos.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>x</b>	Indica el comienzo en el eje de las abscisas o eje de las X. Su valor por defecto es 0.
<b>y</b>	Indica el comienzo en el eje de las ordenadas o eje de las Y. Su valor por defecto es 0.
<b>width</b>	Indica el ancho de la forma. Su valor por defecto es 0.
<b>height</b>	Indica el alto de la forma. Su valor por defecto es 0.
<b>result</b>	Especifica el identificador para el resultado. Es útil cuando se desean combinar diferentes resultados de efectos o filtros.

### Ejemplo:

```
<svg width="400" height="200" viewBox="400 200">
<defs>
<filter id="tile" primitiveUnits="objectBoundingBox">
<feImage href="zen-circle.png"
x="0" y="0"
width="0.25" height="0.5" />
<feTile />
</filter>
</defs>
<rect width="100%" height="100%" style="filter:url(#tile);"></rect>
</svg>
```

El resultado de ejecutar este código debería ser similar a lo siguiente:



### Fusionar: primitiva **feBlend**

La primitiva **FEBLEND** realiza, pixel a pixel, la combinación entre dos imágenes de entrada. Los modos de fusión que se pueden utilizar son: normal, Multiplicar, pantalla, oscurecer y aclarar.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>in</b>	Especifica el objeto origen.
<b>in2</b>	Especifica el objeto destino.
<b>mode</b>	Especifica el tipo de fusión a realizar y tiene, como posibles, valores NORMAL, MULTIPLY, SCREEN, DARKEN y LIGHTEN.
<b>result</b>	Especifica el identificador para el resultado. Es útil cuando se desean combinar diferentes resultados de efectos o filtros.

Cabe destacar que, sea cual sea la modalidad de fusión, el resultado de la opacidad se calcula a través de la siguiente fórmula:

$$qr = 1 - (1 - Qa) * (1 - Qb)$$

En donde:

- $QR$  es el valor de la opacidad resultante.
- $QA$  es el valor de la opacidad de un píxel dado por la imagen A.
- $QB$  es el valor de la opacidad de un píxel dado por la imagen B.

## MODO NORMAL

El modo NORMAL, permite que el elemento referenciado por IN2 sólo sea visible si el elemento referenciado por IN contiene transparencia. El color resultante se calcula a través de la fórmula:

$$CR = (1 - Q) * CB + CA$$

En donde:

- $CR$  es el color resultante RGB.
- $Q$  es el valor de la opacidad de un píxel dado por la imagen.
- $CA$  es el color (RGB) en un píxel dado por la imagen A.
- $CB$  es el color (RGB) en un píxel dado por la imagen B.

Supongamos dos rectángulos idénticos superpuestos de la siguiente manera:



Para conseguir este supuesto, podríamos hacer algo como lo siguiente:

```

<svg viewBox="0 0 400 400" xmlns="http://www.w3.org/2000/svg">
<defs>
<filter id="flood-blend"
filterUnits="userSpaceOnUse"
x="0"
y="0"
width="100%"
height="100%">
<feFlood x="10"
y="10"
width="300"
height="100"
flood-color="#888"
flood-opacity="1"
result="figure1" />
<feFlood x="30"
y="30"
width="300"
height="100"
flood-color="#888"
flood-opacity="1"
result="figure2" />
<feBlend mode="normal" in="figure1" in2="figure2"></feBlend>
</filter>
</defs>
<use style="filter: url(#flood-blend);"></use>
<text x="100" y="220" dominant-baseline="middle" text-anchor="middle">
mode="normal"
</text>
</svg>

```

Cabe destacar que, el modo NORMAL, es equivalente al valor OVER de la primitiva FECOMPOSITE. Además, coincide con el método de mezcla usada por FEMERGE y con la técnica ALFA COMPOSITING SIMPLE que se usa en SVG para toda la composición exterior de efectos de filtro.

## MODO MULTIPLY

El modo MULTIPLY, resta los valores de brillo de las dos imágenes en cada punto. El color resultante se calcula a través de la fórmula:

$$CR = (1 - QA) * CB + (1 - QB) * CA + CA * CB$$

En donde:

- $CR$  es el color resultante RGB.
- $QA$  es el valor de la opacidad de un píxel dado por la imagen A.

- $QB$  es el valor de la opacidad de un píxel dado por la imagen B.
- $CA$  es el color (RGB) en un píxel dado por la imagen A.
- $CB$  es el color (RGB) en un píxel dado por la imagen B.

Si estableciésemos el código anterior, pero con el modo de la primitiva FBLEND establecido al valor MULTIPLY, el resultado debería ser algo como la imagen mostrada a la derecha:



## MODO SCREEN

El modo SCREEN, suma los valores de brillo de las dos imágenes en cada punto. El color resultante se calcula a través de la fórmula:

$$CR = CB + CA - CA * CB$$

En donde:

- $CR$  es el color resultante RGB.
- $CA$  es el color (RGB) en un píxel dado por la imagen A.
- $CB$  es el color (RGB) en un píxel dado por la imagen B.

Si estableciésemos el código anterior, pero con el modo de la primitiva FBLEND establecido al valor SCREEN, el resultado debería ser algo como la imagen mostrada a la derecha:



## MODO DARKEN

El modo DARKEN, extrae el valor más oscuro de las dos imágenes en cada punto. El color resultante se calcula a través de la fórmula:

$$CR = \text{Min}((1 - QA) * CB + CA, (1 - QB) * CA + CB)$$

En donde:

- $CR$  es el color resultante RGB.

- $QA$  es el valor de la opacidad de un píxel dado por la imagen A.
- $QB$  es el valor de la opacidad de un píxel dado por la imagen B.
- $CA$  es el color (RGB) en un píxel dado por la imagen A.
- $CB$  es el color (RGB) en un píxel dado por la imagen B.

Si estableciésemos el código anterior, pero con el modo de la primitiva FBLEND establecido al valor DARKEN, el resultado debería ser algo como la imagen mostrada a la derecha:



Aunque en este ejemplo no es posible ver la diferencia, si en vez de ser dos rectángulos grises, fuesen uno rojo y otro azul, la intersección entre ambos sería negro.

## MODO LIGHTEN

El modo LIGHTEN, extrae el valor más claro de las dos imágenes en cada punto. El color resultante se calcula a través de la fórmula:

$$CR = \text{Max}((1 - QA) * CB + CA, (1 - QB) * CA + CB)$$

En donde:

- $CR$  es el color resultante RGB.
- $QA$  es el valor de la opacidad de un píxel dado por la imagen A.
- $QB$  es el valor de la opacidad de un píxel dado por la imagen B.
- $CA$  es el color (RGB) en un píxel dado por la imagen A.
- $CB$  es el color (RGB) en un píxel dado por la imagen B.

Si estableciésemos el código anterior, pero con el modo de la primitiva FBLEND establecido al valor LIGHTEN, el resultado debería ser algo como la imagen mostrada a la derecha:



Aunque en este ejemplo no es posible ver la diferencia, si en vez de ser dos rectángulos grises, fuesen uno rojo y otro azul, la intersección entre ambos sería rosa.

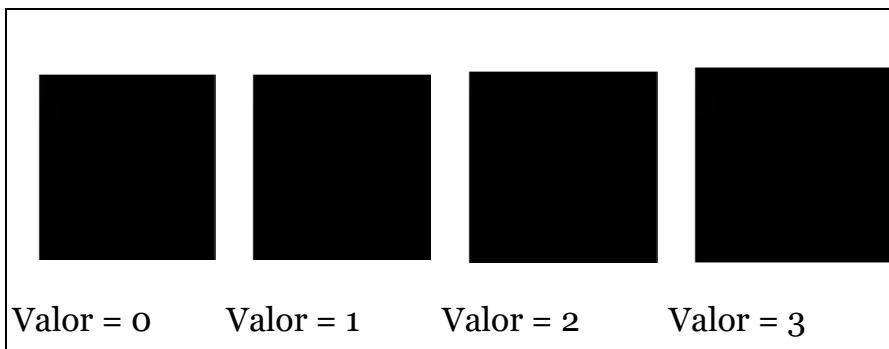
## Desenfoque gausiano: primitiva feGaussianBlur

La primitiva FEGAUSSIANBLUR realiza, pixel a pixel, una mezcla ligera de los colores vecinos, lo que genera que la imagen pierda nitidez provocando una suavidad en los alrededores de cada pixel. Podría decirse que es como cuando se toma una fotografía desenfocada.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>in</b>	Especifica el objeto al que aplicar el filtro. Define dónde y cómo se realizará la operación de desenfoque.
<b>stdDeviation</b>	Especifica la desviación estándar para la operación de desenfoque.
<b>result</b>	Especifica el identificador para el resultado. Es útil cuando se desean combinar diferentes resultados de efectos o filtros.

A continuación, se muestra un ejemplo un ejemplo de cómo afecta el valor del atributo STDDEVIATION en una primitiva RECT.



### Ejemplo:

```
<svg viewBox="0 0 200 200"
xmlns="http://www.w3.org/2000/svg">
<defs>
<filter id="blur">
<feGaussianBlur in="SourceGraphic" stdDeviation="2" />
</filter>
</defs>
<rect x="10" y="10" width="75" height="75" filter="url('#blur')"/>
</svg>
```

## Filtro matrices de color: primitiva feColorMatrix

La primitiva FECOLORMATRIX realiza una recodificación de colores dentro de una imagen, basándose en la capacidad de multiplicar los niveles de cada píxel por coeficientes numéricos. Esta recodificación se realiza a través de producto de matrices, no obstante, la primitiva FECOLORMATRIX cuenta con unas matrices predefinidas que se han asociado a los filtros

comunes de saturación, cambio de color y negativo, por lo que algunos de sus atributos sólo requieren de un valor entero o decimal.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>in</b>	Especifica el objeto al que aplicar el filtro.
<b>type</b>	Especifica el tipo de matriz de color a utilizar, sea o no predefinida.
<b>values</b>	Especifica la matriz de valores a aplicar. Si el tipo es SATURATE, HUEROTATE o LUMINANCETOALPHA, este atributo se alimenta de valores enteros o decimales.
<b>result</b>	Especifica el identificador para el resultado. Es útil cuando se desean combinar diferentes resultados de efectos o filtros.

A continuación, se muestra una tabla con los distintos valores que puede tomar la propiedad TYPE:

Tipo	Descripción
<b>matrix</b>	Utiliza la matriz pasada a través del atributo VALUES compuesta de 20 valores separados por espacios y que se interpretan como 4 vectores de 5 elementos cada uno. El primer vector será para definir los rojos, el segundo definirá los verdes, el tercero los azules y el cuarto se utilizará como valor alfa.
<b>saturate</b>	Sólo necesita de un número real comprendido entre 0 y 1 pasado a través del atributo VALUES. El valor 0 se interpretará como desaturar y 1 como la imagen con el color original.
<b>huerotate</b>	Sólo necesita un valor comprendido entre 0 y 360 pasado a través del atributo VALUES. El valor de 0 no tendrá efecto en la imagen y el valor 180 será el equivalente a invertir los colores.
<b>luminancetoalpha</b>	Hace el efecto de negativo o inversión de colores. El atributo VALUES no es aplicable.

Cuando el tipo se trabaja con el tipo MATRIX, generalmente se define una de 4x5 valores, en dónde el resultado para cada píxel viene determinado por una multiplicación de esa matriz por el vector que conforma el color a tratar. Para verlo algo más claro, veamos un ejemplo:

Supongamos que se desea ver el resultado de multiplicar un píxel rojo puro sin transparencia con una matriz que contiene los siguientes coeficientes:

R	G	B	A	F	
1	0	0	0	0	R
0	1	0	0	0	G
0	0	1	0	0	* B

	o	o	o	1	o	A
--	---	---	---	---	---	---

Cada columna de esta matriz de 4x5 representa un coeficiente que se multiplicará por el vector que conforman los valores del color a tratar. Por tanto, si realizamos el producto tendremos el siguiente resultado:

$$R' = 1 * 255 + 0 * 0 + 0 * 0 + 0 * 1 + 0 = 255$$

$$G' = 0 * 255 + 1 * 0 + 0 * 0 + 0 * 1 + 0 = 0$$

$$B' = 0 * 255 + 0 * 0 + 1 * 0 + 0 * 1 + 0 = 0$$

$$A' = 0 * 255 + 0 * 0 + 0 * 0 + 1 * 1 + 0 = 1$$

Como se puede observar, el color resultante es exactamente el mismo que el original, sin embargo, si invertimos los coeficientes de la matriz de las columnas RGB, lo que tendremos es algo muy diferente.

R	G	B	A	F		
0	0	1	0	0		R
0	1	0	0	0		G
1	0	0	0	0	*	B
0	0	0	1	0		A

$$R' = 0 * 255 + 0 * 0 + 1 * 0 + 0 * 1 + 0 = 0$$

$$G' = 0 * 255 + 1 * 0 + 0 * 0 + 0 * 1 + 0 = 0$$

$$B' = 1 * 255 + 0 * 0 + 1 * 0 + 0 * 1 + 0 = 255$$

$$A' = 0 * 255 + 0 * 0 + 0 * 0 + 1 * 1 + 0 = 1$$

Esto es lo que se suele llamar filtro de inversión de colores. Como se puede observar en el ejemplo, lo que antes era rojo, ahora es azul y, si hiciésemos la operación correspondiente, lo que antes era azul, ahora es rojo.

## EJEMPLO DE MATRIZ DE COLOR CON FONDO INVERTIDO

```
<filter id="bgInvert">
<feColorMatrix type="matrix"
values="1 0 0 0 0
      0 1 0 0 0
      0 0 1 0 0
      0 0 0 1 1" />
</filter>
```

## EJEMPLO DE CONVERSIÓN A ESCALA DE GRISES

```
<filter id="grayScale">
<feColorMatrix type="saturate"
values="0">
</feColorMatrix>
</filter>
```

## EJEMPLO DE CONVERSIÓN A ESCALA DE GRISES CON ALTO CONTRASTE

```
<filter id="grayScaleHighContrast">
<feColorMatrix type="matrix"
values="0 0 0 -1 0
0 0 0 -1 0
0 0 0 -1 0
-1 0 0 1 0">
</feColorMatrix>
</filter>
</svg>
```

## EJEMPLO DE ROTACIÓN DE COLOR

```
<filter id="hueRotate">
<feColorMatrix type="hueRotate" values="180"></feColorMatrix>
</filter>
```

## EJEMPLO DE IMAGEN SOBRESATURADA:

```
<filter id="overSaturate">
<feColorMatrix type="matrix"
values=" 1.5 -0.25 -0.25 0 0
-0.25 1.5 -0.25 0 0
-0.25 -0.25 1.5 0 0
0 0 0 1 0">
</feColorMatrix>
</filter>
```

## EJEMPLO DE CONVERSIÓN A NEGATIVO:

```
<filter id="negative">
<feColorMatrix type="luminanceToAlpha"
values="" />
</filter>
```

## Componentes de transferencia: primitiva feComponentTransfer

La primitiva FECOMPONENTTRANSFER realiza una recodificación independiente de cada uno de los cuatro canales de color: R, G, B, y A (alfa). Permite realizar operaciones como el ajuste del brillo, el contraste, equilibrio de color o umbral través de diferentes funciones que son aplicables a uno o varios canales de una imagen.

La primitiva FECOMPONENTTRANSFER se define a través de sus métodos FEFUNCR, FEFUNCG, FEFUNCB Y FEFUNCA que crean y agregan una función de transferencia para cada canal R G B A y, cada una de ellas puede tener, además del indicador de tipo, uno o varios parámetros.

Los cálculos se realizan sobre los valores de color no tratados o premultiplicados. Si los gráficos de entrada se componen de valores de color premultiplicados, se restauran automáticamente a los valores de color no premultiplicados para esa operación.

Además, se debe tener en cuenta que es posible deshacer y rehacer de la premultiplicación si FEFUNCA se establece a IDENTITY y todos los valores alfa de la fuente gráfico se establecen a 1.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>in</b>	Especifica el objeto al que aplicar el filtro. Por defecto, tiene establecido el valor SOURCEGRAPHIC.
<b>type</b>	Especifica la función a utilizar.
<b>tablevalues</b>	Especifica una lista de valores separados por espacios o comas, siempre y cuando el atributo TYPE no sea IDENTITY.
<b>slope</b>	Indica la pendiente de la función lineal. Por defecto, si no se especifica, tiene un valor de 1. En general tendrá un valor numérico excepto para el tipo linear que será una función.
<b>intercept</b>	Indica el intercepto de la función lineal. Por defecto, si no se especifica, tiene un valor de 0. En general tendrá un valor numérico excepto para el tipo linear que será una función.
<b>amplitude</b>	Indica la amplitud de la función gamma. Por defecto, si no se especifica, tiene un valor de 1. En general tendrá un valor numérico excepto para el tipo gamma que será una función.
<b>exponent</b>	Indica el exponente de la función gamma. Por defecto, si no se especifica, tiene un valor de 1. En general tendrá un valor numérico excepto para el tipo gamma que será una función.
<b>offset</b>	Indica el desplazamiento de la función gamma. Por defecto, si no se especifica, tiene un valor de 0. En general tendrá un valor numérico excepto para el tipo gamma que será una función.
<b>result</b>	Especifica el identificador para el resultado. Es útil cuando se desean combinar diferentes resultados de efectos o filtros.

A continuación, se muestra una tabla con los distintos tipos de componentes de transferencia:

Tipo	Descripción
<b>identity</b>	El componente resultante es el mismo que el original.
<b>discrete</b>	Se define como una función de paso determinado entre los valores dados en el atributo TABLEVALUES. La lista tiene n valores referenciados de 0 a N-1 que identifica la función de paso que consiste en n pasos. Se puede utilizar para Posterizar una imagen (que es reducirlo a valores de menor color).
<b>linear</b>	Se define como una ecuación lineal que utiliza como parámetros la pendiente y el intercepto.

	Cambiando sólo el atributo SLOPE de los canales R G B, es posible gestionar el brillo oscureciendo el gráfico si su valor está comprendido entre 0 y 1.0 e iluminando si su valor está comprendido entre 1.0 y 2.0.
<b>table</b>	Se define como una interpolación lineal entre los valores dados en el atributo TABLEVALUES. La tabla tiene n+1 valores referenciados de 0 a N que especifican los valores de inicio y fin para n regiones de interpolación de tamaño uniforme.
<b>gamma</b>	Se define como una ecuación exponencial que utiliza como parámetros la amplitud, el exponente y el offset. Cambiando sólo el exponente de los canales R G B podemos gestionar el contraste si sus valores son mayores que 1.

## EJEMPLO DE POSTERIZACIÓN

```
<filter id="posterization">
<feComponentTransfer>
<feFuncR type="discrete"
tableValues="0 .5 1" />
<feFuncG type="discrete"
tableValues="0 .5 1" />
<feFuncB type="discrete"
tableValues="0 .5 1" />
<feFuncA type="discrete"
tableValues="0 .5 1" />
</feComponentTransfer>
</filter>
```

## EJEMPLO DE TINTE DE COLOR

```
<filter id="tinte-de-color">
<feComponentTransfer>
<feFuncR type="linear"
slope="-2"
intercept="0" />
<feFuncG type="linear"
slope="1"
intercept="0" />
<feFuncB type="linear"
slope="2"
intercept="0" />
<feFuncA type="linear"
slope="1"
intercept="0" />
</feComponentTransfer>
</filter>
```

## EJEMPLO DE AJUSTE DE CONTRASTE

```
<filter id="feComponentTransfer4">
<feComponentTransfer>
```

```

<feFuncR type="gamma"
  amplitude="2"
  exponent="5"
  offset="0" />
<feFuncG type="gamma"
  amplitude="2"
  exponent="3"
  offset="0" />
<feFuncB type="gamma"
  amplitude="2"
  exponent="1"
  offset="0" />
<feFuncA type="gamma"
  amplitude="1"
  exponent="1"
  offset="0" />
</filter>

```

## Composiciones: primitiva feComposite

La primitiva FECOMPOSITE permite la combinación de dos imágenes pixel a pixel en un mismo espacio usando el canal alfa de Porter-Duff. Este filtro puede ser una opción interesante, no obstante, hay que tener en cuenta que, en una cierta medida, la imagen resultante puede aumentar de tamaño, incluso si las dos imágenes se superponen.

La primitiva FECOMPOSITE puede ser utilizado para calcular medias e intersecciones. Además, permite tanto la superposición de imágenes como la mezcla relativa de sus valores de los píxeles. Al igual que FEMERGE se necesitan dos entradas IN y IN2. Por defecto, IN está establecido a SOURCEGRAPHIC.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>in</b>	Especifica el primer objeto que se desea combinar.
<b>in2</b>	Especifica el segundo objeto que se desea combinar.
<b>operator</b>	Especifica el tipo de composición que se va a utilizar. Puede tomar los valores de OVER, IN, OUT, ATOP, XOR, y ARITHMETIC. Todos ellos, excepto ARITHMETIC, son atributos de un único valor. Sin embargo, cuando se especifica ARITHMETIC, se deben establecer otros cuatro que son K1, K2, K3, K4.
<b>k1</b>	Especifica el factor múltiplo que representa a ambas imágenes.
<b>k2</b>	Especifica el efecto lineal de la primera imagen.
<b>k3</b>	Especifica el efecto lineal de la segunda imagen.
<b>k4</b>	Especifica el intercepto o ajuste de brillo.

<b>result</b>	Especifica el identificador para el resultado. Es útil cuando se desean combinar diferentes resultados de efectos o filtros.
---------------	--

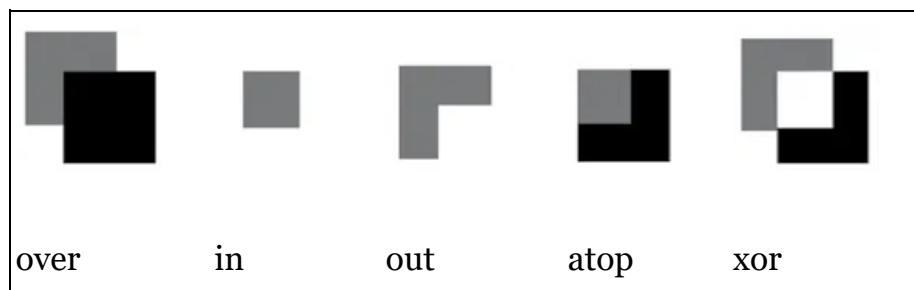
Por ejemplo, cuando el operador está en ARITHMETIC, si K1 es 0, K2 es 1, K3 es -1 y K4 es 1, lo que indica es que el gráfico en primer plano (IN) contribuye de manera positiva, que la imagen de fondo (IN2) contribuye negativamente y que el brillo ha sido aumentado.

En lo referente al atributo OPERATOR, sus posibles valores son:

Tipo	Descripción
<b>arithmetic</b>	Permite fusionar basándose en el porcentaje de mezcla de las imágenes como pueda ser el canal alfa o por efectos más complejos.
<b>atop</b>	Indica que se muestre ambas imágenes, pero, en lo referente a la imagen en primer plano, sólo debe mostrarse la intersección con la imagen que está en segundo plano.
<b>in</b>	Indica que, de la intersección entre ambas imágenes, se muestre únicamente la imagen en primer plano.
<b>Out</b>	Indica que, de lo que no es parte de la intersección entre ambas imágenes, se muestre únicamente la imagen en primer plano.
<b>over</b>	Indica que las imágenes deben mostrarse tal y como vengan definidas, es decir, la imagen en primer plano, sobre la que está en segundo plano.
<b>xor</b>	Indica que debe mostrarse lo que no es parte de la intersección entre ambas imágenes.

Cabe destacar que, el modo OVER, es equivalente al valor NORMAL de la primitiva FEBLEND. Además, coincide con el método de mezcla usada por FEMERGE y con la técnica ALFA COMPOSITING SIMPLE que se usa en SVG para toda la composición exterior de efectos de filtro.

A continuación, se muestran algunos posibles resultados:



### Ejemplo:

```
<svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%">
<defs>
<filter id="composite-atop" filterUnits="objectBoundingBox" >
<feFlood x="10"
y="10"
width="300"
height="100"
flood-color="#888"
```

```

flood-opacity="1"
result="figure1" />
<feFlood x="30"
y="30"
width="300"
height="100"
flood-color="#888"
flood-opacity="1"
result="figure2" />
<feComposite in="figure1" in2="figure2" operator="atop" />
</filter>
</defs>
<use style="filter: url(#composite-atop);"></use>
</svg>

```

## Matrices de convolución: primitiva feConvolveMatrix

La primitiva FECONVOLVEMATRIX permite definir un filtro de convolución de tipo matriz. El concepto de convolución hace referencia al tratamiento de una matriz que realiza un producto con otra que denominan KERNEL. En términos generales, es una combinación de los píxeles de la imagen con sus píxeles adyacentes, lo que genera nuevos píxeles y, por tanto, una nueva imagen.

Habitualmente, este tipo de filtros dan como resultado efectos de relieve, detección de bordes, estampado, biselado, enfocado y desenfocado, entre otros.

Para entender mejor este concepto pensemos en una imagen como si fuese una matriz bidimensional de píxeles que se desea tratar con otra matriz KERNEL que expresa el efecto deseado.

La matriz KERNEL, suele ser cuadrada, generalmente compuesta de N por N elementos, siendo N algún número impar y, frecuentemente, un valor de 3 o 5. Dicha matriz, se va desplazando sobre la matriz bidimensional de la imagen de tal forma que, el elemento central coincide con cada uno de los píxeles. En cada desplazamiento, el valor de cada píxel se multiplica por la matriz KERNEL o de convolución, dando un nuevo valor que, en conjunto con los demás, generan un efecto visual diferente.

Supongamos el siguiente ejemplo.

10	20	30	40	50			
11	21	31	41	51	01	0	
12	22	32	42	52	*	000	
13	23	33	43	53	000		

14	24	34	44	54			
----	----	----	----	----	--	--	--

Si nos fijamos en la ilustración anterior, el radio de acción de la matriz KERNEL (la de la derecha) para el píxel 32 es el conjunto de valores envueltos por el marco gris y, la forma de resolver el producto es multiplicar la fila de la matriz imagen por la columna de la matriz KERNEL. Es decir, aplicando la siguiente fórmula:

$$R_{FC} = A_{F1} * B_{1C} + A_{F2} * B_{2C} + A_{F3} * B_{3C}$$

En donde:

- $A$ ,  $B$  y  $R$  son las matrices imagen, kernel y resultado respectivamente.
- $F$  es la fila con la que operar.
- $C$  es la columna con la que operar.

Por tanto, el resultado debería ser:

0	21	0
0	22	0
0	23	0

Dicho esto, entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>in</b>	Especifica el objeto que se desea tratar.
<b>kernerMatrix</b>	Especifica la lista de coeficientes que constituyen la matriz del KERNEL. Los valores están separados por espacios en blanco. El número de entradas en la lista debe coincidir con los valores de los atributos ORDERY y ORDERX.
<b>divisor</b>	Especifica el valor por el que se debe dividir el resultado de operar con el atributo KERNELMATRIX. Si el divisor resulta ser la suma de todos los valores de la matriz, el valor por defecto, el efecto será una especie de anochecer con un tono de color general similar al del resultado. No obstante, si la suma es igual a cero, entonces el valor por defecto del atributo DIVISOR será el valor 1.
<b>order</b>	Especifica el número de células en cada dimensión de la matriz especificada por el atributo KERNELMATRIX. Los valores proporcionados deben ser enteros y mayores que cero. Los valores de estas dimensiones deben estar separados por espacios en blanco y en orden de ORDERX e ORDERY. Si no se especifica ORDERY, se establece al valor de ORDERX. Por defecto el valor del atributo ORDER es 3.
<b>bias</b>	Especifica un valor que se debe añadir una vez que se haya calculado la nueva matriz y operado

	con el divisor. Una aplicación típica de BIAS es cuando se desea tener la mitad del valor de gris estando la respuesta del filtro a cero. Por defecto, el valor de BIAS es cero.
<b>targetx</b>	Determina la posición en X de la matriz de convolución con respecto a un píxel objetivo dado en la imagen de entrada. La columna más a la izquierda de la matriz es la columna cero. El valor debe ser tal que $0 \leq \text{TARGETX} < \text{ORDERX}$ . Por defecto, la matriz de convolución se centra en X sobre cada píxel de la imagen de entrada, es decir, $\text{TARGETX} = \text{FLOOR}(\text{ORDERX} / 2)$ .
<b>targety</b>	Determina la posición en Y de la matriz de convolución con respecto a un píxel objetivo dado en la imagen de entrada. La fila superior de la matriz es la fila cero. El valor debe ser tal que: $0 \leq \text{TARGETY} < \text{ORDERY}$ . Por defecto, la matriz de convolución se centra en Y sobre cada píxel de la imagen de entrada, es decir, $\text{TARGETY} = \text{FLOOR}(\text{ORDERY} / 2)$ .
<b>edgemode</b>	Especifica la forma de extender la imagen de entrada, según sea necesario, con valores de color de modo que las operaciones matriciales se puedan aplicar cuando el núcleo está situado en el borde o cerca de él. Sus valores pueden ser DUPLICATE, WRAP o NONE. DUPLICATE indica que la imagen de entrada se extiende a lo largo de cada uno de sus bordes cuantos sea necesario mediante la duplicación de los valores de color en cada borde de la imagen de entrada. WRAP indica que la imagen de entrada se amplía mediante la adopción de los valores de color desde el borde opuesto de la imagen. NONE indica que la imagen de entrada se extiende con valores de píxeles de cero para todos los canales (R, G, B y A).
<b>preserveAlpha</b>	Especifica si se debe preservar o no la transparencia de la imagen. Sus posibles valores son TRUE y FALSE.
<b>result</b>	Especifica el identificador para el resultado. Es útil cuando se desean combinar diferentes resultados de efectos o filtros.

## EJEMPLO DE ESTILIZACIÓN DE BORDES

```
<filter x="0" y="0" width="1" height="1">
<feConvolveMatrix in="SourceGraphic" order="4" divisor="2"
kernelMatrix="1 -3.8 -1.5 6
0 0 -1 0
0 0 -1 0
0 0 0 0"
preserveAlpha="true" />
</filter>
```

## EJEMPLO DE BORDES AÑADIDOS

```
<filter x="0" y="0" width="1" height="1">
<feConvolveMatrix in="SourceGraphic" order="4" divisor="2"
kernelMatrix="3 0.3 0.2 0.5
```

```
1 0 0 0  
-1 0 0 -1  
-1 0.2 0 0"  
preserveAlpha="true" />  
</filter>
```

## EJEMPLO DE RELIEVE

```
<filter x="0" y="0" width="1" height="1">  
<feConvolveMatrix in="SourceGraphic" divisor="2"  
kernelMatrix="6 10 3  
10 -4 -15  
3 -5 -6"  
preserveAlpha="true" />  
</filter>
```

## EJEMPLO DE TRAZOS PINCEL

```
<filter x="0" y="0" width="1" height="1">  
<feConvolveMatrix in="SourceGraphic" order="4" divisor="2"  
kernelMatrix=" 2 1 1 1  
-1 0 0 0  
-1 0 0 0  
-1 0 0 0"  
preserveAlpha="true" />  
</filter>
```

## Iluminaciones: primitivas feEspecularLighting y feDiffuseLighting

Aunque SVG posee elementos para trabajar con la iluminación, para componer un haz de luz, o simplemente hacer un efecto de iluminación se necesitan de otros elementos secundarios que se verán más adelante.

### ILUMINACIÓN ESPECULAR: PRIMITIVA FEESPECULARLIGHTING

La primitiva FEESPECULARLIGHTING permite iluminar una imagen utilizando el canal alfa como un mapa de relieve haciendo uso de la variable de cálculo de iluminación explicado en el modelo de iluminación de Phong.

El modelo de iluminación d Phong se basa en una luz ambiente que viene reflejada desde todas partes, una luz difusa que proviene de manera directa desde la fuente, pero que rebota en todas partes y que, combinada con la ambiental define el color del objeto y, una luz specular que proviene directamente de la fuente, pero que rebota en una única dirección y define el brillo de la superficie.

Por tanto, la imagen resultante será una imagen RGBA basada en el color de la luz que dependerá del color de luz de posición, la luz y la geometría de la superficie del mapa de relieve de entrada.

La primitiva FEESPECULARLIGHTING supone que el espectador está en el infinito en todas direcciones y produce un mapa que contiene una parte de la reflexión especular como parte del cálculo de iluminación. Dicho mapa se debe combinar más tarde utilizando el método ARITHMETIC de la primitiva FECOMPOSITE. Dado que produce mapas independientes podemos crear múltiples fuentes de luz para conseguir el efecto que se desee.

A diferencia de FEDIFFUSELIGHTING, la primitiva FESPECULARLIGHTING produce una imagen no opaca. Esto es debido al hecho de que el resultado especular está destinado a ser añadido a la imagen. Además, el canal alfa resultante es el máximo de los componentes de color, de manera que cuando la luz especular es cero, no se añade cobertura adicional a la imagen ni tono blanquecino alguno que destaque sobre la opacidad.

Entre los atributos que admiten en su configuración, se deben destacar los siguientes:

<i>Atributo</i>	<i>Descripción</i>
<b>in</b>	Especifica el objeto dónde se aplicará el filtro.
<b>surfaceScale</b>	Especifica el grado o la intensidad de luz que se debe aplicar. Cuanto mayor es el valor, menos luz es aplicada.
<b>specularContant</b>	Especifica el valor de la constante K de Phong que controla la relación de reflexión. Cuanto mayor es el valor, más fuerte es el reflejo.
<b>specularExponent</b>	Especifica el valor del foco de la fuente de luz. Cuanto mayor es el valor, más intensa será la luz.
<b>result</b>	Especifica el identificador para el resultado. Es útil cuando se desean combinar diferentes resultados de efectos o filtros.

## ILUMINACIÓN DIFUSA: PRIMITIVA FEDIFFUSELIGHTING

La primitiva FEDIFFUSELIGHTING permite iluminar una imagen utilizando el canal alfa como un mapa de relieve.

El mapa de la luz producido por esta primitiva se puede combinar con una textura usando el operador ARITHMETIC de FEDIFFUSELIGHTING. De hecho, se pueden simular múltiples fuentes de luz mediante la adición de varios de estos mapas de iluminación antes de aplicarlo a una textura.

A diferencia de FESPECULARLIGHTING, la primitiva FEDIFFUSELIGHTING produce una imagen opaca. Esto es debido al hecho de que el resultado no está destinado a ser añadido a una imagen. Además, el canal alfa resultante es constante, de manera que añade una cobertura adicional a la imagen y, por este motivo, adquiere un tono blanquecino que destaca sobre la opacidad.

Cabe mencionar que, esta primitiva, en general produce resultados dependientes la resolución. Sin embargo, es posible hacer FEDIFFUSELIGHTING produzca resultados independientes de la resolución a través del atributo FILTERRES de la primitiva FILTER o a través del atributo KERNELUNITLENGTH de FEDIFFUSELIGHTING.

Entre los atributos que admiten en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>in</b>	Especifica el objeto dónde se aplicará el filtro.
<b>surfaceScale</b>	Especifica el grado o la intensidad de luz que se debe aplicar. Cuanto mayor es el valor, menos luz es aplicada.
<b>diffuseConstant</b>	Especifica el valor de la constante K de Phong que controla la relación de reflexión. Cuanto mayor es el valor, más grande y fuerte será la iluminación.
<b>result</b>	Especifica el identificador para el resultado. Es útil cuando se desean combinar diferentes resultados de efectos o filtros.

## PRIMITIVA FEPOINTLIGHT

La primitiva FEPOINTLIGHT permite realizar un efecto de luz como el que provoca el Sol. Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>x</b>	Especifica el valor de la coordenada X mínima para el área de representación del elemento.
<b>y</b>	Especifica el valor de la coordenada Y mínima para el área de representación del elemento.
<b>z</b>	Especifica el valor de la coordenada Z mínima para el área de representación del elemento.

### Ejemplo:

```

<filter>
  <feDiffuseLighting result="diffOut"
    in="SourceGraphic"
    diffuseConstant="1"
    lighting-color="white">
    <fePointLight x="20" y="20" z="10"></fePointLight>
  </feDiffuseLighting>
  <feComposite in="SourceGraphic"
    in2="diffOut"
    operator="arithmetic"
    k1="1"
    k2="1"
    k3="1"
    k4="0" />
</filter>
```

## PRIMITIVA FEDISTANTLIGHT

La primitiva FEDISTANTLIGHT permite realizar efectos de iluminación en la imagen para lograr que las imágenes se vuelvan más claras. Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>azimuth</b>	Especifica la dirección de la luz en el plano XY en el sentido de las agujas del reloj. Sus valores van establecidos en grados desde el eje X.
<b>elevation</b>	Especifica la dirección de la luz en el plano XY hacia el plano Z. Si el valor es positivo, la luz estará apuntando en la dirección de quién ve la imagen.

### Ejemplo:

```
<filter>
<feDiffuseLighting result="diffOut"
in="SourceGraphic"
diffuseConstant="10"
lighting-color="#4b1919">
<feDistantLight azimuth="45" elevation="0" />
</feDiffuseLighting>
<feComposite in="SourceGraphic"
in2="diffOut"
operator="arithmetic"
k1="2"
k2="2"
k3="0"
k4="0" />
</filter>
```

## PRIMITIVA FESPOTLIGHT

La primitiva FESPOTLIGHT permite realizar un efecto de iluminación tipo foco. Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>x</b>	Especifica el valor de la coordenada X mínima para el área de representación del elemento.
<b>y</b>	Especifica el valor de la coordenada Y mínima para el área de representación del elemento.
<b>z</b>	Especifica el valor de la coordenada Z mínima para el área de representación del elemento.
<b>pointsAtX</b>	Especifica la ubicación de X en el sistema de coordenadas definido por el atributo PRIMITIVEUNITS del elemento FILTER.
<b>pointsAtY</b>	Especifica la ubicación de Y en el sistema de coordenadas definido por el atributo PRIMITIVEUNITS del elemento FILTER.
<b>pointsAtZ</b>	Especifica la ubicación de Z en el sistema de coordenadas definido por el atributo

	PRIMITIVEUNITS del elemento FILTER.
<b>specularExponent</b>	Especifica la intensidad del foco o haz de luz. Cuanto mayor sea su valor, más iluminación obtendrá la imagen.
<b>limitingConeAngle</b>	Especifica la región con forma de cono dónde se proyecta la luz. Sus posibles valores son de 0 a 360 y se corresponde al ángulo en grados que debe establecerse entre el foco de luz donde apunta y el cono de luz puntual.

### Ejemplo 1:

```
<filter>
<feDiffuseLighting result="diffOut"
in="SourceGraphic"
diffuseConstant="5"
lighting-color="#ffffff">
<feSpotLight x="20"
y="420"
z="50"
pointsAtX="120"
pointsAtY="50"
pointsAtZ="0"
limitingConeAngle="6" />
</feDiffuseLighting>
<feComposite in="SourceGraphic"
in2="diffOut"
operator="arithmetic"
k1="1" k2="1" k3="0" k4="0" />
</filter>
```

### Ejemplo 2:

```
<filter>
<feSpecularLighting result="diffOut"
in="SourceGraphic"
specularConstant="10"
lighting-color="#4b1919">
<feDistantLight azimuth="90" elevation="99.5552" />
</feSpecularLighting>
<feComposite in="SourceGraphic"
in2="diffOut"
operator="arithmetic"
k1="2" k2="2" k3="0" k4="0" />
</filter>
```

### Ejemplo 3:

```
<filter>
<feSpecularLighting result="diffOut"
in="SourceGraphic"
specularExponent="25"
specularConstant="1.5"
```

```

lighting-color="white">
<fePointLight x="20" y="20" z="10">
<animate attributeName="z"
values="0; 20; 1"
dur="2s"
repeatCount="indefinite" />
</fePointLight>
</feSpecularLighting>
<feComposite in="SourceGraphic"
in2="diffOut"
operator="arithmetic"
k1="1" k2="1" k3="1" k4="0" />
</filter>

```

## Mapas de desplazamiento: primitiva feDisplacementMap

La primitiva FEDISPLACEMENT permite convertir los valores de todos los píxeles que conforman la imagen en distorsiones geométricas. El procedimiento para realizar estas distorsiones es utilizar un canal especificado (R, G, B, o A) del objeto destino como valores de desplazamiento para determinar la dirección y la distancia de cada pixel en la que se moverán a través de los ejes del sistema.

En otras palabras, este filtro utiliza los valores de píxeles de la imagen destino para desplazar espacialmente la imagen origen. Esta transformación se realiza mediante:

$$P'(x, y) <- P(x + scale * (XC(x, y) - .5), y + scale * (YC(x, y) - .5))$$

En donde:

- $P(X, Y)$  es la imagen origen, referenciada por IN.
- $P'(X, Y)$  es la imagen destino, referenciada por IN2.
- $XC(X, Y)$  y  $YC(X, Y)$  son los valores de los componentes definidos por los atributos XCHANNELSELECTOR e YCHANNELSELECTOR.

Los cálculos que utilizan los valores de los píxeles de IN2 deben ser valores no premultiplicados de color. Si la imagen de IN2 se compusiera de valores de color premultiplicados, lo que hará el sistema es convertirlos automáticamente en no premultiplicados antes de realizar esta operación.

Cabe mencionar que, este filtro puede tener un efecto no localizado arbitrario en la entrada que podría requerir un almacenamiento en búfer sustancial en la canalización de procesamiento y que puede determinarse por los valores de rango máximos de desplazamiento en X o Y.

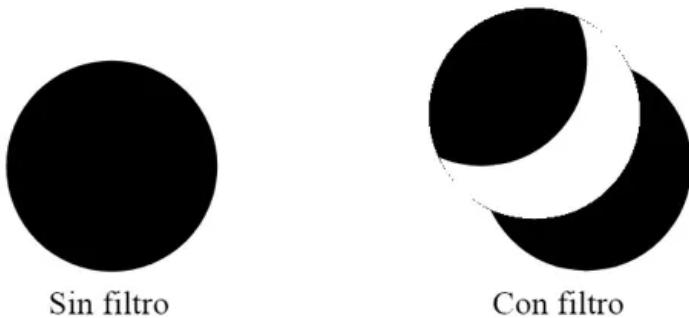
La primitiva FESPOTLIGHT permite realizar un efecto de iluminación tipo foco. Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>in</b>	Especifica el objeto origen.
<b>in2</b>	Especifica el objeto destino.
<b>scale</b>	Especifica el factor de desplazamiento que se debe utilizar. El valor debe ir expresado bajo el sistema de coordenadas que tenga establecido el atributo PRIMITIVEUNITS de la primitiva FILTER.
<b>xChannelSelector</b>	Indica el canal de destino a utilizar para desplazar los píxeles del objeto origen a lo largo del eje X. Sus valores pueden ser R, G, B o A.
<b>yChannelSelector</b>	Indica el canal de destino a utilizar para desplazar los píxeles del objeto origen a lo largo del eje Y. Sus valores pueden ser R, G, B o A.

### Ejemplo:

```
<svg viewBox="0 0 400 200" xmlns="http://www.w3.org/2000/svg">
<filter id="desplacement" x="-20%" y="-20%" width="150%" height="150%">
<feDisplacementMap scale="50"/>
</filter>
<circle cx="100" cy="100" r="50" />
<text x="70" y="170">Sin filtro</text>
<circle cx="300" cy="75" r="50" style="filter: url(#desplacement);"/>
<text x="280" y="170">Con filtro</text>
</svg>
```

Y el resultado debería ser algo como:



## Morfologías: primitiva feMorphology

La primitiva FEMORPHOLOGY permite realizar dilataciones o erosiones en las imágenes. Por entendernos, una dilatación se podría definir como un crecimiento de píxeles de la figura (lo que está en primer plano) sobre el fondo (lo que está en segundo plano) mientras que una erosión podría definirse, por tanto, como el proceso contrario.

Los filtros morfológicos se suelen aplicar para conseguir, entre otros efectos, granulometrías y texturas en todo tipo de imágenes, sin embargo, en donde más se suelen aplicar es en imágenes a escala de grises.

En este último tipo de imágenes, el resultado de aplicar un filtro de dilatación provoca que los detalles más claros resalten y que los detalles oscuros se reduzcan o eliminen dependiendo de sus valores. Por ende, ayuda a llenar los espacios ubicados dentro de la región de aplicación.

Por el contrario, el resultado de aplicar un filtro de erosión provoca que los detalles más claros se reduzcan o eliminen y que los detalles oscuros tomen más presencia. Por ende, ayuda a enfatizar los espacios ubicados dentro de la región de aplicación.

Matemáticamente hablando, la dilatación, consiste en una función máximo en donde el píxel de salida resulta ser el máximo valor de la componente individual que corresponde a los valores R, G, B, A del núcleo del rectángulo de la imagen de entrada.

En la erosión, consiste en una función mínimo en donde el píxel de salida resulta ser el mínimo valor de la componente individual que corresponde a los valores R, G, B, A del núcleo del rectángulo de la imagen de entrada.

El núcleo de dilatación o erosión se corresponde con una forma rectangular con un ancho y alto equivalente a 2 veces el valor de la componente por el valor del atributo del radio.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>operator</b>	Especifica la operación que se desea aplicar. DILATE o ERODE.
<b>radius</b>	Especifica el radio para la operación. Si se proporcionan 2 valores, el primero, representa al radio X y, el segundo, al radio Y. Si sólo se proporciona un valor, dicho valor representará tanto al radio X como al radio Y. Cabe destacar que los valores deben estar en el sistema de coordenadas establecido por el atributo PRIMITIVEUNITS de la primitiva FILTER y que no puede tomar valores negativos. Si el valor es cero el filtro no será aplicado. Por defecto, su valor es 0.

### Ejemplo:

```
<svg width="800" height="300" viewBox="0 0 400 300">
<filter id="erode">
<feMorphology operator="erode" radius="4"></feMorphology>
</filter>
<filter id="dilate">
<feMorphology operator="dilate" radius="4"></feMorphology>
</filter>
<image href="zen-circle.png"
width="90%" height="90%" x="-100%" y="5%" />
<image href="zen-circle.png"
width="90%" height="90%" x="0%" y="5%" filter="url(#dilate)" />
<image href="zen-circle.png"
width="90%" height="90%" x="100%" y="5%" filter="url(#erode)" />
</svg>
```

El resultado de este ejemplo nos muestra lo siguiente:



## Combinaciones: primitiva feMerge

La primitiva FEMERGE permite combinar filtros concurrentemente, en vez de secuencialmente y suele utilizarse como recurso para crear sombras con, o sin efectos de desenfoque.

Por tanto, en lugar de que cada primitiva de filtro sea aplicada a la salida que genera la primitiva de filtro anterior, se proporciona un método para almacenar los resultados de cada filtro y poder manipularlos como se deseé.

Evidentemente, para que este filtro produzca unos buenos resultados, las capas superiores deberían tener, al menos, una cierta transparencia dentro del en el área afectada por el filtro, con el objetivo de permitir que esas capas por debajo sean visibles.

Puede que alguno piense que este tipo de combinaciones puede realizarse a través del uso de primitivas FECOMPOSITE, pero es más conveniente realizar FEMERGE puesto que ofrece algo más de flexibilidad.

Entre las principales características de la primitiva FEMERGE es que permite establecer uno o varios elementos FEMERGENODE en los que especificar un atributo IN y hace que se renderice de forma integral en una única capa RGBA. No obstante, cabe aclarar que, si la imagen situada en primer plano es SOURCEGRAPHIC y FEMERGE es la última primitiva definida en la primitiva FILTER, el sistema renderizará las capas hasta ese momento y, en un último paso, renderizará el SOURCEGRAPHIC directamente sobre la parte superior.

Entre los atributos que admite en su configuración, sólo permite la declaración de elementos FEMERGENODE que especifican el origen de la entrada para la aplicación del filtro.

### Ejemplo:

```
<svg width="400" height="300" viewBox="0 0 400 300">
<defs>
<filter id="sombra">
<feGaussianBlur in="SourceAlpha"
stdDeviation="1"
result="blur" />
```

```

<feOffset in="blur"
dx="5"
dy="5"
result="offsetBlur" />
<feMerge>
<feMergeNode in="offsetBlur" />
<feMergeNode in="SourceGraphic" />
</feMerge>
</filter>
</defs>
<rect x="10"
y="10"
width="200"
height="150"
fill="#888"
filter="url(#sombra)" />
</svg>

```

El resultado de este ejemplo debería ser algo como:



## Turbulencias: primitiva feTurbulence

La primitiva FEURBULENCE permite crear efectos de deformación o ruido a través del algoritmo de ruido . Suele ser aplicado para generar texturas tipo mármol o con forma de nubes, aunque también es habitual verlo aplicado como patrón de ruido que rellenan los bordes de la figura con remolinos suavizados.

El ruido de Perlin es un tipo de ruido de gradiente que se basa en la creación de gradientes aleatorios o pseudoaleatorios a través de la interpolación de puntos y suele tener un efecto más fuerte cuanto mayor es la frecuencia.

Al igual que sucede FEFLOOD, la primitiva FETURBULENCE rellena la figura con nuevos contenidos. Estos contenidos vienen definidos fundamentalmente a través del atributo BASEFREQUENCY, aunque la modificación de los atributos NUMOCTAVES, TYPE, STITCHTILES y SEED puede ser determinante para la generación de las partículas, nubes, arena, ondas o deformaciones que componen la nueva imagen.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	Descripción
<b>basefrequency</b>	Especifica la frecuencia base para la función de ruido. Si se proporcionan dos valores, el primero, representa una frecuencia de base en el eje de las abscisas y, el segundo, en el eje de las ordenadas o eje Y. Por el contrario, si sólo se proporciona un único valor, se utilizará en ambos ejes. Por defecto, su valor es 0.
<b>numoctaves</b>	Especifica el “tamaño” de grano del patrón. En general, si este valor aumenta, su complejidad fractal también aumenta debido a que se vuelve más estrecho o fino. Por defecto, su valor es 1.
<b>seed</b>	Especifica el valor inicial para generar valores pseudoaleatorios. Por defecto, su valor es 0.
<b>stitchtitles</b>	Sus valores son NOSTITCH y STITCH. Si se establece a NOSTITCH, no se hace intento alguno para lograr una transición, lo que generará en ocasiones claras discontinuidades en los bordes. Si se establece a STITCH, se ajustarán automáticamente los valores DE BASEFREQUENCY-X y DE BASEFREQUENCY-Y de tal forma que el ancho y altura del filtro contenga un valor integral de anchura y altura del mosaico para la primera octava.
<b>type</b>	Especifica el tipo de función a aplicar. Sus posibles valores son FRACTALNOISE y TURBULENCE. Por defecto, su valor está establecido a FRACTALNOISE.

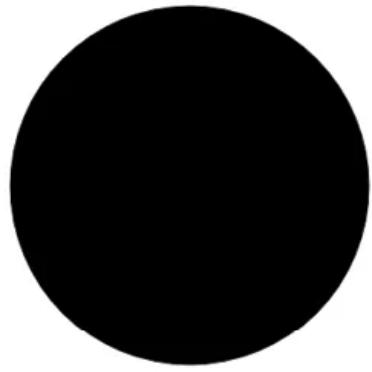
### Ejemplo:

```

<svg viewBox="0 0 400 200" xmlns="http://www.w3.org/2000/svg">
<filter id="distor">
<feTurbulence type="turbulence"
baseFrequency="0.3"
numOctaves="2"
result="turbulence" />
<feDisplacementMap in2="turbulence"
in="SourceGraphic"
scale="20" />
</filter>
<circle cx="100" cy="100" r="50" />
<text x="85" y="170">Sin filtro</text>
<circle cx="300" cy="100" r="50" style="filter: url(#distor);"/>
<text x="285" y="170">Con filtro</text>
</svg>

```

El resultado debería ser similar a lo siguiente:



Sin filtro



Con filtro

# 11

## USABILIDAD WEB

### DEFINICIÓN DE USABILIDAD WEB

La usabilidad es un término que no forma parte del diccionario de la Real Academia Española (RAE), aunque es bastante habitual en el ámbito de la informática y la tecnología.

Según la Wikipedia, el término Usabilidad se refiere a la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto. La usabilidad también puede referirse al estudio de los principios que hay tras la eficacia percibida de un objeto.

En interacción persona-ordenador, la usabilidad se refiere a la claridad y la elegancia con que se diseña la interacción con un programa de ordenador o un sitio web. El término también se usa a menudo en el contexto de productos como la electrónica de consumo o en áreas de comunicación, y en objetos que transmiten conocimiento (libro de recetas) o al diseño eficiente de objetos mecánicos (un martillo).

El grado de usabilidad de un sistema es, por su parte, una medida empírica y relativa de la usabilidad del mismo. Se mide a partir de pruebas empíricas y relativas.

- **Empírica** porque no se basa en opiniones o sensaciones, sino en pruebas de usabilidad realizadas en laboratorio u observadas mediante trabajo de campo.
- **Relativa** porque el resultado no es ni bueno ni malo, sino que depende de las metas planteadas (por lo menos el 80% de los usuarios de un determinado grupo o tipo definido deben poder instalar con éxito el producto X en N minutos sin más ayuda que la guía rápida) o de una comparación con otros sistemas similares.

El concepto de usabilidad se refiere a una aplicación (informática) de (software) o un aparato (hardware), aunque también puede aplicarse a cualquier sistema hecho con algún objetivo particular.

El modelo conceptual de la usabilidad, proveniente del diseño centrado en el usuario, no está completo sin la idea utilidad. En inglés, utilidad + usabilidad es lo que se conoce como USEFULNESS.

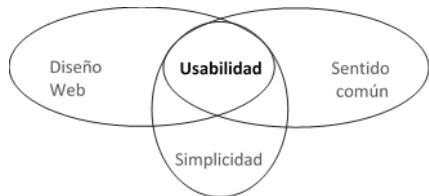
Jakob Nielsen definió la usabilidad como el atributo de calidad que mide lo fáciles que son de usar una interfaz o sistema.

Según la ISO/IEC 9126 y la ISO/IEC 9241 la usabilidad es una medida de efectividad, eficiencia y satisfacción referida a la capacidad que posee un software para ser comprendido, aprendido, utilizado y seductor para el usuario dentro de un contexto de uso específico.

En definitiva, como se puede ver existen muchas formas de explicar lo que es la Usabilidad, no obstante, no hay que olvidar un detalle importante. Cuando se habla de usabilidad, también se está hablando de experiencia de usuario (UX) porque es una de las partes que forma el conglomerado que dan sentido al término. Sin embargo, la usabilidad

puede que sea una de las más importantes, si no la más, porque es la que puede marcar la diferencia entre el éxito o el fracaso de un proyecto.

Por ello, la usabilidad también podría definirse como el equilibrio entre diseño, sentido común y simplicidad. Puede que alguno piense que la usabilidad son más cosas. Por ejemplo, algunos pensarán que sin seguridad tampoco hay Usabilidad, sin embargo, esa parte ya estaría contemplada en la sección de sentido común.



En resumen, una interfaz usable es aquella que es atractiva y en la que los usuarios pueden interactuar de la forma más sencilla, cómoda, evidente y segura posible. Un sistema o interfaz usable denota calidad, genera confianza y se posiciona positivamente sobre otras alternativas.

La usabilidad tiene como factores clave aumentar la eficacia, la eficiencia y la satisfacción. Estos factores clave se pueden desglosar en:

### **Usabilidad objetiva o inherente**

Aquella que puede ser evaluada por observación del usuario mientras realiza tareas de interacción u otros métodos tradicionales. La usabilidad objetiva o inherente mide la eficacia (facilidad con la que los usuarios encuentran lo que buscan) y la eficiencia (tiempo que tardan en encontrar lo que están buscando).

### **Usabilidad subjetiva o aparente**

Indica la usabilidad percibida o la satisfacción de uso y es difícil de entender y evaluar. La usabilidad subjetiva o aparente trata de medir la satisfacción que el usuario obtiene tras realizar una tarea por la interfaz o sistema.

### **Cómo se puede asegurar la usabilidad**

La usabilidad es una labor que está asociada a la calidad de los productos o sistemas, incluyendo las interfaces. La usabilidad no debe ser algo adicional o complementario a los diseños o desarrollos. Mejora la imagen que tienen los usuarios de la marca puesto que aumenta su satisfacción y, por lo general, esto se traduce en un incremento de los beneficios de la organización.

La usabilidad se asegura si el producto empatiza con los usuarios finales en todo momento, tiene un desarrollo iterativo e incremental y se realizan test de usabilidad con las métricas cuantitativas y cualitativas definidas desde que se inicia el proyecto.

## **IMPORTANCIA DEL DISEÑO WEB CENTRADO EN EL USUARIO**

Si el producto está destinado a cubrir unas necesidades concretas de los usuarios lo mejor es centrar todo el desarrollo en ellos. Este tipo de diseño requiere utilizar unos métodos, técnicas y procedimientos concretos y se caracteriza porque durante todo el desarrollo, desde que el prototipo se encuentra disponible, está supervisado por los usuarios finales. De esta forma se consigue un mayor grado de satisfacción.

Dicho esto, el Diseño Centrado en el Usuario (DCU) tiene varias posibles definiciones y, por tanto, nunca queda claro del todo cuál de ellas es la correcta o la más adecuada. No obstante, la expresión DCU suele identificarse más como

una metodología que engloba un conjunto de técnicas que pueden aplicarse durante todo el ciclo de vida del proyecto y que se caracteriza por:

- Se basa en una comprensión explícita de usuarios, tareas y entornos.
- Los usuarios son involucrados como parte activa y ayudan a dirigir y refinar durante todo el proceso de diseño y el desarrollo de manera iterativa.
- El equipo de diseño está formado por perfiles multidisciplinares.
- Está dirigido a toda la experiencia del usuario.

El número de técnicas utilizadas en el DCU puede estar comprendido entre un número pequeño y un número que puede llegar a ser decenas y que, habitualmente, suelen cubrir todas las necesidades o requisitos.

Cada técnica puede aplicarse en diferentes situaciones y momentos del diseño y desarrollo porque, generalmente, suelen estar definidas de forma específica, pero pueden ser utilizadas en contextos determinados.

Por tanto, el objetivo del Diseño Centrado en el Usuario es conseguir que los usuarios reales y potenciales queden satisfechos con la interfaz o sistema, gracias a una buena elección de las tecnologías, un buen cumplimiento de sus expectativas y un fácil manejo que les permita llevar a cabo sus tareas con éxito.

## DIFERENCIA ENTRE USABILIDAD Y ACCESIBILIDAD WEB

Aunque en muchos sentidos, no hay una diferencia clara en ambos conceptos, la accesibilidad es la cualidad de accesible, un adjetivo que se refiere a aquello que es de fácil acceso, trato o comprensión. El concepto se utiliza para nombrar al grado en el que todas las personas, más allá de sus capacidades físicas o técnicas, pueden utilizar un cierto objeto o acceder a un servicio.

Existen diversas ayudas técnicas para promover la accesibilidad y equiparar las posibilidades de todas las personas. Esto supone que un lugar que presenta buenas condiciones de accesibilidad puede recibir a toda clase de gente sin que exista un perjuicio o una dificultad para nadie.

Una de estas ayudas técnicas más comunes es lo que se denomina tecnología asistiva. **Una tecnología asistiva** (TA) es una herramienta utilizada para permitir que personas o usuarios con discapacidad puedan beneficiarse de las mismas ventajas que sus pares sin discapacidad.

Cuando se habla accesibilidad Web, en realidad, se hace referencia a una serie de normas de diseño que van a permitir a todo tipo de usuarios (con o sin discapacidad) percibir, entender, navegar e interactuar con una interfaz o sistema.

Un grupo de estas normas se conocen como **Pautas de Accesibilidad para Agentes de Usuario** (UAAG) y muestran cómo hacer que las herramientas formadas por navegadores, reproductores multimedia y tecnologías asistivas, entre otras, sean accesibles para personas con discapacidad.

Otro grupo de normas son las denominadas **Pautas de Accesibilidad para Herramientas de Autor** (ATAG) y tienen como objetivo definir la forma en la que las herramientas ayudan a los desarrolladores o diseñadores a producir un contenido que cumpla todas las Pautas de Accesibilidad al Contenido en la Web (WCAG).

Las ATAG están pensadas principalmente para desarrolladores entre las que se incluyen, editores de HTML y XML de WYSIWYG (What You See Is What You Get), procesadores de texto o paquetes de publicación, herramientas de conversión que transforman formatos de publicación a HTML, edición y producción de vídeo, paquetes de autor de

SMIL, gestores de contenido (CMS), herramientas de conversión instantánea o de publicación de sitios Web y herramientas de diseño (SASS, SVG o gráficos vectoriales, minificadores, etc.).

En resumen, no hay que confundir la accesibilidad con la Usabilidad porque la Usabilidad está más referida a la facilidad de uso, la accesibilidad está referida a que todas las personas puedan conseguir un resultado óptimo independientemente de la discapacidad que presenten y de los dispositivos que utilicen para acceder a los contenidos.

## **VENTAJAS E INCONVENIENTES DE CREAR SITIOS WEB USABLES**

Las principales ventajas o beneficios de la usabilidad son que:

- Es utilizable desde cualquier dispositivo y en cualquier resolución sin perder legibilidad, estructuración ni funcionalidad.
- La carga cognitiva es menor, por lo que se frustran menos y se sienten más cómodos.
- Reutiliza el conocimiento adquirido, por lo que el sistema o interfaz se vuelve mucho más intuitivo y el aprendizaje mucho más rápido y sencillo.
- Se basa en la simplicidad, por lo que los diseños son más minimalistas, claros, estéticos y limpios.
- Reduce los errores del sistema o interfaz, por lo que los usuarios obtienen una mayor satisfacción y aumentan la seguridad y las ganas de volver.
- Aumenta el rendimiento y disminuye los tiempos de desarrollo.
- Aumenta la tasa de conversión de visitantes.

Por el contrario, los principales inconvenientes son:

- El coste económico de las fases previas al diseño puede verse incrementado bastante dependiendo de los métodos o técnicas que se apliquen.
- El tiempo de desarrollo puede aumentar de manera considerable al principio a causa de la complejidad inherente del proyecto.
- La aplicación de algunos estándares, principios y leyes.
- Saber elegir la arquitectura y tecnologías adecuadas.
- La recuperación de datos fiables para llegar a las conclusiones correctas que permitan una implementación orientada y adecuada a los usuarios finales.

## **ANÁLISIS DE REQUERIMIENTOS DE USUARIO**

Existen varias técnicas para conocer a los usuarios, sin embargo, las técnicas más conocidas quizás sean el Diseño participativo o cooperativo, Design Thinking y esqueumorfismo.

### **Diseño conceptual**

El diseño conceptual es considerado una parte fundamental del diseño, entre otras razones, porque representa el proceso cognitivo a partir del cual se llega a una solución concreta.

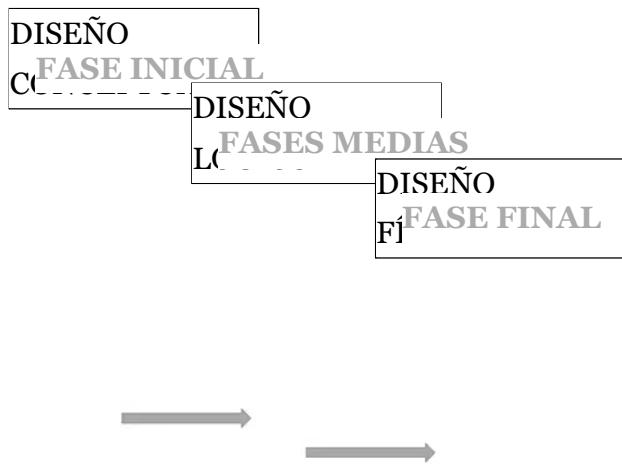
Si buscamos por Internet, encontraremos varias posibles definiciones, no obstante, todas vienen a decir un poco lo mismo. Es el proceso cognitivo por el que obtiene la solución a un problema concreto a partir de los requisitos, especificaciones y/o necesidades del usuario / cliente.

Suele ir precedido de algún proceso de investigación de mercado que justifica su planteamiento y, a menudo, se caracteriza porque es abstracto, con carencias de información y algo de incertidumbre.

También define cómo se va a organizar y cómo va a funcionar el sistema. Por ello, es importante que el diseño sea legible y comprensible, que no genere dudas, puesto que los usuarios no suelen ser expertos en informática y pueden no llegar a entender lo que en él se expone.

El diseño conceptual es un tipo de proceso que se ocupa de definir los atributos, propiedades y funciones del producto, en este caso, del sistema o interfaz, desde una abstracción a grandes rasgos, hasta un nivel de detalle suficiente para poder definir la solución.

Para que la solución sea algo más clara, antes de empezar a diseñar, se deben elegir los niveles de abstracción adecuados, evitar casos de uso generales, evitar situaciones homologas, soslayar expresiones que puedan dar lugar a ambigüedades o dudas y utilizar un estilo estandarizado.



Fases del Diseño de Software

## Diseño participativo o cooperativo (DPoC)

Diseño Participativo o Cooperativo (DPoC) es una metodología de trabajo que data de los años 60 y sostiene que el proceso de diseño debe involucrar que todos los actores que usen o puedan utilizar un producto.

Por tanto, si un producto está destinado a cubrir unas necesidades genéricas que contemplan varios tipos de usuarios o roles, lo mejor es utilizar un diseño participativo tratando de conseguir que todas las partes involucradas cooperen con el fin de asegurar la calidad final del producto.

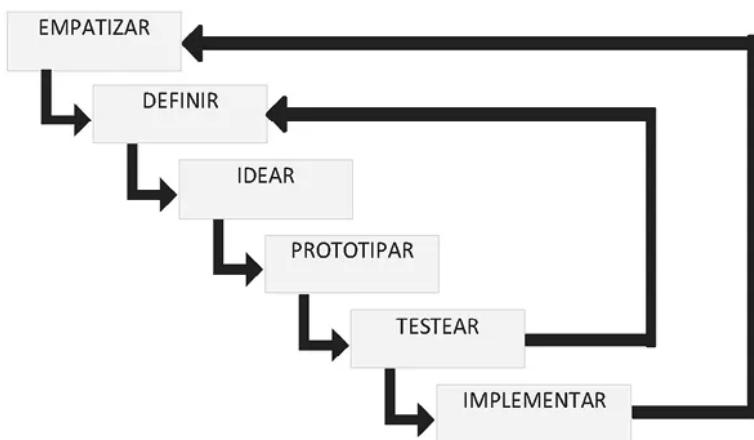
Que todas las partes involucradas sean tratadas de manera activa por igual actuando como consejeros técnicos, puede ser una muy buena técnica para conseguir propuestas más sólidas y realistas porque no se ciñe por los criterios de unos pocos expertos, sino por el criterio de todos.

El único inconveniente que tiene la metodología de DPoC es que se basa en acuerdos colectivos, requiere que los participantes o grupos aprendan unos de otros y que se transfieran información relevante. Por ello, puede no ser factible en muchas empresas u organizaciones.

Cabe destacar que, el Diseño Participativo o Cooperativo (DPoC) no sólo se utiliza en diseño web, también se utiliza en diseño urbano, arquitectura, diseño gráfico e, incluso, en otras áreas como la sustentabilidad o la medicina.

## Design Thinking

El Design Thinking es una metodología de DCU que se basa en innovar con la idea inicial de satisfacer a los usuarios y consta de 6 fases que están relacionadas en ambos sentidos.



Etapas de la metodología Design Thinking

- **Empatizar:** Esta fase trata de empatizar con los usuarios comprendiendo su entorno y sus necesidades.
- **Definir:** En esta etapa, una vez conocidas las necesidades de los usuarios, se especifican los problemas, limitaciones y desafíos que se tienen que resolver dentro de un cuadro innovador. Si existen dudas se debe volver a la fase de empatía para terminar de entender las necesidades y deseos de los usuarios.
- **Idear:** En esta etapa se basa en el brainstorming o tormenta de ideas para conseguir el mayor número de opciones posibles. Aquí no se deben tener prejuicios de valor ni menospreciar a ninguna persona ya que las soluciones pueden venir de cualquier integrante, sea usuario final, desarrollador senior, becario o cualquier otro. Podría darse el caso de que las ideas tuvieran que redefinirse y, entonces, se regresaría a la etapa anterior.
- **Prototipar:** En esta fase es dónde se construye un modelo a partir de las ideas extraídas en la fase de idealización. Este modelo debe ser una solución tangible que permita recolectar la máxima cantidad de información con el mínimo esfuerzo. Se debe tratar de que no consuma demasiados recursos ni tiempo para poder entrar cuanto antes en la etapa de testeо. Si se encuentra alguna necesidad no definida se puede a la etapa de empatía o a la como a la de idealización, según sea necesario.
- **Testar:** En esta etapa se recupera el feedback de los usuarios finales que están probando el producto. Una vez se tengan las conclusiones, se incorporarán al sistema para mejorar el producto.

## Esqueumorfismo

El esqueumorfismo es otra técnica de diseño centrada en el usuario que se está utilizando cada vez más. Su origen proviene del griego y viene a significar algo como “forma de herramienta”.



Ejemplo de esqueumorfismo. Autor: Klaus Göttling (Commons Wikipedia, 2017)

Cuando se utiliza en el área del diseño trata de conseguir que el diseño de una interfaz sea lo más realista o parecida posible al objeto original que imita.

Un ejemplo típico de esqueumorfismo es el micrófono dibujado que aparece en algunas aplicaciones móviles de grabación de voz o una casilla de verificación que parece un interruptor de la luz.

Sin embargo, este concepto no sólo se aplica a temas visuales, se puede aplicar a más ámbitos. También el sonido que hace una cámara analógica es un ejemplo de esqueumorfismo ya que los dispositivos móviles lo utilizan como indicador de que se ha realizado una fotografía.

## CREACIÓN DE PROTOTIPOS ORIENTADOS AL USUARIO



### El proceso de prototipado

El proceso de prototipado puede realizarse durante todo el ciclo de vida del proyecto, es decir, se puede utilizar para mostrar una idea o reunir requisitos en etapas tempranas, para validar las especificaciones en etapas medias, o para averiguar las diferentes soluciones a problemas de diseño específicos durante las etapas finales.

Este proceso es iterativo y va evolucionando desde su origen hasta adquirir una resultado fidedigno y bien parecido. Podríamos decir que, primeramente, trata de entender el propósito del sistema o interfaz a diseñar pensando cómo se va a mostrar para que los usuarios lo puedan entender cuando se vaya a presentar.

Una vez que se tiene una idea de cómo puede plantearse se toma nota de los requerimientos de los usuarios (a grandes rasgos) a través de una sesión de diseño de aplicación conjunta (JAD). Este proceso JAD resulta ser una especie de taller dónde los usuarios y el equipo de diseño se reúnen y discuten. Esto resulta es muy útil porque suele agilizar el desarrollo, mejora la calidad de las especificaciones e incrementa la participación de los usuarios.

Después de esta primera toma de requisitos, habitualmente, se construye un prototipo inicial baja fidelidad, que organiza estas especificaciones y se presenta y evalúa con los miembros involucrados.

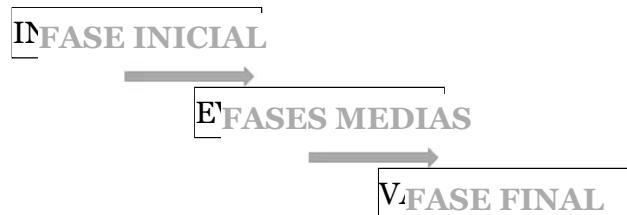
Finalmente, y de forma iterativa, se vuelve a la fase de requerimientos para conocer las nuevas necesidades y se sigue rediseñando y evaluando hasta que todos los miembros involucrados estén de acuerdo en que el prototipo es un diseño presenta una suficiente fidelidad.

## Tipos de prototipos

Si bien un prototipo tiene como finalidad validar las suposiciones iniciales y averiguar los posibles errores y oportunidades de mejora, la manera de conseguir esto puede hacer a través de varias formas.

### TIPOS DE PROTOTIPOS SEGÚN EL GRADO DE DETALLE

Según algunos expertos, los prototipos pueden ser de inspiración, evolución y validación.



#### Tipos de prototipos según el grado de detalle

Los **prototipos de inspiración** suelen darse durante el desarrollo de una idea, por lo que no suelen ser muy sólidos y fidedignos, sin embargo, sirven para formalizar las ideas iniciales dando paso a otras mucho más desarrolladas.

Los **prototipos de evolución** suelen aparecer después de haber diseñado un prototipo de inspiración, el cual está mucho más desarrollado y elaborado porque, en general, suelen tener más funcionalidades, si no todas.

Al igual que sucede con los prototipos de inspiración, la exposición de estos prototipos provoca que aparezcan fallas en el diseño, las cuales dan lugar a nuevas mejoras. No obstante, el número de fallas debería ser mucho menor que en su tipo predecesor.

Los **prototipos de validación** suelen aparecer ya en las etapas finales de los proyectos, puesto que su finalidad es evaluar la utilidad del producto y ultimar con los detalles.

Ahora bien, como llevar a cabo estos prototipos es otra historia. De hecho, existen multitud de formas de materializar o crear prototipos que permiten conseguir diferentes niveles de realismo. Sin embargo, en Usabilidad web lo más frecuente es recurrir a mockups y storyboards.

### TIPOS DE PROTOTIPO SEGÚN LA FASE DEL PROYECTO

Cuando se trata de desarrollo de aplicaciones y/o servicios web, siempre hay unos pasos que conviene seguir, aunque algunos expertos decidan obviar alguno de ellos por razones propiamente suyas.

En general, los hitos deberían ser los siguientes:

**S**I INSPIRACIÓN

**W**EVALUACIÓN

**M**EVALUACIÓN

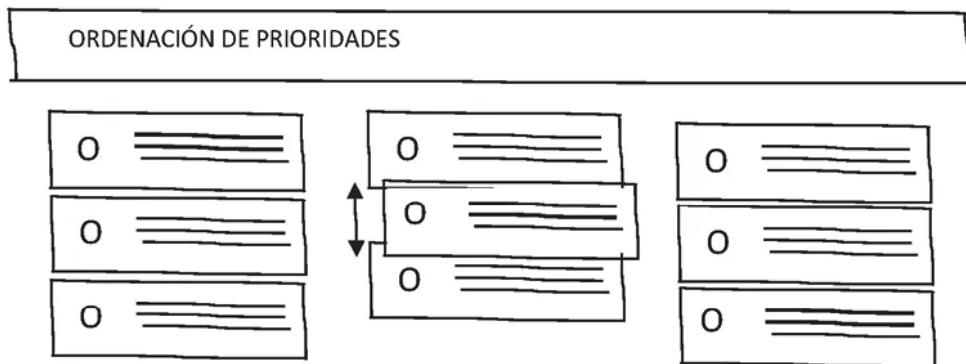
**P**l VALIDACIÓN

## Sketches y storyboards

La palabra sketch es un término inglés que significa “bosquejo” o “esquema” y viene a trasmitirnos la idea de que estamos en un contexto dónde las ideas son presentadas a modo de dibujo.

Un sketch es un tipo de prototipo basado en el esbozo de ideas y que suele estar creado más bien de una forma rudimentaria. En general, podríamos decir que son dibujos realizados a mano alzada con utensilios comunes como tijeras o lápices.

Normalmente, se realizan para representar una primera aproximación de la idea de un producto, funcionalidad o aspecto concreto de la interfaz o sistema.



### Ejemplo de sketch

Cuando se trata de sitios o aplicaciones web, un sketch debe responder a cuestiones como la situación del logo de la empresa u organización, del menú de navegación y de los servicios de redes sociales. Además, debería mostrar dónde están las áreas de contenido y cuál es el servicio que se va a presentar.

Sin embargo, este tipo de prototipos se utilizan en muchas otras ocasiones. Por ejemplo, es muy frecuente usarlo cuando los equipos de desarrollo o de diseño se reúnen para buscar una solución a un problema en particular.

Cuando ya se tienen varios sketches es posible crear un storyboard. De forma básica, un storyboard consiste en un conjunto de imágenes a modo de pantallas o bocetos que muestran, de manera ordenada, cómo los usuarios progresan utilizando el sistema o interfaz. Además, este tipo de prototipos suelen mostrar, de manera fidedigna, la secuencia acciones que se van a poder desarrollar dentro de un escenario concreto. Por ejemplo, un storyboard bien podría ser el proceso de logado en una aplicación.

Este tipo de prototipo suele darse más en etapas tempranas del proyecto y es de baja fidelidad, puesto que no se parece al producto final. Además, resulta ser una adaptación simple y económica que se suele crear con bastante fluidez y rapidez.

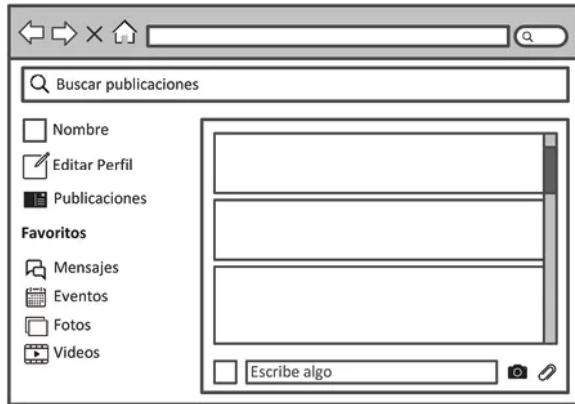
## Wireframes

La palabra wireframe es un término inglés que significa “alambre” y viene a trasmitirnos la idea de que estamos en un espacio bidimensional que se muestra cómo se asigna el espacio, cómo se prioriza el contenido, cuáles son las funcionalidades de las que dispone el interfaz o sistema y, en ocasiones, cómo se comporta.

Un wireframe es un tipo de prototipo más refinado que el sketch y, a menudo, se suelen utilizar para mostrar cómo se conectan la arquitectura de la información o las estructuras conceptuales con el diseño de interacción que se está proponiendo.

Aunque sean más visibles y elegantes no dejan de ser un prototipo que poco se parece al producto final, sin embargo, proporciona una visión mucho más fidedigna en cuanto a los contenidos, estructuras y niveles de legibilidad se refiere porque ayudan a establecer una relación visual entre las pantallas y las funcionalidades.

En lo referente a cómo crearlos, las recomendaciones son no utilizar colores, no utilizar imágenes e iconografía a no ser que ayuden al entendimiento y no causen una distracción que pueda hacer que los usuarios pierdan la noción de lo que se está trasmitiendo y utilizar un único tipo de fuente o tipografía.



Ejemplo de wireframe

Como se puede apreciar en la ilustración anterior, un wireframe permite a quién lo observa tomar una idea mucho más fiel de lo que será el producto final, no obstante, se debe tener cuidado de no cometer malas prácticas como uso de colores o elementos visuales que puedan confundir al evaluador.

Este tipo de prototipo suele darse más en etapas medias del proyecto y resulta ser de baja fidelidad, puesto que no se parece al producto final. Es una adaptación algo más compleja, aunque económica, y suele llevar algo más de tiempo realizarla.

## MockUp

La palabra mockup es un término inglés que significa “maqueta” y viene a trasmitirnos la idea de que estamos en un contexto dónde las ideas y funcionalidades son mostradas a modo de presentación visual mucho más fehaciente con el producto final.

Aunque algunos diseñadores sostienen que los mockups o maquetas no son necesarias, pueden llegar ser decisivos a la de validar el diseño antes de comenzar la etapa de desarrollo.

Un mockup, por tanto, es un tipo de prototipo que podría definirse como una representación visual más avanzada y realista de un proyecto. En este sentido, existen diseñadores que opinan que este tipo de diseños deben ser de alta fidelidad, mientras que otros piensan que debe ser más un producto transitorio.

En lo referente a cómo crearlos, aquí las recomendaciones son justo lo contrario que en los wireframes. Es decir, es recomendable utilizar los colores adecuados, utilizar imágenes y una iconografía acorde a la idea que se desea.

trasmitir y utilizar diferentes tipos de fuentes o tipografías para diferenciar secciones, títulos y textos. Además, suelen implementarse mediante HTML5, JavaScript y CSS con un planteamiento receptivo/fluido y con unos contenidos y lenguaje representativos.



### Ejemplo de wireframe

Entre los detalles importantes a la hora de crearlos, la recomendación es no ser demasiado innovador en el tema de animaciones y detalles, utilizar sistemas de rejilla responsive, utilizar los colores corporativos y adecuados para los contenidos y los mensajes de intercambio y, utilizar tipografías que sean legibles en cualquier tipo de dispositivo móvil o de escritorio, siempre y cuando respete la identidad del cliente.

## TIPOS DE PROTOTIPOS SEGÚN GRADO DE CALIDAD

Por último, sólo comentaremos que existe una clasificación según el grado de calidad o de fidelidad. A los **prototipos de baja fidelidad** se les denomina sí porque el aspecto que proporciona no es nada parecido al diseño final entregable. Un ejemplo bien podría ser un boceto o un wireframe.

Luego están los **prototipos de alta fidelidad** en dónde su aspecto suele ser mucho más parecido o fidedigno al diseño final que será entregado, aunque con matices. Un ejemplo bien podría ser un mockup o prototipo navegable.

## El proceso de desarrollo de un prototipo

Antes de empezar a realizar el proceso de prototipado puede ser una buena idea, realizar un estudio de mercado, definir el contexto cultural, político, económico, social y tecnológico y precisar el modelo de negocio con sus posibles problemas de contexto de uso. También podemos ayudarnos de diagramas o matrices DAFO (Debilidades, Amenazas, Fortalezas, Oportunidades) y de fluogramas de procesos y actividades.

## DEFINIR LA PERSONA

Según dijo Alla Cooper, la técnica de personaje y escenario es un mecanismo para definir los objetivos una aplicación y desarrollar sólo aquellas tareas o características que ayuden a cumplir dichos objetivos.

Cuando se habla de persona, en realidad, se está haciendo referencia a un perfil, usuario potencial o cliente. Por ello, en este contexto, las personas se deben tratar de definir lo más preciso posible, con un rol específico y con un mapa de empatía concreto que nos permita definirle de manera única.

Parece claro pues que, al concepto de persona, también referida como actor o personaje, se le puede catalogar como un arquetipo hipotético de un usuario real que representa un papel con unos objetivos concretos. Normalmente, los datos necesarios para crear este arquetipo se obtienen del análisis de requisitos o del estudio de datos estadísticos.

Por lo que se puede deducir que no hay que diseñar para todos los usuarios, hay que diseñar para cada arquetipo e intentando que cada funcionalidad nueva que se implemente no perjudique o afecte de manera negativa a los otros arquetipos.

<b>Plantilla de perfil</b>		
Nombre		Foto
Edad		
Sexo		
Adicional		
<b>Contexto de uso</b>		
Cuando		
Dónde		
Tipo dispositivo	Qué tipo de dispositivo utiliza. Móvil, escritorio, portátil,...	
<b>Misión</b>		
Objetivo		
Expectativas		
<b>Motivaciones</b>		
Urgencia		
Deseo		
<b>Actitud hacia la tecnología:</b>		
Miedoso, precavido, valeroso, agresivo, ...		

Ejemplo de plantilla de persona

## DEFINIR LOS OBJETIVOS

Es esta parte del proceso de desarrollo de un prototipo es habitual realizar una fase de brain storming, que no tiene por qué ser la primera, para captar todas las posibles ideas. No se deben hacer juicios de valor ni establecer límites de ninguna clase, puesto que todas las ideas son bienvenidas y todas ellas pueden llegar a aportar algo, desde las de un experto en la materia, hasta las de un usuario nuevo que acaba de llegar.

Cabe destacar que, los objetivos no son tareas. Los objetivos son la condición final tras realizar una o varias tareas. Pensemos que, las tareas pueden cambiar, pero el objetivo es inamovible e inevitable.

## DEFINIR LOS ESCENARIOS Y TAREAS

Con todas esas ideas lo siguiente que se debe hacer es definir los escenarios y tareas que trazan el recorrido, desde la primera interacción hasta que pueda realizar su objetivo.

Estos escenarios incorporarán las tareas al diseño y muestra cómo evoluciona el usuario hasta conseguir su objetivo. No se trata de describir de manera concisa cada tarea, se trata de describir el escenario en todo su esplendor.

En lo referente a los tipos de escenarios se suelen identificar tres tipos, los de uso diario, de uso necesario y de uso casual o casos límite.

Los **escenarios de uso diario** son los que se producen con mayor frecuencia y precisan buena solidez en la interacción.

Los **escenarios de uso necesario** son los que incluyen las acciones a realizar, aunque no suelen ser frecuentes. Estos escenarios tienen una cosa buena y es que, a los usuarios no les suele importar adaptarse a ellos si consiguen su objetivo.

Los **escenarios de uso casual o de casos límite** son los que abarcan anomalías o excepciones del tipo ¿y si ocurre esto o aquello? Se les suele nombrar casos límite porque se suelen utilizar en situaciones de emergencia o nada habituales. Este tipo de escenario no debe ser omitido, aunque sí se debe intentar priorizar.

Por ejemplo, un escenario podría ser una persona que quiere acceder a una plataforma de aprendizaje en la que está matriculada, pero resulta que ha insertado los datos de acceso de forma errónea. Entonces, el sistema le indica que los datos son incorrectos y que vuelva a intentarlo. Una vez que introduce sus datos de acceso de manera correcta, el sistema le enseña su página de inicio donde aparece información sensible, pero se va dejando el ordenador desbloqueado a vista de todos.

Plantilla de escenario	
Datos persona	
Nombre	Foto
Objetivo	
Escenario	

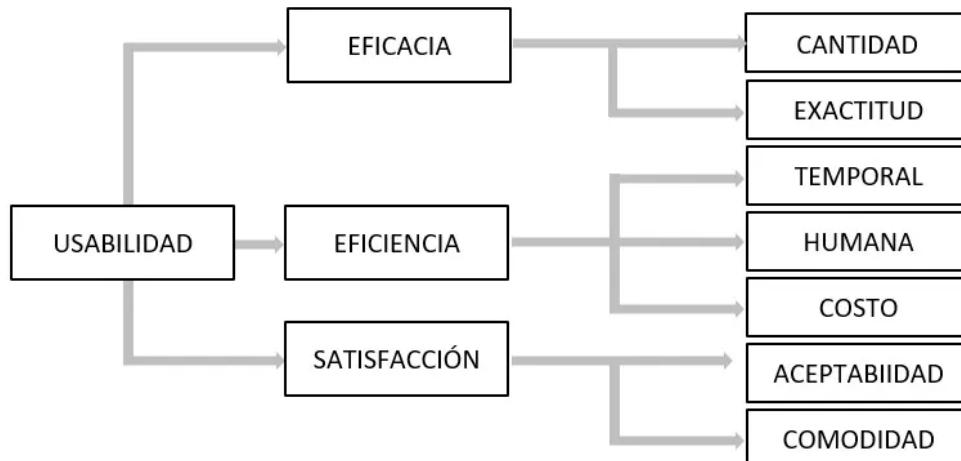
Ejemplo de plantilla de escenario

Como se puede apreciar, los escenarios suelen presentar una situación con uno actor que quiere cubrir uno o varios objetivos a través de la consecución de una o varias acciones que suelen ser producto su forma de pensar o de su comportamiento.

## DEFINIR LOS CONTENIDOS

El proceso de definición de contenidos puede requerir de mapas conceptuales, un inventario de recursos de información del producto y/o de la realización del árbol de contenidos.

Los mapas conceptuales son una herramienta gráfica que engloba y muestra los conceptos clave y enlaces de una forma semántica. Su principal ventaja es que ayudan, o pueden ayudar, a planificar y evaluar los proyectos.



Ejemplo mapa conceptual (Dimensiones de la usabilidad. Estandar ISO 9241-11)

El árbol de contenidos contiene todas las categorías y subcategorías de las tareas y representa las acciones que haría un usuario estructurándolas de una forma intuitiva y fácil de entender.

Si se desea crear un árbol de contenidos, lo más fácil es categorizar en orden descendente, es decir, se empieza por definir las categorías superiores, que suelen corresponderse con el menú de principal de navegación. Cada una de estas categorías principales se vuelven a dividir en otras de menor tamaño hasta que se dé con un elemento indivisible, pero siempre intentando que, este árbol, no supere los cuatro niveles de profundidad.

Nivel 1	Nivel 2	Nivel 3	Nivel 4
Inicio			
Acerca de Nosotros			
Servicios			
	Desarrollo		
	Diseño		
	Tecnologías		
	Asesoramiento		
Portfolio			
Blog			
	Desarrollo y Diseño Web		
		HTML5 y CSS3	
			Contenido 1
			Contenido 2
			Contenido 3
		JavaScript	
			Contenido 4

Ejemplo de árbol de contenidos

Como se puede ver, los contenidos se encuentran jerarquizados, es decir, primero reflejan la información más relevante o importante y se va disminuyendo en la relevancia hasta el elemento final.

De este árbol de contenidos se podrán extraer o definir las tareas. En lo referente a esto, la redacción de las tareas es importante que utilicen una terminología clara y concisa porque los test pueden fallar a causa de que los usuarios no entiendan las narraciones que se proponen.

Estas tareas deben dejar claras las áreas de negocio que se tienen o se desean asumir y las tareas que desea acometer el usuario. También deben comprender las secciones conflictivas, ya sea porque los usuarios presentan problemas para acceder a ellas o bien porque tienen una complejidad inherente muy elevada.

## **PAUTAS PARA LA CREACIÓN DE SITIOS WEB USABLES**

Jakob Nielsen, autoridad reconocida en el campo de la usabilidad propone, entre otros, diez principios básicos que una web debería cumplir:

### **Visibilidad del estado del sistema**

El sistema debe informar a los usuarios del estado del sistema, dando una retroalimentación apropiada en un tiempo razonable.

Para cumplir este principio se debe informar al usuario a través de barras de progreso o mensajes informativos durante la ejecución de procesos no inmediatos (envío de correos, carga de archivos o imágenes, conexiones remotas, ...) y al finalizar los mismos. Asimismo, se debe proporcionar migas de pan y diferenciar las diferentes partes de los procesos, por ejemplo, el “Proceso de compra”.

## **Relación entre el sistema y los usuarios**

El sistema debe utilizar un lenguaje al que los usuarios estén habituados, con una jerga o terminología que les sea conocida y legible, en lugar de términos o mensajes genéricos que se utilizan en el sistema, para que al usuario se sienta cómodo al utilizar el sistema.

Para cumplir este principio la información debe aparecer en un orden lógico y natural y utilizar iconos o imágenes claras que no dejen lugar a dudas (la papelera como símbolo de eliminar).

## **Control y libertad para el usuario**

En casos en los que los usuarios puedan seleccionar una opción del sistema por error, se debe contar con las opciones de deshacer y rehacer para proveer al usuario de una salida fácil que no le provoque frustración.

Para cumplir este principio se deben proporcionar botones o acciones que le permitan al usuario volver al estado anterior y, si la situación lo requiere, disponer de una “salida de emergencia”. Los botones de volver o cancelar para regresar a la página anterior, “rehacer” o “deshacer” de los editores de textos, el botón de eliminar del “carrito de la compra”, editar la información de nuestro perfil o habilitar la tecla “escape” para volver atrás son buena prueba de ello.

## **Consistencia y estándares**

El usuario debe seguir las normas y convenios de la plataforma sobre la que está implementando el sistema, para que no se generen dudas y tenga que preguntar el significado de las palabras, situaciones o acciones del sistema.

Para cumplir este principio se debe asegurar que los nombres de las acciones se corresponden con lo que hacen, no existen enlaces rotos, que los títulos son representativos del contenido y que cada enlace o botón lleva a un único destino. Si, además, la web debe ser accesible se deben cumplir los niveles de adecuación adecuados.

## **Prevención de errores**

Es importante que el usuario entienda lo que sucede en el sistema para poder prevenir las situaciones que puedan provocar frustración.

Para cumplir este principio se deben proporcionar métodos para salvar los errores más frecuentes, utilizar mensajes claros y eliminar las acciones predispuestas al error. Utilizar buscadores predictivos o autocompletos para que los usuarios no tengan que escribir toda la palabra, proporcionar una confirmación de contraseña con doble campo de entrada en los formularios, impedir que se puedan introducir letras en un campo dónde sólo se admiten números (como un campo edad o campo fecha) o validar los campos de un formulario en tiempo real son un claro ejemplo de ello.

## **Reconocer antes que recordar**

El sistema debe minimizar la información que el usuario debe recordar mostrándosela a través de objetos, acciones u opciones. El usuario puede que no recuerde toda la información que recibió con anterioridad ya sea un momento

cercano o no.

Para cumplir este principio se deben proporcionar instrucciones para el uso del sistema en todo momento y estar en un lugar visible. Es muy efectivo que una acción sea referida a través de colores, tamaños y/o posiciones distintas.

Ejemplos de este principio son que los botones de “cancelar” sean claramente diferenciables mediante una forma o color, que el menú de navegación esté arriba a la derecha, utilizar vistas previas en la selección de fuentes en los editores de texto, diferenciar bien los títulos de lo que es contenido, mostrar la lista de artículos y su número dentro del “carrito de la compra”, mostrar los artículos que ha visitado con anterioridad.

## **Flexibilidad y eficiencia de uso**

Se deben proporcionar métodos abreviados o atajos de teclado para que tanto los usuarios novatos como los expertos aprovechen la capacidad del sistema y se puedan adaptar sin importar el nivel de conocimiento de la plataforma.

Para cumplir este principio se debe proporcionar una forma para personalizar las acciones más frecuentes en el sistema. Utilizar enlaces personalizados por nombre de usuario en el menú de navegación, proporcionar un buscador avanzado con valores parametrizables además del buscador normal o utilizar atajos de teclado programables o personalizables y utilizar los ya estandarizados como F5 para recargar la página son ejemplo de ello.

## **Diseño estético y minimalista**

La interfaz no debe contener información que no sea relevante o innecesaria. Cada información extra que se introduzca competirá con la información relevante y disminuirá su visibilidad.

Para cumplir este principio se debe evitar los diseños recargados, que carguen rápido, con contenidos bien estructurados, colores equilibrados bien seleccionados y con, únicamente, las acciones necesarias.

En Internet existen multitud de ejemplos con diseño estéticamente simple y minimalista, sin embargo, hay que aclarar que algo minimalista puede no ser estético o viceversa. Por ejemplo, si se desea combinar los típicos colores asociados a algunas acciones como el rojo para eliminar o el verde para enviar o guardar con los colores corporativos de una empresa u organización, debe hacerse con cuidado porque algunas combinaciones pueden resultar cargantes o inesperadas y eso puede también frustrar a los usuarios.

## **Reconocimiento, diagnóstico y recuperación de errores**

Los mensajes de error deben expresarse en un lenguaje claro, indicar exactamente el problema y ser constructivos.

Para cumplir este principio se deben evitar mensajes de error como “Error 34-x1” o “Error 404” que no indican nada al usuario de cómo recuperarse cuando se produce dicho error. Los mensajes deben proporcionen alternativas para que el usuario pueda continuar realizando la tarea. Por ejemplo, un mensaje de “página no encontrada” debe tener una descripción más larga y que le permita al usuario saber qué hacer, tanto si el error es por una inexistencia de coincidencia en una búsqueda o por un error del sistema.

## **Ayuda y documentación**

Aunque es mejor que el sitio web o aplicación pueda ser usado sin ayuda, puede ser necesario proveer cierto tipo de ayuda.

Si este requerimiento se vuelve necesario, la ayuda debe ser fácil de localizar, se deben especificar los pasos necesarios y no ser muy extensa.

Cuando se trata de aplicaciones móviles, hoy en día, es frecuente utilizar un minitutorial o tour en donde, de manera sencilla, se exponen las funcionalidades principales que evitan tener que leer documentos extensos de ayuda y que pueden resultar aburridos o agotadores.

## PRÁCTICA 12: VERIFICANDO Y MEJORANDO LA USABILIDAD WEB DE UNIVERSSES

### Código del ejemplo

<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-12>



### Objetivo de la práctica

Verificar y solucionar los posibles problemas de rendimiento, semántica web, SEO, usabilidad. Aunque esta práctica también se contempla algo de accesibilidad web, esta parte se verá más profundamente en el próximo capítulo.

### Resolución

Si analizamos las pautas para la creación de sitios web anteriormente comentadas, veremos que, en general, se cumplen todas ellas en la página de inicio de nuestro sitio web de UniversES. El diseño es estético y minimalista, los botones llevan a un único destino, el lenguaje es adecuado para los usuarios reales y potenciales, etcétera.

Puede que alguno haya caído en la cuenta de que para hacer el diseño de UniversES no hemos recurrido a prototipos de sketches o Wireframes. La razón de habernos saltado estos prototipos es porque nosotros ya teníamos claro lo que queríamos hacer y eso nos ha permitido pasar directamente a la maquetación.

No obstante, crear pensando en la Usabilidad es un proceso que no siempre puede darse. Recordemos que, para que un diseño sea usable, se tiene que diseñar pensando en la usabilidad desde el inicio del proyecto, no al final. De hecho, una vez que ya tengamos la web en Producción, puede que no mejoremos nada, y digo puede, la usabilidad de las páginas.

Hagamos la prueba con LA PÁGINA DE INICIO de sitio web UniversES. Para ello, lo único que debemos hacer es abrir la consola del navegador (pulsando F12 o CTRL + SHIFT + J) y entre las pestañas que nos ofrece, seleccionar la opción de LIGHTHOUSE.

Dentro de esta pestaña denominada LIGHTHOUSE, seleccionaremos las opciones de Rendimiento, Aplicación Web progresiva, Mejores prácticas, Accesibilidad y SEO y, en tipo de dispositivo, seleccionaremos la opción de Mobile.

### NOTA

Por si alguno se pregunta por qué hemos seleccionado la opción de Mobile, en vez de la opción Desktop, es porque la opción Mobile es mucho más restrictiva y, además, el diseño que hemos hecho hasta, era todo pensado para escritorio.

Tras pulsar en el botón de “Generate Report”, se nos debería mostrar un resumen con los principales indicadores de la evaluación. Si todo ha ido como debe, el informe de auditoría de Lighthouse mostrar algo como lo siguiente:



Como se puede apreciar en la imagen, dependiendo del rango de valores en el que nos encontramos, estaremos cumpliendo o incumpliendo los requisitos mínimos establecidos por Google. En el caso expuesto, lo que llama la atención son los dos primeros puntos de la izquierda, aunque todos son importantes.

Si observamos las anotaciones del resultado sobre el rendimiento, lo que vemos es que, entre otras cosas, para la página analizada, se están estilos innecesarios, que las reglas de estilo no están minimizadas y que las imágenes tardan bastante en cargarse.

## MEJORANDO EL RENDIMIENTO

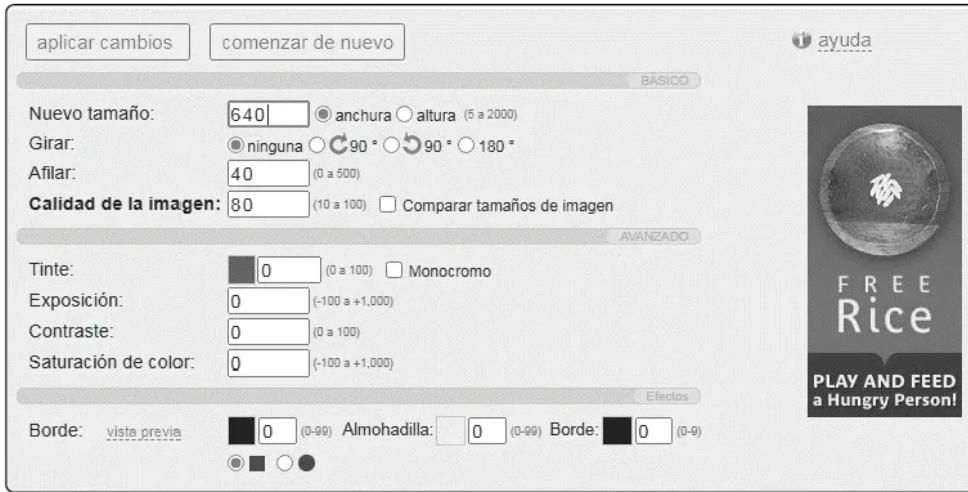
### Mejorando el tiempo de carga para las imágenes

Para mejorar el tiempo de carga de las imágenes, lo que se puede hacer es indicar varias fuentes y resoluciones. Esto, como se vio en capítulos anteriores, es posible hacerlo mediante el atributo SRCSET de HTML, o a través del elemento PICTURE ya que, ambos métodos, resultan ser compatibles con la inmensa mayoría de los navegadores actuales incluyendo Microsoft Edge versión 13.

Para ello, si bien se ha de modificar el HTML de nuestra página, primero adaptaremos las imágenes a las resoluciones adecuadas. En nuestro caso, contemplaremos las resoluciones de 340 píxeles de ancho para dispositivos móviles, 640 píxeles de ancho para dispositivos de resolución media y Full HD y, 1280 píxeles de ancho para dispositivos de alta definición como puedan ser pantallas con resoluciones de 2K o 4K.

Cabe mencionar que, la adaptación de todas estas imágenes debería ser más un proceso mecánico que laborioso. Así, por ejemplo, esta tarea es posible realizarla a través de PhotoShop, mediante la opción de Imagen / Tamaño de Imagen y, dentro del diálogo emergente que aparece, cambiando las dimensiones de la imagen. No obstante, también se puede recurrir a herramientas online como Web Resizer.

Si recurrimos esta herramienta online, sólo debemos ir a la dirección deseada (en este caso <http://webresizer.com/resizer/>) y seleccionar el archivo desde nuestro equipo. Una vez seleccionado, pulsamos en subir la imagen y, en la opción de “nuevo tamaño”, establecemos el valor deseado y pulsamos en “aplicar cambios”.



Seguidamente, guardaremos la imagen resultante situándonos encima de la imagen de la izquierda, pulsando en el botón derecho del ratón y seleccionando “Guardar imagen como ...”.

#### NOTA

Este proceso hay que hacerlo para los tamaños de 640 y 340 píxeles, puesto que la resolución de 1280 píxeles ya la tenemos descargada. También, es posible ir a la URL de la imagen original (Pixabay) y descargar las versiones que coincidan con el ancho requerido para nuestro ejemplo.

Cuando hayamos realizado este proceso repetitivo de crear las diferentes versiones para todas nuestras imágenes, modificaremos el HTML de nuestra página. Para llevar a cabo esta tarea, lo que haremos es sustituir la definición de cada una de ellas por una declaración de PICTURE con dos elementos SOURCE y un IMG.

#### Utilizar imágenes en formatos de última generación

Si bien el redimensionado y adaptación de las imágenes puede bajar el tiempo de latencia de forma considerable en una página, la compresión puede disminuirlo aún más.

Mientras que los formatos de JPEG y PNG presentan una compresión más que aceptable, prácticamente sin pérdidas importantes, los formatos de última generación como JPEG 2000, JPEG XR y WebP pueden aumentar esta compresión hasta un 30 por ciento más y con unas características de calidad superior.

No obstante, el formato que más parece extenderse es WebP, el cual es compatible con todos los navegadores actuales, si exceptuamos a Safari. Además, proporciona una mejor compresión con o sin pérdida de calidad a elección del usuario.

Para conseguir las imágenes en formato WebP podemos utilizar el plugin de PhotoShop o, en su defecto, utilizar una herramienta online. En esta ocasión vamos a recurrir <https://www.online-convert.com/es>. Una vez que nos registramos de forma gratuita, accedemos al conversor de imágenes y convertimos todas nuestras imágenes JPEG al formato deseado, en este caso, WebP.

Cuando hayamos realizado el proceso de conversión, lo siguiente será modificar, otra vez, el HTML de nuestra página para que coja las nuevas imágenes. Podríamos haber hecho este paso directamente, pero la idea es ir descubriendo, poco a poco, todos los detalles que hacen que una web suba su usabilidad web.

Si no fijamos en la definición de dichos elementos PICTURE, puede que caigamos en que, esta solución, puede resultar incompatible algunos navegadores. Por esta razón, aunque se establecen las nuevas imágenes en formato

WebP, se mantiene el elemento IMG con la imagen en formato antiguo, precisamente, por si hay algún usuario que no disponga de un navegador con soporte de WebP.

Ya hemos modificado las imágenes de nuestra página de inicio. Ahora, nos falta actualizar las imágenes involucradas en la hoja de estilos.

Esta tarea es algo más complicada porque, para que sea compatible con todos los navegadores debemos crear una función JavaScript que nos permita conocer si el sistema es o no compatible con WebP. Por tanto, antes de hacer los cambios en nuestro archivo STYLES.CSS, lo que haremos es abrir el archivo SCRIPTS.JS y añadir una función de detección al principio del archivo que llamaremos ISSUPPORTSWEBP.

Básicamente, esta función utiliza la funcionalidad de CANVAS para comprobar si puede o no renderizar en memoria una imagen en formato WebP. De no ser así, añadirá la clase WEBP-NO-SUPPORTED a la etiqueta BODY, la cual utilizaremos después.

Ahora, tras haber modificado el JavaScript de la página, modificaremos el archivo STYLES.CSS para que contemple las nuevas reglas del banner, con y sin soporte a WebP.

Este proceso, esencialmente, es ir replicando algunas de las reglas y propiedades para que, en función de si el elemento BODY posee la clase WEBP-NO-SUPPORTED, se carguen las imágenes en uno u otro formato.

## Minificar archivos CSS

Ahora, aunque tengamos las imágenes optimizadas, debemos minificar el tamaño de las hojas de estilos, es decir, reducirlas hasta que únicamente contengan las reglas que usa la página.

El proceso de reducción de CSS no es algo complicado, aunque sí algo tedioso. Para entender mejor este proceso veamos un ejemplo:

```
/* EJEMPLO DE CÓDIGO SIN MINIFICAR */
html, body {
font-size: 14px;
height: 100%;
margin: 0;
padding: 0;
}
body * {
}
/* EJEMPLO DE CÓDIGO MINIFICADO */
html, body{font-size:14px; height:100%; margin:0; padding:0}
```

Como se puede observar en el ejemplo anterior, el proceso de minificación es básicamente, la eliminación de saltos de línea, espacios en blanco y signos de puntuación innecesarios y la eliminación de reglas de estilo con definiciones vacías.

Podríamos haber realizado la declaración de reglas de estilo de manera reducida desde un principio, sin embargo, no es una buena idea puesto que los archivos CSS minificados se vuelven inmanejables si se tienen que editar o corregir.

De hecho, lo habitual es que el desarrollador edite las reglas de estilo sin minificar, con cada propiedad CSS en una línea, y se recurra a alguna herramienta que se integre con el sistema de forma que, la minificación se realice de forma automática. A continuación, se muestran algunas de las más populares:

Nombre	Descripción
<b>Nano CSS</b>	Para automatizar desde terminal o desde PostCSS.

<b>Clean CSS</b>	Basado en NodeJS para automatizar desde terminal.
<b>CSSO</b>	Optimizador de CSS que limpia, comprime y reestructura.
<b>YUI Compressor</b>	Compresor de CSS de Yahoo.

No obstante, si esta posibilidad no pudiese darse por alguna razón, en su defecto, la alternativa puede ser utilizar una herramienta online. En este contexto, una posible opción puede ser utilizar la herramienta online CSS Minifier, accesible desde la URL <https://cssminifier.com/>.

The screenshot shows a web-based CSS minification tool. On the left, under 'Input CSS', there is a large block of unminified CSS code. On the right, under 'Minified Output', the same code is shown in a much more compact, minified format. At the bottom of the interface, there are four buttons: 'Minify' (highlighted in dark grey), 'Download as File', 'RAW', and 'Clear'.

```

position: absolute;
top: 10px;
width: 8px;
}

.contact_maximized fieldset form {
height: calc(100% - 28px);
overflow: auto;
}

@media screen and (max-width: 768px) {
.contactfieldset h1 { font-size: 1.8rem; }
.contactfieldset * { font-size: 1.2rem; }
.contactfieldset form input + div > span { width: 100%; }
}

@import url(https://fonts.googleapis.com/css?family=Roboto&display=swap);@import url(https://fonts.googleapis.com/css?family=Megrim);@import url(https://fonts.googleapis.com/css?family=Tuareg&display=swap);@import url(https://fonts.googleapis.com/css?family=Inconsolata:wght@200,300,400,500,700,800,900&display=swap);/*box-sizing:border-box*/body{font-size:14px; height:100%; margin:0; padding:0;}body{font-family:Roboto, sans-serif; display:block; body { box-sizing:border-box; outline:0; none; }h1 { font-size:1.8rem; font-variant:small-caps; margin:1.2rem 0.5px; padding:0; }h2 { font-size:1.8px; color:#fff; display:block; height:100px; object-fit:cover; position:relative; margin-bottom:10px; width:100%; img{error:attr(img:not([src]))::after{img{src=""}};after{background-color:#000; border:0.5px solid #fff; border-radius:50%; height:100%; left:50%; margin-left:-50px; position: absolute; top:50%; width:100%; display: block; height:100%; left:0; overflow:hidden; padding:5px; position: absolute; top:0; width:100%; text-align:center; width:100%; z-index:2; loader}}
```

Una vez hayamos minificado el CSS del archivo STYLES.CSS, crearemos un nuevo archivo que se llame STYLES.MIN.CSS y, en él, pegaremos el resultado del CSS minificado. Después, en nuestro HTML, sustituiremos la fuente de reglas de CSS al nuevo archivo minificado.

```
<link href=".//css/styles.min.css" rel="stylesheet" type="text/css" />
```

## Eliminar los estilos no utilizados

La aplicación de esta premisa no suele darse puesto que, para que pueda cumplirse, cada página debería tener una hoja de estilos propia que incluyese únicamente las reglas de estilo que usa.

Existe una posibilidad, y es cargar las reglas de manera dinámica a demanda, es decir, crear un sistema autónomo que sea capaz de identificar las reglas de estilo que requiere la página que se va a ejecutar y comprimir las en un único archivo CSS, sin embargo, esto no suele darse por la dificultad que puede conllevar.

Al final, lo normal es que todas las reglas CSS de los sitios web se encuentren unificadas en un único archivo, el cual se carga de manera predeterminada por todas las páginas. Esto supone una pequeña pérdida en lo que a tiempos de latencia se refiere, pero no complica tanto el desarrollo.

Por esta razón, en lo que respecta a esta parte, no realizaremos ninguna modificación en la hoja de reglas de estilo.

## Estableciendo la preconexión a orígenes requeridos

La conexión con orígenes requeridos es una característica que vemos casi a diario. Es muy frecuente que las páginas se conecten con algún CDN (Red de Distribución de Contenidos) como pueda ser Google Fonts, o Cloud Flare.

Dado que en nuestra página utilizamos estas fuentes de terceros, siempre será una buena idea añadir una de las cláusulas de exploración previa para aumentar el rendimiento de la página. Si recordamos, las cláusulas PRECONNECT o DNS-PREFETCH permitían conectarse con algún otro origen agilizando la carga, sin embargo, si no se realizan las peticiones en los siguientes 10 segundos, el agente de usuario podría cerrar la conexión y se habrán desperdiciado recursos de CPU y conexión temprana.

Dicho esto, para conseguir que nuestra página disminuya el tiempo de carga, estableceremos la cláusula PRECONNECT a todas nuestras fuentes externas. Esto es, añadiremos las siguientes líneas de código justo debajo de la etiqueta TITLE de nuestra página.

```
<link rel="preload" href="https://fonts.googleapis.com">
<link rel="preload" href="https://fonts.gstatic.com/">
<link rel="preload" href="https://cdnjs.cloudflare.com/">
```

## Habilitar la compresión de texto

La compresión de texto es una característica que hace que las páginas se sirvan comprimidas, gracias a los formatos de compresión GZIP y Deflate. Esto supone un ahorro importante en los tiempos de transferencia porque, al enviar los archivos comprimidos, su tamaño se reduce entre un 50 y un 70 por ciento con respecto a su tamaño original y eso se convierte en una menor tasa de transferencia y tiempo de latencia.

Existen varias formas de habilitar la compresión de texto, sin embargo, la más extendida es crear un archivo denominado .HTACCESS en el directorio raíz de nuestro proyecto. Este archivo, nos proporciona una manera de comunicarnos con el Servidor de Aplicaciones y la capacidad de establecer una configuración adecuada a nuestras necesidades.

Cabe destacar que, para que esta característica funcione, se requiere que, en el Servidor de Aplicaciones Apache de nuestro equipo, se habiliten los módulos de DEFLATE y FILTER. Para ello, iremos al archivo HTTPD.CONF, ubicado en el directorio CONF de nuestro Apache y se deberá descomentar las siguientes líneas:

### NOTA

Para nosotros, el archivo HTTPD.CONF debería encontrarse en el directorio C:\XAMPP\APACHE\CONF.

```
LoadModule deflate_module modules/mod_deflate.so
LoadModule filter_module modules/mod_filter.so
```

Una vez habilitados los módulos de Apache y reiniciado el Servidor de Aplicaciones, lo que se deberá hacer es copiar el siguiente fragmento código dentro del .HTACCESS que se acaba de crear.

```
<IfModule mod_deflate.c>
# Compress HTML, CSS, JavaScript, Text, XML and fonts
AddOutputFilterByType DEFLATE application/javascript
AddOutputFilterByType DEFLATE application/rss+xml
AddOutputFilterByType DEFLATE application/vnd.ms-fontobject
AddOutputFilterByType DEFLATE application/x-font
AddOutputFilterByType DEFLATE application/x-font-opentype
AddOutputFilterByType DEFLATE application/x-font-otf
AddOutputFilterByType DEFLATE application/x-font-truetype
AddOutputFilterByType DEFLATE application/x-font-ttf
AddOutputFilterByType DEFLATE application/x-javascript
AddOutputFilterByType DEFLATE application/xhtml+xml
AddOutputFilterByType DEFLATE application/xml
AddOutputFilterByType DEFLATE font/opentype
AddOutputFilterByType DEFLATE font/otf
AddOutputFilterByType DEFLATE font/ttf
AddOutputFilterByType DEFLATE image/svg+xml
AddOutputFilterByType DEFLATE image/x-icon
AddOutputFilterByType DEFLATE text/css
AddOutputFilterByType DEFLATE text/html
AddOutputFilterByType DEFLATE text/javascript
AddOutputFilterByType DEFLATE text/plain
AddOutputFilterByType DEFLATE text/xml
# Remove browser bugs (only needed for really old browsers)
```

```

BrowserMatch ^Mozilla/4 gzip-only-text/html
BrowserMatch ^Mozilla/4\.0[678] no-gzip
BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
Header append Vary User-Agent
</IfModule>
<ifModule mod_gzip.c>
mod_gzip_on Yes
mod_gzip_dechunk Yes
mod_gzip_item_include file .(html?|txt|css|js|php|pl)$
mod_gzip_item_include handler ^cgi-script$
mod_gzip_item_include mime ^text/*.
mod_gzip_item_include mime ^application/x-javascript.*
mod_gzip_item_exclude mime ^image/*.
mod_gzip_item_exclude rspheader ^Content-Encoding:.*gzip.*
</ifModule>
# SVG
AddType image/svg+xml .svg .svgz
AddEncoding x-gzip .svgz

```

## Habilitar la caché

La caché es una memoria intermedia que guarda información con el objetivo de utilizarla en un futuro próximo. Esta información, a menudo viene en forma de archivos de texto que representan todas y cada una de las páginas que se han visitado alguna vez y, en general, disminuye bastante el tiempo de latencia entre páginas.

Por tanto, una buena gestión de los activos estáticos hará que el sistema los sirva de manera más rápida y eficiente. Para ello, volveremos a modificar el archivo .HTACCESS y añadir las siguientes líneas de código:

```

# Un año para las imágenes
<filesMatch ".(jpg|jpeg|png|gif|ico)$">
Header set Cache-Control "max-age=31536000, public"
</filesMatch>
# Un mes para los CSS y JS
<filesMatch ".(css|js)$">
Header set Cache-Control "max-age=2628000, public"
</filesMatch>
<IfModule mod_expires.c>
# Turn on the module.
ExpiresActive on
# Set the default expiry times.
ExpiresDefault "access plus 2 days"
ExpiresByType image/jpg "access plus 1 year"
ExpiresByType image/jpeg "access plus 1 year"
ExpiresByType image/gif "access plus 1 year"
ExpiresByType image/png "access plus 1 year"
ExpiresByType image/webp "access plus 1 year"
ExpiresByType image/svg+xml "access plus 1 year"
ExpiresByType image/icon "access plus 1 year"
ExpiresByType image/x-icon "access plus 1 year"
ExpiresByType text/css "access plus 1 month"
ExpiresByType text/javascript "access plus 1 month"
ExpiresByType application/javascript "access plus 1 month"
ExpiresByType application/x-shockwave-flash "access plus 1 month"
ExpiresByType text/html "access plus 600 seconds"
</IfModule>

```

## El modo keep-alive

La expresión KEEP ALIVE, se refiere al modo de mantener las conexiones con el servidor. Cuando un sitio web tiene establecido el modo Keep Alive, las conexiones que no se finalizan hasta que el cliente o servidor interrumpe la conexión de manera imperativa. Esto mejora el rendimiento y los tiempos de carga.

Para tanto, volveremos a modificar el archivo .htaccess y añadir las siguientes líneas de código:

```
<ifModule mod_headers.c>
Header set Connection keep-alive
</ifModule>
```

### NOTA

Puede que haga falta activar el módulo cabeceras de Apache "headers\_module".

## MEJORANDO LA ACCESIBILIDAD

La accesibilidad Web es un tema que veremos a continuación, sin embargo, solucionar el problema de la accesibilidad para esta página es muy sencillo. Básicamente se trata de establecer una propiedad HTML en unos lugares concretos.

La propiedad ARIA-LABEL es una etiqueta accesible que define una descripción para el elemento que la especifica. Esta propiedad suele ser necesaria cuando el botón o enlace no tienen un texto asociado o cuando tiene texto, pero no está visible en pantalla. Pongamos un ejemplo:

```
<i class="icon menu-hamburguesa"></i>
```

Aunque con leer el atributo CLASS se podría deducir qué ícono es, los crawlers y las herramientas de asistencia, como los lectores de pantalla utilizados por personas con discapacidad o incapacidad visual total o parcial, requieren una descripción clara y legible de su función u objetivo, porque pueden no entender qué significan esas tres barras horizontales.

Si, en vez de eso, declaramos el ícono de la siguiente manera:

```
<i class="icon menu-hamburguesa" aria-label="Acceso al Menú principal"></i>
```

Todos los usuarios, todas las herramientas de asistencia y todos los robots, incluyendo los crawlers, serán capaces de saber inequívocamente cuál es su función u objetivo.

Dicho esto, solucionar el problema de accesibilidad de la página de inicio es tan sencillo definir un ARIA-LABEL en el botón que contiene la clase MENU-TOGGLE y a los enlaces del pie de página que dan acceso a las redes sociales.

## MEJORANDO LAS BUENAS PRÁCTICAS

Si nos fijamos en los comentarios que nos dice la herramienta de Lighthouse, vemos que nos indica que no tiene activado el protocolo de HTTPS/2. Aunque, en una prueba de maquetación no es importante, cuando el sistema está en producción sí que es de vital importancia. Por ello, lo único que se debe hacer para superar este requerimiento es instalar un certificado SSL en nuestro dominio.

Una posible forma de conseguir este objetivo es instalar el software MKCERT desde la dirección <https://github.com/FiloSottile/mkcert/releases>.

Una vez descargado, sólo deberemos ejecutar los comandos expuestos a continuación:

```
C:\Users\NombreUsuario\Downloads> mkcert -install
C:\Users\NombreUsuario\Downloads> mkcert localhost
Using the local CA at "C:\Users\Islavisual\AppData\Local\mkcert" ?
Created a new certificate valid for the following names ??
```

```
- "localhost"  
The certificate is at "./localhost.pem" and the key at "./localhost-key.pem" ?  
C:\Users\NombreUsuario\Downloads>
```

Una vez que hayamos ejecutado ambas instrucciones, cortaremos los archivos “LOCALHOST.PEM” y “LOCALHOST-KEY.PEM” que acabamos de crear y los pegaremos en la carpeta “KEY” del nuestro directorio CONF del servidor de Aplicaciones, en este caso, C:\XAMPP\APACHE\CONF\KEY.

## NOTA

Esto no hay por qué hacerlo desde el símbolo de sistema, se puede realizar a través del Explorador de Windows o cualquier otra aplicación.

Seguidamente, abriremos el archivo de configuración del Servidor de Aplicaciones “HTTPD.CONF” y comprobaremos que tiene descomentadas las líneas siguientes, referentes a los certificados SSL.

```
LoadModule ssl_module modules/mod_ssl.so  
Include conf/extra/httpd-ssl.conf  
LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
```

Y finalmente, modificaremos el archivo HTTPD-SSL.CONF de nuestro Apache a los siguientes valores. En nuestro caso, el archivo se encuentra en “C:\XAMPP\APACHE\CONF\EXTRA\HTTPD-SSL.CONF”.

```
Define SRVROOT "c:\XAMPP\Apache"  
<VirtualHost 127.0.0.1:443 _default_:443>  
DocumentRoot "d:/htdocs"  
ServerName localhost:443  
ServerAdmin islavisual@gmail.com  
ErrorLog "${SRVROOT}\logs\error.log"  
TransferLog "${SRVROOT}\logs\access.log"  
SSLSessionCache "shmcb:${SRVROOT}\logs\ssl_scache(512000)"  
SSLCertificateFile "${SRVROOT}\conf\key\localhost.pem"  
SSLCertificateKeyFile "${SRVROOT}\conf\key\localhost-key.pem"  
SSLCipherSuite ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256  
CustomLog "${SRVROOT}\logs\ssl_request.log" \  
"%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"  
Protocols h2 h2c http/1.1  
</VirtualHost>
```

## NOTA

Para saber si hay algún error en la sintaxis, una vez realizados los cambios en el HTTPD-SSL.CONF se puede ejecutar, desde la ventana de símbolo de sistema, el comando C:\XAMPP\Apache\httpd -t

## MEJORANDO EL SEO

Esta sección se refiere, sobre todo, a la semántica web y el lenguaje utilizado en las páginas. En nuestro caso, para mejorar el SEO sólo hay que cambiar los literales de los enlaces por otros que resulten más legibles para los crawlers.

Aunque, el literal INICIO, es lo suficientemente claro para los usuarios a los que va destinado este sitio web, la realidad es que, para el resto de los usuarios y crawlers, no es nada claro. Por ello, lo único que debemos hacer en esta página para mejorar el SEO y conseguir una puntuación de 100 sobre 100 es cambiar la palabra INICIO por HOME en el menú de navegación declarado en la estructura HEADER del elemento BODY.

## VERIFICANDO LOS CAMBIOS

Si ahora, volvemos a pulsar en el botón de “Generate Report” para comprobar el estado de la página podremos comprobar que:



Cabe destacar que los valores de rendimiento pueden variar en función de la velocidad de acceso a disco, de la memoria RAM, de si se tienen más o menos pestañas abiertas o, incluso, de si es o no un Apache distinto a XAMPP. Por ello, es importante hacer el test en unas condiciones adecuadas y tener en cuenta que, no siempre merece la pena realizar un esfuerzo por conseguir el 100 sobre 100, puesto que estos valores pueden depender, en gran medida, de factores ajenos a la propia página web.

# 12

## ACCESIBILIDAD WEB

### DEFINICIÓN DE ACCESIBILIDAD WEB

Cuando se trata de crear páginas web, a veces, es necesario crearlas de modo que, el contenido, pueda ser accedido / utilizado por todo el mundo, sin importar si posee o no una discapacidad o incapacidad.

Cabe destacar que, siempre es una buena idea crear una página accesible porque, al ser accesible, se volverá más usable y, como consecuencia de ello, la utilizarán más usuarios y, por tanto, tendrá más posibilidades de conseguir nuevos clientes.

Dicho esto, la accesibilidad es la cualidad de accesible, un adjetivo que se refiere a aquello que es de fácil acceso, trato o comprensión. El concepto se utiliza para nombrar al grado en el que todas las personas, más allá de sus capacidades físicas o técnicas, pueden utilizar un cierto objeto o acceder a un servicio.

Existen diversas ayudas técnicas para promover la accesibilidad y equiparar las posibilidades de todas las personas. Esto supone que un lugar que presenta buenas condiciones de accesibilidad puede recibir a toda clase de gente sin que exista un perjuicio o una dificultad para nadie.

Una de estas ayudas técnicas más comunes es lo que se denomina tecnología asistiva. Una tecnología asistiva o de asistencia (TA) es una herramienta utilizada para permitir que personas o usuarios con discapacidad puedan beneficiarse de las mismas ventajas que sus pares sin discapacidad.

Cuando se habla accesibilidad Web, en realidad, se hace referencia a una serie de normas de diseño que van a permitir a todo tipo de usuarios (con o sin discapacidad) percibir, entender, navegar e interactuar con una interfaz o sistema.

Un grupo de estas normas se conocen como Pautas de Accesibilidad para Agentes de Usuario (UAAG) y muestran cómo hacer que las herramientas formadas por navegadores, reproductores multimedia y tecnologías asistivas, entre otras, sean accesibles para personas con discapacidad.

Otro grupo de normas son las denominadas Pautas de Accesibilidad para Herramientas de Autor (ATAG) y tienen como objetivo definir la forma en la que las herramientas ayudan a los desarrolladores o diseñadores a producir un contenido que cumpla todas las Pautas de Accesibilidad al Contenido en la Web (WCAG).

Las ATAG están pensadas principalmente para desarrolladores entre las que se incluyen:

- Editores de HTML y XML de WYSIWYG (What You See Is What You Get).
- Procesadores de texto o paquetes de publicación.
- Herramientas de conversión que transforman formatos de publicación a HTML.
- Edición y producción de vídeo, paquetes de autor de SMIL.
- Gestores de contenido (CMS), herramientas de conversión instantánea o de publicación de sitios Web.
- Herramientas de diseño (SASS, SVG o gráficos vectoriales, minificadores, ...).

## Tipos de discapacidad

Actualmente, muchos de los sistemas, por no decir la mayoría, son inaccesibles (en mayor o menor medida) lo que dificulta o imposibilita la utilización Internet para muchos usuarios con discapacidad.

La accesibilidad Web engloba los tipos de discapacidades en cuatro grandes grupos:

- La **discapacidad visual** es una anomalía parcial o total del sentido de la vista y que puede referirse desde a una pérdida de visión hasta a una sensibilidad especial a la fotografía o a la luz.
- La **discapacidad física** es un tipo de anomalía que imposibilita o dificulta, a quien la padece, el control de las funciones motoras o de su cuerpo.
- La **discapacidad auditiva** es una anomalía parcial o total del sentido del oído y que puede referirse desde a una pérdida de audición parcial, lo que se denomina hipoacusia, hasta a una pérdida total, lo que se conoce como cofosis.
- La **discapacidad intelectual** es una anomalía que imposibilita o dificulta realizar funciones de tipo mental como es el habla, el cuidado personal o la integración social y no tiene por qué estar asociada a ninguna enfermedad o trastorno ya que, mucha de la población mundial, tiene algún tipo de discapacidad intelectual. También se la suele denominar **discapacidad cognitiva** si va referida al desarrollo intelectual y/o la adaptación social de algunas personas.

Viendo la cantidad de discapacidades que existen y la cantidad de usuarios que poseen una o varias de ellas, se hace imperioso la necesidad de suministrar accesibilidad a las interfaces o sistema. No sólo porque aumente su usabilidad, ni porque pueda tener mejor indexación con los motores de búsqueda, sino porque lo importante son los usuarios.

## Ventajas e inconvenientes de la accesibilidad web

Existen múltiples beneficios distribuidos en varias categorizaciones. Por ejemplo, en lo referente a la cuota de mercado y a la audiencia web, la accesibilidad web mejora la Usabilidad Web y el acceso a los contenidos para todos los usuarios, con o sin discapacidad.

En lo referente a la eficiencia y tiempo de respuesta, la accesibilidad puede reducir los costes mantenimiento de las páginas puesto que son menos propensas a tener errores y mejora, en general, el motor de búsqueda haciendo que los resultados sean más legibles y localizables.

En lo referente a la responsabilidad social, puede marcar la diferencia con la competencia, mejora la imagen de negocio y permite llegar a varios millones de personas que poseen algún tipo de discapacidad que les impide o les dificulta su acceso a la web.

En lo referente a los estándares y normativas, puede mejorar varios aspectos de la usabilidad como son la legibilidad y compresión, aumenta la producción de los empleados, ayuda al reconocimiento de las entidades mejorando el acceso a la información y reduce la carga cognitiva.

En cuanto a los inconvenientes, se podría decir que únicamente hay uno y es que requiere un mayor tiempo de desarrollo y eso puede suponer un gran aumento en la inversión del proyecto.

## **TECNOLOGÍAS DÓNDE LA ACCESIBILIDAD ES APLICABLE**

### **HTML y XHTML**

Los lenguajes de marcado HTML y XHTML pueden ser buenos recursos a la hora de hacer una web accesible.

Mientras que HTML está basado en la tecnología denominada Standard Generalized Markup Language (SGML; ISO 8879: 1986), XHTML está basado en Extensible Markup Language, también conocido como XML. La principal diferencia es que XHTML es mucho más estricto y, por ello, algunos métodos pueden ser mucho más difíciles de conseguir, sin contar que, XHTML, no es semántico.

Realizar una web semántica no implica más tiempo de desarrollo, ni más coste que una web no semántica. De hecho, cuando se aplican estructuras semánticas, el desarrollo se vuelve más fácil con el tiempo, se mejora el Posicionamiento SEO y los diseños receptivos se vuelven más sólidos. Además, puede disminuir el tamaño de los archivos y aumentar el rendimiento en general.

No obstante, una web no se vuelve accesible sólo por estar construida bajo una estructura semántica, también necesita de, atributos, propiedades y/o metadatos que mejoren el acceso a los contenidos.

Los datos personalizados son un tipo de atributos que suelen utilizarse para guardar datos privados en las páginas, no obstante, también sirven para asignar descriptores como es el caso de la WAI-ARIA.

Los metadatos y los elementos de cabecera, como puedan ser H1...H6 pueden ser también de gran ayuda en lo referente a mejorar la accesibilidad web, así como la integración con otras tecnologías como JavaScript, CSS o SMIL.

### **CSS**

El lenguaje CSS es un lenguaje de diseño que permite la personalización de documentos estructurados escritos con otro lenguaje de marcado, como pueda ser HTML o XHTML.

El uso de CSS debe intentar utilizarse para contenidos que no sean relevantes, ni tampoco como elemento diferenciador de accesibilidad. Lo que sí se puede hacer es apoyarse en él para aumentar ayudar o aumentar la accesibilidad. Por ejemplo, un contenido no textual decorativo, como pueda ser una imagen de fondo, debe ser expuesta a través de CSS.

No obstante, también tiene otras cosas interesantes, como es el módulo de discurso o CSS Speech Module. Este complemento de CSS permite definir cómo se hablan o pronuncian los elementos de un documento.

Entre otras cosas, permite definir el volumen y distribución espacial de la voz, cómo se debe realizar la descripción auditiva del contenido de voz, dónde, cuándo y de cuánto deben ser los silencios o las pausas antes o después de los elementos, dónde, cuándo y qué sonidos se deben reproducir antes o después de los elementos, el énfasis, velocidad, tipo y género de la voz y los estilos en elementos de tipo lista y contador.

### **JavaScript**

Cuando se desea realizar una web accesible se debe tratar de no abusar del JavaScript porque puede bajar el rendimiento del sistema o interfaz. Además, no debe ser intrusivo, es decir, las funcionalidades de la página deben seguir funcionando, aunque el usuario decida desactivar la interpretación del código JavaScript.

También es importante que las acciones y eventos no se ejecuten por sí solas, es decir, no se deben mostrar diálogos emergentes, anuncios, llamadas a servidor, etc., si no el usuario no lo ha solicitado de manera expresa.

Si la ejecución de una acción implica la apertura de una ventana emergente o nueva, se debe informar previamente al usuario.

Y, cómo no, al igual que sucede con otros lenguajes como HTML y CSS, debe estar separado del resto, es decir, el CSS debe estar en un archivo diferente a los de HTML y JavaScript.

## **Flash**

Si se decide utilizar esta tecnología, que personalmente no la recomiendo porque existen otras opciones mejores, se debe proporcionar equivalencias textuales siempre que se pueda, un contexto de la estructura de la película, en un orden correcto, con todos los controles de animación visibles y proporcionar acceso por teclado a todos los controles que puedan ser manipulados por el dispositivo de puntero.

Además, se deben utilizar todos los controles o funciones relacionados con la accesibilidad web, esto es, se deben utilizar los componentes de simple button, check box, radio button, label, text input, text area, combo box, list box, window, alert y data grid.

También es importante que se proporcionen subtítulos cuando se usen vídeos o audios, que se pueda controlar el vídeo o audio sin que interfiera con los asistentes de voz y, en general, dar soporte a los usuarios con discapacidad total o parcial de visión.

## **PDF**

Un PDF es accesible si el contenido puede ser utilizado por usuarios con o sin discapacidad e independientemente del contexto de uso.

Básicamente, para que un PDF sea accesible, se debe indicar el idioma del archivo, incluir textos alternativos a las imágenes informativas, proporcionar un etiquetado de todos los elementos del documento y asignarle todos los metadatos necesarios para que las tecnologías de asistencia puedan describirlo adecuadamente.

Además, es importante revisar el orden de lectura y la paginación, incluir textos alternativos a todos los enlaces describiendo su objetivo y contexto, asegurarse de que la secuencia de tabulación tiene el orden correcto y que los ajustes de seguridad no interfieran el acceso a la información que debe poder acceder el lector de pantalla o tecnología de asistencia.

## **XML/XSL**

XSL es un conjunto de recomendaciones que se utilizan para definir y mantener la transformación, presentación e interacción de información estructurada, sobre todo, en documentos XML.

Desde que se estandarizó HTML5 y se abandonaron los desarrollos en HTML estricto y transicional, cada vez menos se recurre a este tipo de tecnologías, a no ser que se esté trabajando en arquitecturas requerimientos muy específicos porque JSON es más ligero, rápido y personalizable.

Para mejorar la accesibilidad XSL permite presentar información visual y no visual con pretensiones de ayudar a CSS posibilitando funcionalidades no definidas a través de CSS, como pueda ser la reordenación de elementos.

## **Reproducción multimedia**

Uno de los problemas que tienen los y videos y el multimedia es que no todos los navegadores soportan la reproducción a pantalla completa y, además, tener que personalizarlos de forma corporativa o sofisticada puede volverse una tarea muy ardua y tediosa.

Históricamente, los desarrolladores sólo podían incrustar un archivo que no tenía la posibilidad de reproducirse sin descargarlo por completo, adquirir un desarrollo de terceros (que normalmente no era compatible con todos los navegadores) o utilizar un servidor de medios dedicado (lo que suponía un incremento muy alto de mantenimiento).

Actualmente varias hay opciones para poder realizar streaming desde las interfaces, aunque según qué navegador vaya a reproducir el contenido multimedia, requerirá utilizar uno u otro tipo de codificación diferente.

En lo referente al tema que nos ocupa, al igual que pasa con las animaciones, sliders u otros componentes como son los banners, todos ellos pueden implementar a través de HTML5 y CSS. Eso sí, su implementación debe cumplir con todos los requerimientos descritos en la recomendación WCAG 2.1 comentada anteriormente.

## Otras tecnologías

### SVG

Los Gráficos Vectoriales Escalables o SVG son una forma de crear gráficos más accesibles, rápidos y efectivos. Se podría decir que los principales usuarios que se benefician son sólo los que presentan algún tipo de discapacidad o incapacidad, o aquellos usuarios que disponen de dispositivos y conexiones lentas, sin embargo, nada más lejos.

Por supuesto que beneficia, y mucho, a aquellas personas que presentan una discapacidad visual total o parcial o a los usuarios que usan tecnologías asistivas, pero también los demás usuarios se benefician porque reducen de manera considerable el tamaño de las páginas y transferencias, requieren menos recursos de memoria y CPU y permiten ser mostradas en cualquier resolución sin perder calidad.

La forma de hacer que un gráfico vectorial sea más accesible es proporcionar textos descriptivos en los objetos que indiquen su función, proveer a los controles de cualidades únicas que no se basen únicamente en el color, no incluir texto como paths o imágenes y no utilizar el elemento “g” o descripciones lógicas para cosas que no sean estructurar los documentos.

Además, es recomendable que se utilicen altos contrastes, medidas relativas y, si procede, se representen las relaciones matemáticas con algún lenguaje de marcado matemático como es MathML.

### SILVERLIGHT

Microsoft Silverlight es una herramienta para aplicaciones web del mismo modo que lo hace Adobe Flash. Entre otras cosas, agrega funcionalidades multimedia como la reproducción de vídeos, gráficos vectoriales, animaciones e interactividad.

Aunque es una tecnología que ya no se utiliza en gran medida, las Pautas de Accesibilidad para el Contenido Web contemplan varias casuísticas y proporcionan muchos ejemplos para ayudar a solucionar todos los posibles problemas.

## NORMATIVA Y ESTÁNDARES SOBRE ACCESIBILIDAD WEB

El Portal de la Administración Pública (PAe) pone a disposición de todos los ciudadanos, organismos, empresas y organizaciones el acceso y descarga de todas las normativas y legislación aplicables a la geografía española. La presente redacción es está extraída a febrero de 2020.

### Norma EN 301 549:2018

#### Más información en

<https://administracionelectronica.gob.es/PAe/accesibilidad/une-en-301549-2019.pdf>



La norma EN 301 549:2018, titulada Requisitos de accesibilidad para productos y servicios TIC, actualmente en la versión 2.1.2 especifica los requisitos funcionales de accesibilidad aplicables a los productos y servicios que incluyan TIC (sitios web, software, apps nativas, documentos, hardware, etcétera). Además de describir los procedimientos de prueba y la metodología de evaluación a seguir para cada requisito de accesibilidad.

En esta nueva versión, declarada por la Comisión Europea como estándar armonizado para la aplicación de la Directiva de Accesibilidad Web en la Decisión de Ejecución (UE) 2018/2048 de la Comisión, debe ser aplicada desde el 21 de diciembre de 2018, para todas las Administraciones Públicas españolas.

## **Norma UNE 139803:2012**

### **Más información en**

<http://administracionelectronica.gob.es/PAe/accesibilidad/UNE139803=2012.pdf>



La UNE 139803:2012, titulada, Requisitos de Accesibilidad para contenidos en la web, es una norma que establece los requisitos referentes a las Pautas de Accesibilidad para el Contenido Web (WCAG), de la Iniciativa de Accesibilidad Web (WAI) y del Consorcio de la World Wide Web (W3C). Es equivalente a la WCAG 2.0 AA.

## **Estándar ISO/IEC 40500:2012**

### **Más información en**

[http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=58625](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=58625)



El estándar ISO/IEC 40500:2012, titulada, Pautas de Accesibilidad para el Contenido Web (WCAG) 2.0, cubre una amplia gama de recomendaciones para hacer que el contenido web sea más accesible haciendo que personas con discapacidad auditiva, visual, física, intelectual o cognitiva puedan beneficiarse de Internet.

## **Comparativa de estándares sobre accesibilidad web**

A continuación, se resumen algunas normas gubernamentales sobre los estándares de accesibilidad a nivel mundial. En general, estas normas se suelen aplicar a los sistemas de las agencias gubernamentales más que a sistemas comerciales, a excepción de Australia y Noruega, donde todos los sistemas e interfaces deben cumplir la normativa.

<b>País</b>	<b>Estándar / Legislación</b>
Australia	WCAG 2 AA / <a href="#">Disability Discrimination Act</a>
Canadá	WCAG 2 AA / Human Rights Act 1977
UE	WCAG 2 AA / European Parliament Resolution (2002)
Francia	<a href="#">RGAA 3</a> (basado en WCAG 2) / <a href="#">Law No 2005-102, Article 47</a>
Alemania	<a href="#">BITV 2</a> (basado en WCAG 2) / Federal Disabled Equalization Law (BGG)
Hong Kong	WCAG 2 AA
India	GIGW (basado en WCAG 2 A) / <a href="#">Guidelines for Indian Government Websites</a>

Irlanda	WCAG 2 AA / <a href="#">The Disability Act 2005</a>
Israel	WCAG 2 AA / <a href="#">Equal Rights of Persons with Disabilities Law, 5758-1998</a>
Italia	WCAG 2 / <a href="#">Law No. 4/2004 (Stanca Law)</a>
Japón	<a href="#">X 8341-3:2016</a> (igual a WCAG 2)
Países Bajos	WCAG 2 AA
N. Zelanda	WCAG 2 AA / <a href="#">Human Rights Amendment Act 2001</a>
Noruega	WCAG 2 AA (con excepciones) / <a href="#">LOV 2008-06-20 nr 42</a>
Ontario	<a href="#">AODA</a> (equivalente a WCAG 2 AA)
Quebec	SGQRI 008 (basado en WCAG 2) / <a href="#">Standards sur l'accessibilité du Web</a>
España	WCAG 2 AA (UNE 301 549:2019)
Reino Unido	WCAG 2 AA / <a href="#">Equality Act 2010</a>
USA	Section 508 (basado en WCAG 1) / <a href="#">Section 508 of Rehabilitation Act</a>

Tabla 12.1. Resumen de normas a nivel mundial.

## EL ESTÁNDAR SMIL

### Más información en

<http://www.w3.org/tr/smil3/>



SMIL (Synchronized Multimedia Integration Language) es un estándar de la W3c que está basado en XML y que permite a los diseñadores integrar audio, video, imágenes, texto o cualquier otro contenido multimedia a las interfaces. Ahora mismo está vigente la versión 3.0.

Su antecesor, SMIL1.0 permitía a los desarrolladores o diseñadores describir el comportamiento temporal de la presentación, su disposición en la pantalla y asociar enlaces a los objetos.

Según se ha ido avanzando en número de versión, se han ido ganando nuevas características que han ayudado a la navegación y animación, que han proporcionado soporte para realizar broadcast, han incluido nuevas funcionalidades en el formato visual y un incremento en el rendimiento, característica de gran importancia si se contextualiza en el mundo de los dispositivos móviles.

Finalmente, en diciembre de 2008 aparece la recomendación SMIL 3.0 que se desarrolla pensando en la construcción de aplicaciones multimedia en plataformas que soportan los estándares Web. Por ejemplo, en este nuevo estándar se pueden añadir presentaciones multimedia de forma segura a otras aplicaciones XML, incluyendo HTML y SVG. SMIL 3.0, además, facilita el desarrollo de aplicaciones multimedia sobre plataformas móviles y posee una versión llamada “SMIL Tiny” que es un perfil mínimo de SMIL 3.0 perfecto para sistemas incrustados y aplicaciones ligeras, como reproductores multimedia.

SMIL 3.0 es un estándar que beneficia a todos, pero especialmente, a los usuarios con discapacidad visual ya que permite cubrir sus necesidades de una forma sencilla y organizada.

## Módulos de SMIL

- **Módulo de disposición:** Permite definir las propiedades o atributos para posicionar los contenidos, el orden de visualización en espacios coincidentes, tamaño y posición de las regiones, el color de fondo o la forma de ajuste.

- **Módulo de sincronización:** Permite definir las propiedades o atributos para establecer los valores de inicio y fin por defecto, la duración, el modo de reproducción, el tipo de iteración, número de veces a iterar y los valores máximo y mínimo del objeto multimedia, entre otros.
- **Módulo de animaciones:** Permite cambiar dinámicamente las propiedades de objetos de contenido como el color o posición, el modo de cambio y/o el tipo de cambio.
- **Módulos de control de contenidos:** Permite definir las propiedades o atributos para controlar la representación de uno u otro contenido mediante son el bitrate, el idioma, el tamaño de la pantalla, los subtítulos y/o la CPU.
- **Módulo de enlaces:** Permiten definir algunas propiedades para interactuar con los usuarios.
- **Módulo de metadatos:** Permite definir la descripción del contenido como es el autor, el título o el email, por ejemplo.
- **Módulo de transiciones:** Permite definir cómo se van a realizar las transiciones en el objeto multimedia, su duración, color de desvanecimiento y el modo de transición, entre otros.

Existen bastantes reproductores que permiten leer e interpretar ficheros SMIL y trascibir las acciones que en él se describen.

## Ejemplo de contenido SMIL

```
<smil>
<head>
<meta name="author" content="Pablo Fernández"/>
<meta name="title" content="Ejemplo multimedia"/>
<meta name="copyright" content="(c)2018 PEFC"/>
</head>
<body>
<switch>
<par system-bitrate="700000"><!-- Resoluciones >= 1280x720 -->
<audio src="audio/audioHD.snd"/>
<video src="video/videoHD.avi"/>
<image src="lyrics/imagenHD.jpg"/>
</par>
<par system-bitrate="350000"><!-- Resoluciones >= 320x240 -->
<audio src="audio/audioMobile.snd"/>
<video src="video/videoMobile.avi"/>
<image src="lyrics/imagenMobile.jpg"/>
</par>
</switch>
</body>
</smil>
```

## LA INICIATIVA WAI ARIA

### Más información en

<https://www.w3.org/WAI/standards-guidelines/aria/>



La WAI ARIA (Web Accessibility Initiative Accessible Rich Internet Applications) es una iniciativa del W3C que define o describe una forma de realizar contenidos accesibles. Es muy eficiente con contenidos dinámicos y desarrollos creados bajo los lenguajes HTML, Ajax o JavaScript.

El objetivo principal de este estándar es proporcionar información adicional y útil en las diferentes partes del contenido, sirviendo de ayuda para los usuarios finales que utilizan tecnologías asistivas tales como un lector de pantalla.

La WAI ARIA proporciona una serie de atributos que funcionan como identificadores de las diferentes partes de la aplicación que interactúa con el usuario. También se incluyen mapeo de controles, roles y eventos para la accesibilidad de las APIs (Application Programming Interfaces).

## Partes de la WAI ARIA

WAI ARIA propone a los desarrolladores una serie de soluciones destinadas a hacer accesibles widgets, áreas activas y demás componentes enriquecidos que se encuentran en la mayoría de las aplicaciones web en la actualidad.

Para ello se describen unos roles y propiedades con la finalidad de otorgar de información a los productos de apoyo y para que interactúen adecuadamente con los componentes más normales de las aplicaciones web.

### ROLES

Los roles de WAI ARIA proporcionan un nombre que identifica la funcionalidad de la estructura o contenido. A continuación, se muestra el ejemplo de un widget sencillo en el que se ha querido representar una barra de herramientas con las tres funcionalidades propias del portapapeles.

```
<ul role="toolbar" tabindex="0" aria-activedescendant="copy">
<li id="copy">Copiar</li>
<li id="cut">Cortar</li>
<li id="paste">Pegar</li>
</ul>
```

Código 12.2. Ejemplo de barra de navegación para gestión del portapapeles.

Otro ejemplo de muestra sobre el atributo role es el utilizado en HTML5 para formar estructuras usables y accesibles:

```
<nav id="nav" role="navigation">
<!-- contenido de navegacion -->
</nav>
<section id="main" role="main">
<!-- contenido principal -->
</section>
<div id="banner" role="banner">
<!-- contenido anuncios -->
</div>
```

Código 12.3. Ejemplo de descripción de estructura.

### ESTADOS Y PROPIEDADES

Las propiedades pueden establecer diferentes estados en los componentes, definir regiones dónde actualizar contenidos o describir las funciones de arrastrar y soltar.

A diferencia de los roles que sólo disponen de un atributo para definir los valores, los atributos de estados y propiedades son muchos y cada uno de ellos puede tomar uno o varios valores. Además, algunos de los estados y propiedades son aplicables de manera global a todos los elementos independientemente de si se aplica un rol o no.

```
<h1 id="title1">Vista de la Vía Láctea desde Ávila</h1>
<p id="description">
<!-- contenido de la descripción -->
</p>
<picture>

```

```
</picture>
```

Código 12.4. Ejemplo de descripción de una imagen accesible.

## Atributos de componente

Estos están pensados para apoyar a los roles y para definir elementos de E/S. A continuación, se muestra un ejemplo típico de definición de atributos WAI ARIA para un campo de entrada de tipo texto.

```
<label for="name">Nombre
<input type="text"
id="name"
name="name"
required="required"
aria-required="true" />
</label>
```

Código 12.5. Ejemplo de descripción de campo de texto accesible.

También se pueden establecer atributos que definan las regiones activas que pueden ser actualizadas aun sin hacerse a petición del usuario.

```
<p aria-live="polite">Nombre
<!-- contenido del párrafo -->
</p>
```

Código 12.6. Ejemplo de definición de región activa.

Si observamos el código anterior, al elemento de párrafo se le ha añadido un atributo de región activa ARIA-LIVE con el valor POLITE. Una tecnología asistiva que reconozca el estándar WAI ARIA sabrá que párrafo será un área activa que podrá ser actualizada, en un futuro, con otro contenido.

Si el valor del atributo es POLITE, el contenido podrá ser actualizado una vez haya acabado las tareas que esté haciendo en ese momento el usuario. Si el valor del atributo es ASSERTIVE, el contenido podrá ser actualizado, aunque el usuario no haya terminado las tareas en ese momento.

## ATRIBUTOS DE ARRASTRAR Y SOLTAR

Los atributos de arrastrar y soltar (drag & drop) permiten proporcionar información de cómo se realiza la funcionalidad.

```
<div role="menuitem" aria-dropeffect="copy move">
<!-- contenido del párrafo -->
</div>
```

Código 12.7. Ejemplo de descripción de efecto drag & drop.

En el ejemplo anterior, el rol con valor MENUITEM permite establecer la propiedad ARIA-DROPEFFECT para indicar que tipo de acción acepta el elemento.

## ATRIBUTOS DE RELACIONES

En ocasiones no se pueden establecer, a partir de las estructuras del documento, las relaciones o pertenencias de los elementos que lo forman. Para estas situaciones, el estándar WAI ARIA permite definir las dependencias a través de atributos relacionales. A continuación, se muestra un ejemplo:

Como se ha visto en el código 11.4, se ha descrito un contenido visual a través de los atributos ARIA-DESCRIBEDBY y ARIA-LABELLEDBY del elemento que contiene la imagen. El atributo ARIA-LABELLEDBY ayuda a identificar el contexto y el atributo ARIA-DESCRIBEDBY proporciona la descripción asociada a ese contexto.

## ACCESO MEDIANTE TECLADO

Cuando se habla de accesibilidad es muy frecuente sacar el tema de los atajos de teclado. Siempre que sea posible se deben establecer atajos de teclado. De esta manera, se podrá dar soporte a las personas que no dispongan de ratón, por ejemplo.

En HTML 4, sólo los enlaces, campos de formulario, objetos, áreas y botones podían tomar el foco. En HTML5, todos los elementos pueden adquirir el foco gestionando el atributo TABINDEX que permite establecer un orden específico de navegación. Su valor por defecto u omisión es 0 y significa que la navegación se realizará en el orden de aparición en el documento. Si el valor se establece a -1, el elemento no podrá ser objeto del foco y, por lo tanto, se saltará.

La WAI ARIA, además, permite especificar otros comportamientos asociados a los hijos de los componentes que enriquecen el documento.

## **Soporte en navegadores y productos de apoyo**

El DOM (Document Object Model) contiene la estructura jerárquica y semántica del documento y, uno de sus usos, es para generar componentes propios de las aplicaciones enriquecidas.

Las tecnologías asistivas pueden utilizar el DOM para identificar los objetos, sin embargo, cuanta más información se proporcione a estas aplicaciones, mejor experiencia de usuario tendrá.

Las API de accesibilidad proporcionan los roles, estados, atributos, etcétera para que puedan ser utilizadas por las tecnologías asistivas, como lectores de pantalla. Cuando se utiliza WAI ARIA, la semántica proporcionada debe estar acorde a los valores que se establecen en estas API para obtener un funcionamiento óptimo de las tecnologías asistivas.

La W3C proporciona un documento técnico dónde explica con detalle la asignación de las diferentes características de WAI ARIA con las Accessibility API más [comunes. \*\*http://www.w3.org/TR/wai-aria-implementation/\*\*](http://www.w3.org/TR/wai-aria-implementation/).



Hoy en día, prácticamente existen muchos productos que soportan la implementación de WAI ARIA, incluyendo navegadores, productos de apoyo y otras herramientas de desarrollo.

## **Principales atributos de la WAI-ARIA**

La accesibilidad web es algo sumamente importante y sumamente difícil de aplicar si no se entiende bien lo que se desea hacer. Por ello, antes de nada, vamos a precisar las principales propiedades con las que se puede conseguir proporcionar accesibilidad en los sitios y páginas web.

### **ATRIBUTO ROLE**

Los roles son unos atributos que se establecen para indicar la función u objetivo del elemento. Esto se vuelve necesario porque, en ocasiones, no es fácil discernir la diferencia u objeto del elemento mostrado. Por ejemplo, no es lo mismo un elemento que representa a una barra de progreso, que un elemento que representa a una barra de carga en proceso.

Los roles pueden ser de dos tipos. De tipo interfaz, que son lo que representan a elementos como árboles, listas, sliders, diálogos emergentes, etcétera y, de tipo estructural, que son los que representan o definen una estructura como pueda ser un menú de navegación o una cabecera o pie de página.

Los roles, deben asignarse a los botones, enlaces, elementos de formulario, iconos y regiones los atributos necesarios de manera que se proporcione una información suficiente sobre lo que representa el elemento.

La forma de hacer esto es establecer ARIA-ROLE a alguno de los siguientes valores:

- **BANNER:** Indica que la región o sección contiene el título principal o el título interno de la página.

- **COMPLEMENTARY:** Indica que es una sección que tiene contenido principal, pero es independiente y significativa por sí sola.
- **CONTENTINFO:** Indica que la sección contiene información relevante sobre el documento principal, como derechos de autor o enlaces a declaraciones de privacidad.
- **FORM:** Indica que la sección representa una colección de elementos de formulario, sean editables o no.
- **MAIN:** Indica la sección de contenido principal en un documento. Aunque puede darse el caso de que no, por norma general, una página tendrá una única región establecida a este valor.
- **NAVIGATION:** Indica que la sección contiene una colección de enlaces o acciones pensados para navegar por el sitio web.
- **SEARCH:** Indica que la sección o elemento tiene la función de buscador para el sitio web.
- **APPLICATION:** Indica que la región es una aplicación web, en vez de un documento web.

### **Ejemplo:**

```
<div id="header" role="banner">...</div>
<div id="sitelookup" role="search">...</div>
<div id="nav" role="navigation">...</div>
<div id="content" role="main">...</div>
<div id="rightsideadvert" role="complementary">...</div>
<div id="footer" role="contentinfo">...</div>
```

### **ATRIBUTO ARIA-LABEL**

El atributo ARIA-LABEL permite establecer un nombre accesible a aquellos elementos que no poseen una descripción textual en su contenido.

Como se vio en la práctica del capítulo de Usabilidad Web, es frecuente utilizarlo cuando los elementos representan una imagen o ícono, pero también es usable en cualquier elemento que requiera una descripción adicional que permita contextualizar y dar un significado inequívoco.

### **Ejemplo:**

```
<a href="#" aria-label="Cerrar">X</a>
```

### **ATRIBUTO ARIA-LABELLEDBY**

El atributo ARIA-LABELLEDBY permite establecer unos identificadores que serán utilizados para generar una descripción accesible. Estos identificadores deben ser valores válidos definidos por atributos ID alcanzables en el mismo contexto. Es decir, no se pueden utilizar valores de ID que estén ubicados o definidos en otra ventana o frame diferente a la actual.

Esto suele ser útil en situaciones donde el contenido a mostrar es un enlace a un archivo descargable (como un PDF o Excel) o cuando el elemento al que hace referencia no está dentro del mismo contenedor.

### **Ejemplo:**

```
<p id="tInforme">Exportar informe en:</p>
<div class="export">
<a href="./mayo.pdf" id="pdf" aria-labelledby="tInforme pdf">PDF</a>
<a href="./mayo.xls" id="xls" aria-labelledby="tInforme xls">Excel</a>
</div>
```

### **ATRIBUTO ARIA-DESCRIBEDBY**

El atributo ARIA-DESCRIBEDBY permite establecer una descripción larga o detallada para todos aquellos elementos mediante el establecimiento del valor de un ID válido. Este identificador debe ser un valor válido definidos por un atributo

ID alcanzables en el mismo contexto. Es decir, no se pueden utilizar valores de ID que estén ubicados o definidos en otra ventana o frame diferente a la actual.

Esto suele ser útil en situaciones donde el contenido a mostrar es un enlace a un archivo descargable (como un PDF o Excel) o cuando el elemento al que hace referencia no está dentro del mismo contenedor.

### Ejemplo:

```
<form role="form">
<label for="username">Nombre de Usuario</label>
<input type="text" id="usr" name="usr" required />
<label for="username">Contraseña</label>
<input type="password" id="pwd" name="pwd" required
aria-describedby="pwdDesc" />
<p id="pwdDesc">La contraseña debe tener, como mínimo, una mayúscula, una minúscula, un número y un carácter especial. Además, no puede tener una longitud menor a 6 caracteres.</p>
<button type="submit" class="accept rounded">Acceder</button>
</form>
```

## ATRIBUTO ARIA-DESCRIBEDAT

El atributo ARIA-DESCRIBEDAT tiene un comportamiento idéntico al atributo ARIA-DESCRIBEDBY, si exceptuamos que lo que se establece es una URL, en vez de un ID.

Esto suele ser útil cuando la descripción no es una frase, sino una descripción larga o representa una explicación que conlleva varios párrafos.

### Ejemplo:

```
<a href="#" aria-describedby="https://es.wikipedia.org/wiki/Magnitud_aparente">
Magnitud aparente
</a>
```

## ATRIBUTO ARIA-LIVE

El atributo ARIA-LIVE permite identificar aquellas zonas del documento que pueden ser actualizadas de forma dinámica o automática. Esto, hace posible que se les notifique a las herramientas de asistencia situaciones en las que se inyectan contenidos dinámicamente a un contenedor actualizable. Las herramientas de asistencia leen automáticamente el contenido de este contenedor (denominado región activa o “Live Region”) y evitan tener que centrarse en dónde se producen los cambios.

Recordemos que, si el valor del atributo ARIA-LIVE es POLITE, el contenido podrá ser actualizado una vez haya acabado las tareas que esté haciendo en ese momento el usuario, pero, si el valor del atributo es ASSERTIVE, el contenido podrá ser actualizado, aunque el usuario no haya terminado las tareas en ese momento.

Este atributo suele combinarse con el atributo ARIA-ATOMIC para indicar si la actualización implica a toda la región o sólo a partes concretas. También, suele combinarse con el atributo ARIA-RELEVANT, el cual, indica qué tipo de actualización debe realizarse o se espera.

## ATRIBUTO ARIA-ATOMIC

El atributo ARIA-ATOMIC permite establecer si la actualización de un contenedor afecta a todas o sólo a algunas partes. Esta actualización será revelada en función de las notificaciones de cambio definidas por el atributo ARIA-RELEVANT.

### Ejemplo:

```
<h3>Contenido del carrito</h3>
<div aria-live="polite" aria-atomic="true">
<div>Su cesta de la compra contiene
<span id="nArt">0</span>
Artículos
```

```
</div>  
</div>
```

## ATRIBUTO ARIA-RELEVANT

El atributo ARIA-RELEVANT permite establecer qué tipo de cambios se va a realizar en un contenedor que posee el atributo ARIA-LIVE establecida a un valor diferente a OFF.

El atributo ARIA-RELEVANT admite una lista de valores prefijados separados por espacios, la cual se muestra a continuación:

- **ADDITIONS**: Indica que sólo es relevante los nodos que sean agregados a la región activa.
- **REMOVALS**: Indica que sólo es relevante los nodos que sean eliminados a la región activa.
- **TEXT**: Indica que sólo es relevante aquellos cambios que afecten a los contenidos textuales dentro de la región activa.
- **ALL**: Indica que es relevante cualquier cambio, sea aditivo, de eliminación o textual.

### Ejemplo:

```
<h3>Contenido del carrito</h3>  
<div aria-live="polite" aria-atomic="true" aria-relevant="all">  
<div>Su cesta de la compra contiene  
<span id="nArt">0</span>  
Artículos  
</div>  
</div>
```

## ATRIBUTO ARIA-EXPANDED

El atributo ARIA-EXPANDED permite establecer si el elemento que representa a un contenido plegable o colapsable como pueda ser un menú desplegable de navegación, está expandido o contraído.

Por ejemplo, un menú lateral deslizante, también conocido como Off-Screen Menu, con un botón tipo hamburguesa podría definirse de la siguiente manera:

### Ejemplo:

```
<button class="menu-toggle"  
aria-label="Menú principal"  
aria-expanded="false">  
<i class="icon bars">≡</i>  
</button>  
<aside class="hidden" role="navigation">  
Ejemplo de menú  
<ul>  
<li><a href=".//home.html">Home / Inicio</a></li>  
<li><a href=".//quienes-somos.html">Quienes Somos</a></li>  
<li><a href=".//servicios.html">Servicios</a></li>  
<li><a href=".//donde-estamos.html">Dónde estamos</a></li>  
</ul>  
</aside>
```

Al pulsar en el botón, el sistema podría realizar la actualización del atributo ARIA-EXPANDED en el elemento a través de JavaScript y, estableciéndolo a TRUE si está mostrando el ASIDE, o a FALSE si está oculto.

## ATRIBUTO ARIA-REQUIRED

El atributo ARIA-REQUIRED permite indicarles a las herramientas de asistencia que el elemento conlleva una entrada de datos que es obligatoria para continuar o finalizar la tarea actual. Básicamente, es lo mismo que el atributo REQUIRED, sin

embargo, ARIA-REQUIRED puede ser útil cuando no se desea realizar la validación nativa del navegador o herramienta de asistencia.

En general, se suele establecer en tiempo de ejecución, tras un proceso de validación.

#### Ejemplo:

```
<label for="nombre">Nombre completo:</label>
<input id="nombre" type="text" aria-required="true" />
```

### ATRIBUTO ARIA-INVALID

El atributo ARIA-INVALID permite indicarles a las herramientas de asistencia que el elemento de formulario es un campo de entrada no cumple con las expectativas o formato indicados.

Esto es útil cuando se están solicitando datos preformateados como puedan ser los correos electrónicos, teléfonos o cualquier otro tipo de entrada que pueda responder a una posible máscara de entrada, pero también es útil para indicar que no está lleno y que, por tanto, es obligatorio.

Los posibles valores que puede tomar el atributo ARIA-INVALID son FALSE, para indicar que no se detectaron errores, GRAMMAR, para indicar que se detectó un error gramatical, SPELLING, para indicar que se detectó un error ortográfico o, simplemente, TRUE, para indicar que el proceso de validación no fue superado.

En general, se suele establecer en tiempo de ejecución, tras un proceso de validación.

#### Ejemplo:

```
<label for="nombre">Nombre completo:</label>
<input id="nombre" type="text"
aria-required="true"
aria-invalid="false"
oninput="checkValidity(this)" />
<script>
function checkValidity(el){
var invalid = (el.value.trim().length == 0);
if (invalid) {
el.setAttribute("aria-invalid", "true");
} else {
el.setAttribute("aria-invalid", "false");
}
}
</script>
```

### ATRIBUTO ARIA-CHECKED

El atributo ARIA-CHECKED permite establecer el estado de aquellos elementos de formulario que resultan ser de tipo casilla de verificación o de tipo radio. Básicamente, el atributo ARIA-CHECKED es idéntico al atributo CHECKED de HTML, salvo por la diferencia de que ARIA-CHECKED permite establecer un estado adicional que indica no es ni activado, ni desactivado.

Los posibles valores que puede tomar el atributo ARIA-CHECKED son FALSE, para indicar que no está verificado o seleccionado, TRUE, para indicar que está verificado o seleccionado y MIXED, para indicar que la verificación o selección es sólo parcial.

Al margen de los valores de estado TRUE y FALSE, el valor de estado MIXED puede ser útil cuando el estado de la casilla de verificación hace referencia a un conjunto de elementos de su mismo tipo en donde hay algunos seleccionados y otros que no.

#### Ejemplo:

```
<table>
<thead>
```

```

<tr>
<th>
<input type="checkbox" id="checkAll" aria-checked="mixed"/>
</th>
<!-- ... otros elementos TH -->
</tr>
</thead>
<tbody>
<tr>
<td>
<input type="checkbox" id="chk01" aria-checked="true"/>
</td>
<!-- ... otros elementos TD -->
</tr>
<tr>
<td>
<input type="checkbox" id="chk02" aria-checked="false"/>
</td>
<!-- ... otros elementos TD -->
</tr>
<!-- ... otros elementos TR -->
</tbody>

```

## ATRIBUTO ARIA-OWNS

El atributo ARIA-OWNS permite indicarles a las herramientas de asistencia que el elemento está vinculado o asociado con otro elemento que se encuentra fuera de la estructura actual o separado en otra zona del documento.

Esto es útil, por ejemplo, cuando se están definiendo menús de navegación que poseen varios niveles.

### Ejemplo:

```

<nav role="navigation">
<ul role="menu">
<li role="menuitem" aria-owns="submenu-new">Nuevo</li>
<li role="menuitem">Abrir</li>
<li role="menuitem">Guardar</li>
<li role="menuitem">Guardar como ...</li>
<li role="menuitem">Información del documento</li>
</ul>
<ul id="submenu-new" role="menu">
<li role="menuitem">Archivo</li>
<li role="menuitem">Carpeta</li>
<li role="menuitem">Proyecto</li>
</ul>
</nav>

```

## ATRIBUTOS ARIA-AUTOCOMPLETE Y ARIA-ACTIVEDESCENDANT

El atributo ARIA-AUTOCOMPLETE permite indicarles a las herramientas de asistencia que el elemento conlleva una búsqueda predictiva con una posterior visualización de resultados total o parcial. Además, permite establecer el tipo de interacción que está asociado al elemento de formulario.

Los posibles valores que puede tomar el atributo ARIA-AUTOCOMPLETE son el valor INLINE, para indicar que el valor resultante de la búsqueda predictiva se mostrará dentro del elemento de formulario al que está asociado, LIST, para indicar que el resultado de la búsqueda predictiva se mostrará en un elemento de lista a parte o BOTH, para indicar que el elemento de formulario ofrece ambos modelos al mismo tiempo.

### Ejemplo:

```

<input id="cb1-edit"
type="text"
aria-activedescendant="opto4"

```

```
aria-owns="resultados-busqueda"
aria-autocomplete="list"
role="combobox" />
<ul aria-expanded="true" role="listbox" id="resultados-busqueda">
<li role="option" id="opto1">HTML para todos</li>
<li role="option" id="opto2">La guía oficial de HTML5</li>
<li role="option" id="opto3">Creación de páginas con HTML5</li>
<li role="option" id="opto4">Accesibilidad Web y HTML5</li>
<li role="option" id="opto5">La usabilidad de HTML5</li>
</ul>
```

Como puede apreciarse, el atributo ARIA-ACTIVEDESCENDANT permite establecer el elemento de la lista que, actualmente, está seleccionado.

## ATRIBUTO TABINDEX

El atributo TABINDEX de HTML es fundamental para poder gestionar con eficiencia el foco del teclado en elementos como enlaces o elementos de formulario.

Cierto es que es posible establecerla en todo tipo de elementos, no obstante, no suele ser una buena opción utilizarlo en elementos que, por definición, no pueden tomar el foco si lo que se desea es que la página sea totalmente accesible.

Aun así, recordemos que el atributo TABINDEX puede tener tres posibles valores. Cuando es "0" toma el elemento puede tomar el foco en el orden de definición o aparición de los elementos. Cuando es "-1" el elemento no puede tomar el foco de ninguna de las maneras y, cuando es otro valor, define un orden de tabulación explícito que será efectivo en función su cuantía, es decir, a menor valor será encocado antes, a mayor valor será enfocado después y, a igual valor, será enfocado por orden de aparición.

### Ejemplo:

```
<i class="icon menu" tabindex="0" onclick="menuToggle(this)">=</i >
```

## LA RECOMENDACIÓN WCAG

La recomendación WCAG ha pasado por varias revisiones. En su versión 1.0, se establecieron una lista de pautas aplicables en el ámbito internacional sobre cómo hacer accesibles los contenidos de la Web a las personas con discapacidad. Esta lista de pautas está englobada en la Iniciativa para la Accesibilidad Web (WAI) del Consorcio de la Web (W3C).

Más tarde, en diciembre de 2008 evolucionaron hasta la versión 2.0, donde se contemplaban nuevas funcionalidades y se sugerían nuevos requerimientos debido, fundamentalmente, a los nuevos factores tecnológicos y la experiencia ya adquirida sobre del uso de las WCAG 1.0.

La recomendación vigente, la WCAG 2.1, es básicamente, una revisión que está pensada para mejorar el acceso a los contenidos para personas con discapacidad cognitiva o del aprendizaje, con baja visión o a cualquier otra persona que tenga discapacidad y acceda desde un dispositivo móvil.

## Principios y directrices

### Perceptible

La información y los componentes de la interfaz deben ser visualizados de forma que los usuarios puedan percibirlos.

- **Pauta 1.1:** Proporcionar alternativas textuales para todos los contenidos que no son textuales para que puedan ser reinterpretados por medio de otros formatos como descripciones ampliadas, braille, lectura por voz, lenguaje de signos u otros lenguajes más sencillos.
- **Pauta 1.2:** Proporcionar alternativas para contenidos multimedia dependientes del tiempo en forma sincronizada.

- **Pauta 1.3:** Proporcionar contenidos que puedan presentarse de diferentes maneras sin que se pierda información o estructura.
- **Pauta 1.4:** Facilitar a los usuarios la posibilidad de ver y oír todo el contenido importante sin distinción de si está en primer plano o no.

## Operable

Los componentes de la interfaz de usuario y la navegación deben ser utilizables.

- **Pauta 2.1:** Proporcionar acceso a todas las funcionalidades a través de atajos de teclado.
- **Pauta 2.2:** Establecer unos tiempos suficientemente extensos como para leer y utilizar los contenidos.
- **Pauta 2.3:** Proporcionar contenidos que no provoquen convulsiones o ataques epilépticos.
- **Pauta 2.4:** Proporcionar ayudas adecuadas para que los usuarios puedan navegar, encontrar contenido y saber dónde se encuentran de forma sencilla.
- **Pauta 2.5:** Facilitar a los usuarios la interoperabilidad a través de diferentes métodos de entrada, incluyendo el teclado.

## Comprensible

La información y el control de la interfaz deben ser entendibles.

- **Pauta 3.1:** Asegurase de que los contenidos textuales sean legibles y comprensibles.
- **Pauta 3.2:** Asegurase de que las interfaces de usuario sean utilizadas y presentadas de una forma predecible.
- **Pauta 3.3:** Ayudar a los usuarios a evitar y corregir los errores.

## Robusto

El contenido deber ser lo más robusto posible para que los agentes de usuario y tecnologías asistivas puedan interpretarlo.

- **Pauta 4.1:** Maximizar y posibilitar la compatibilidad con las aplicaciones actuales y futuras de los usuarios.

## Criterios de Conformidad por nivel de adecuación

Existen 3 niveles de adecuación. Para conseguir pasar cada uno de ellos, se tienen que cumplir una lista de criterios de conformidad.

### Criterios a cumplir para pasar el nivel de adecuación A

Perceptible 1.1.1, 1.2.1, 1.2.2, 1.2.3, 1.3.1, 1.3.2, 1.3.3, 1.4.1 1.4.2.

Operable 2.1.1, 2.1.2, 2.1.4, 2.2.1, 2.2.2, 2.3.1, 2.4.1, 2.4.2, 2.4.3, 2.4.4, 2.5.1, 2.5.2, 2.5.3, 2.5.4.

Comprensible 3.1.1, 3.2.1, 3.2.2, 3.3.1, 3.3.2.

Robusto 4.1.1, 4.1.2

### Criterios a cumplir para pasar el nivel de adecuación AA

Perceptible 1.2.4, 1.2.5, 1.3.4, 1.3.5, 1.4.3, 1.4.4, 1.4.5, 1.4.10, 1.4.11, 1.4.12, 1.4.13.

Operable 2.4.5, 2.4.6, 2.4.7.

Comprensible 3.1.2, 3.2.3, 3.2.4, 3.2.5, 3.3.3, 3.3.4.

Robusto 4.1.3

#### **Criterios a cumplir para pasar el nivel de adecuación AAA**

Perceptible 1.2.6, 1.2.7, 1.2.8, 1.2.9, 1.3.6, 1.4.6, 1.4.7, 1.4.8, 1.4.9.

Operable 2.1.3, 2.2.3, 2.2.4, 2.2.5, 2.2.6, 2.3.2, 2.3.3, 2.4.8, 2.4.9, 2.4.10, 2.5.5, 2.5.6.

Comprendible 3.1.3, 3.1.4, 3.1.5.

## **Descripción de las pautas de accesibilidad web**

A continuación, se describen los Criterios de Conformidad de la WCAG 2.1 extraídos de la herramienta web “Cómo cumplir con WCAG (Referencia Rápida)”, propiedad de la W3.org. Esta herramienta proporciona una lista navegable con todas las Pautas de Accesibilidad al Contenido Web, las técnicas para su implementación e información de apoyo para comprender el estándar WCAG 2. Cabe destacar que, en este capítulo, se exponen sólo los enunciados con referencia hacia los originales.

### **PERCEPTIBLE**

#### **CC-1.1.1. Contenido sin texto**

Pertenece al nivel de adecuación A y su objetivo es hacer que la información transmitida por el contenido no textual sea accesible a través del uso de una alternativa de texto. En pocas palabras, todos los elementos que no tienen texto, como pueda ser una imagen, deben tener un atributo o propiedad que los describa.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/non-text-content.html>

#### **CC-1.2.1. Sólo audio y sólo vídeo (pregrabado)**

Pertenece al nivel de adecuación A y su objetivo es hacer que la información transmitida en las páginas a través de únicamente audio y/o vídeo pregrabado esté disponible para todos los usuarios. Un ejemplo de vídeo pregrabado sin la interacción del usuario o información de audio es una película muda.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/audio-only-and-video-only-prerecorded.html>

#### **CC-1.2.2. Subtítulos (pregrabados)**

Pertenece al nivel de adecuación A y su objetivo es permitir a las personas sordas o con problemas auditivos ver presentaciones multimedia de forma sincronizada a través de subtítulos, excepto cuando es un contenido multimedia alternativo al texto y se encuentra claramente identificado como tal.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/captions-prerecorded.html>

#### **CC-1.2.3. Descripción de audio o medio alternativo (pregrabado)**

Pertenece al nivel de adecuación A y su objetivo es proveer de una alternativa para los medios basados en tiempo o, en su defecto, una descripción de audio del contenido que está sincronizado con el vídeo, excepto cuando ese contenido multimedia sea alternativo al texto y está claramente identificado como tal.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/audio-description-or-media-alternative-prerecorded.html>

#### **CC-1.2.4. Subtítulos (en directo)**

Pertenece al nivel de adecuación AA y su objetivo es que las personas que presentan una discapacidad auditiva puedan ver en tiempo real las presentaciones a través de subtítulos que describan todo lo que sucede, incluyendo efectos de sonido.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/captions-live.html>

#### **CC-1.2.5. Descripción de audio (pregrabado)**

Pertenece al nivel de adecuación AA y su objetivo es proporcionar a aquellas personas que tengan una deficiencia visual total o parcial una descripción de audio con toda la información visual importante de una presentación multimedia de forma sincronizada.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/audio-description-prerecorded.html>

#### **CC-1.2.6. Lenguaje de signos (pregrabado)**

Pertenece al nivel de adecuación AAA y su objetivo es permitir a las personas sordas o con discapacidad auditiva y que tienen fluidez con el lenguaje de signos entender el contenido de la pista de audio en presentaciones multimedia.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/sign-language-prerecorded.html>

#### **CC-1.2.7. Descripción de audio extendida (pregrabada)**

Pertenece al Nivel de adecuación AAA y su objetivo es proveer a aquellas personas con deficiencia visual parcial o total una presentación multimedia sincronizada más detallada que lo que podría ser la descripción de audio estándar.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/extended-audio-description-prerecorded.html>

#### **CC-1.2.8. Alternativa de medios (pregrabado)**

Pertenece al nivel de adecuación AAA y su objetivo es proporcionar una alternativa audiovisual para aquellas personas que poseen una discapacidad visual tal que no pueden leer de forma fiable los subtítulos y para aquellas personas que poseen una discapacidad auditiva tal que no pueden escuchar de forma fiable el diálogo y la descripción de audio.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/media-alternative-prerecorded.html>

#### **CC-1.2.9. Sólo Audio (en directo)**

Pertenece al nivel de adecuación AAA y su objetivo es conseguir que la información auditiva que se transmite en directo, como pueda ser una videoconferencia, un podcast o una retransmisión en streaming, sea accesible a través de una alternativa de texto.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/audio-only-live.html>

### **CC-1.3.1. Información y relaciones**

Pertenece al nivel de adecuación A y tiene como objetivo garantizar que la información, estructura y relaciones implícitas en formato visual o auditivo pueden ser determinadas mediante programación y permiten su conservación cuando cambia el formato de la presentación.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/info-and-relationships.html>

### **CC-1.3.2. Secuencia significativa**

Pertenece al nivel de adecuación A y su objetivo es proporcionar una forma alternativa de presentar los contenidos cuando la secuencia afecta a la lectura.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/meaningful-sequence.html>

### **CC-1.3.3. Características sensoriales**

Pertenece al nivel de adecuación A y su objetivo es asegurar que todos los usuarios tienen las instrucciones necesarias para usar el contenido, incluso cuando no pueden percibir la forma, el tamaño, localización espacial o la orientación.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/sensory-characteristics.html>

### **CC-1.3.4. Orientación**

Pertenece al nivel de adecuación AA y su objetivo es garantizar que el contenido no está restringido a una única orientación de pantalla, a no ser que la orientación de la pantalla sea un requisito.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/orientation.html>

### **CC-1.3.5. Identificar el propósito de entrada**

Pertenece al nivel de adecuación AA y tiene como objetivo garantizar que el propósito de cada elemento de entrada de formulario que recupere información sobre el usuario pueda ser determinado por código.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/identify-input-purpose.html>

### **CC-1.3.6. Identificar propósito**

Pertenece al nivel de adecuación AAA y tiene como objetivo garantizar que el propósito de los elementos que componen y estructuran la página puedan ser determinados a través de código.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/identify-purpose.html>

### **CC-1.4.1. Uso del color**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar que todos los usuarios puedan acceder a la información que se transmite independientemente del color.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/use-of-color.html>

#### **CC-1.4.2. Control del audio**

Pertenece al nivel de adecuación A y su objetivo es controlar el audio de una página web cuando se reproduce durante un tiempo superior a 3 segundos. Si esto sucede, se debe proporcionar un mecanismo para que, los usuarios, puedan pausar o detener el audio y, en su defecto, proporcionar una forma de controlar el volumen del audio de manera independiente al volumen general del dispositivo o sistema.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/audio-control.html>

#### **CC-1.4.3. Contraste (mínimo)**

Pertenece al nivel de adecuación AA y tiene como objetivo proporcionar un contraste suficiente entre el texto y el fondo de manera que pueda ser leído por personas con visión moderadamente baja que no utilizan la tecnología de asistencia para mejorar el contraste.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/contrast-minimum.html>

#### **CC-1.4.4. Cambiar el tamaño del texto**

Pertenece al nivel de adecuación AA y tiene como objetivo proporcionar un método de escalado para el texto sin que sea necesario utilizar una tecnología de asistencia.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/resize-text.html>

#### **CC-1.4.5. Imágenes de texto**

Pertenece al nivel de adecuación AA y tiene como objetivo impulsar a los desarrolladores a que utilicen fuentes de iconos vectoriales u otras tecnologías, que igualmente pueden producir una buena presentación visual, para conseguir que los usuarios puedan ajustarlo a sus necesidades.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/images-of-text.html>

#### **CC-1.4.6. Contraste (mejorado)**

Pertenece al nivel de adecuación AAA y tiene como objetivo proporcionar suficiente contraste entre el texto y el fondo para que pueda ser leído por personas con visión moderadamente baja que no utilizan la tecnología de asistencia de mejora del contraste.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/contrast-enhanced.html>

#### **CC-1.4.7. Audio de fondo bajo o nulo**

Pertenece al nivel de adecuación AAA y tiene como objetivo garantizar que el sonido de la voz que está en primer plano esté lo suficientemente diferenciada como para que un usuario con problemas de audición graves pueda ser capaz de separarla del sonido de fondo.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/low-or-no-background-audio.html>

#### **CC-1.4.8. Presentación Visual**

Pertenece al nivel de adecuación AAA y tiene como objetivo garantizar que la renderización visual de texto se presente de una manera tal que pueda ser percibido sin que el formato pueda interferir con su legibilidad.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/visual-presentation.html>

#### **CC-1.4.9. Imágenes de texto (sin excepción)**

Pertenece al nivel de adecuación AAA y tiene como objetivo permitir a las personas que requieren que el texto tenga un tamaño, color, tipo de fuente, interlineado o alineación determinados.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/images-of-text-no-exception.html>

#### **CC-1.4.10. Reajustar contenido**

Pertenece al nivel de adecuación AA y tiene como objetivo asegurar que el contenido puede presentarse sin pérdida de información y sin requerir un desplazamiento en dos dimensiones.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/reflow.html>

#### **CC-1.4.11. Contraste sin texto**

Pertenece al nivel de adecuación AA y tiene como objetivo asegurar que la presentación visual tiene un contraste mínimo de, al menos, 3:1 en referencia a los colores adyacentes.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/non-text-contrast.html>

#### **CC-1.4.12. Espaciado del texto**

Pertenece al nivel de adecuación AA y tiene como objetivo asegurar que, el lenguaje de marcado utilizado (HTML XHTML, etcétera) no produce pérdida de contenido o funcionalidad a causa de las propiedades de estilo.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/text-spacing.html>

#### **CC-1.4.13. Contenido flotante (hover) y enfocado (focus)**

Pertenece al nivel de adecuación AA y tiene como objetivo asegurar que, los contenidos adicionales que aparecen y desaparecen al mover el puntero del ratón, o al tomar el foco en un elemento concreto puedan percibirse correctamente y descartarse sin pérdida de experiencia sobre la página.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/content-on-hover-or-focus.html>

## **OPERABLE**

### **CC-2.1.1. Teclado**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar, siempre que sea posible, que el contenido puede ser operado a través del teclado o una interfaz de teclado. Esto no implica que no pueda ser operable mediante ratón o un dispositivo de puntero.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/keyboard.html>

### **CC-2.1.2. Sin trampas para el teclado**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar que los contenidos no se vuelvan una “trampa” de teclado en ninguna parte del contenido de una página Web.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/no-keyboard-trap.html>

### **CC-2.1.3. Teclado (sin excepción)**

Pertenece al nivel de adecuación AAA y su objetivo es asegurar que todo el contenido de la página sea operable desde el teclado sin excepción alguna.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/keyboard-no-exception.html>

### **CC-2.1.4. Atajos de teclado**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar que todos los elementos que definen un atajo de teclado son sólo activados cuando toman el foco o, en su defecto, disponen de un mecanismo para desactivarlos o reasignarlos.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/character-key-shortcuts.html>

### **CC-2.2.1. Tiempo ajustable**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar que los usuarios con discapacidad dispongan de tiempo suficiente para interactuar con el contenido web siempre que sea posible.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/timing-adjustable.html>

### **CC-2.2.2. Pausar, detener, ocultar**

Pertenece al nivel de adecuación A y tiene como objetivo evitar distracciones a los usuarios durante su interacción con una página Web en lo que a movimientos, parpadeos o desplazamientos se refiere. Ejemplos de ello son cotizaciones en bolsa, videos, juegos en tiempo real, animaciones, etc.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/pause-stop-hide.html>

### **CC-2.2.3. Sin tiempo**

Pertenece al nivel de adecuación AAA y tiene como objetivo minimizar la aparición de contenido que requiere la interacción cronometrada. Esto beneficia a las personas con ceguera, baja visión, limitaciones cognitivas, motoras o con impedimentos para interactuar con el contenido.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/no-timing.html>

### **CC-2.2.3. Sin tiempo**

Pertenece al nivel de adecuación AAA y tiene como objetivo permitir a los usuarios manipular, posponer o desactivar las actualizaciones automáticas del contenido de una web, u otras interrupciones que no sean de emergencia.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/interruptions.html>

### **CC-2.2.5. Volver a autenticar**

Pertenece al nivel de adecuación AAA y su objetivo es permitir que todos los usuarios puedan completar las transacciones en sesiones con autenticación y que tienen establecido un límite de tiempo de inactividad.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/re-authenticating.html>

### **CC-2.2.6. Tiempos de espera**

Pertenece al nivel de adecuación AAA y tiene como objetivo asegurar que los usuarios están informados sobre la duración de cualquier inactividad que pueda generar una pérdida de datos, a no ser que los datos estén disponibles durante más de 20 horas.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/timeouts.html>

### **CC-2.3.1. Umbral de tres destellos o menos**

Pertenece al nivel de adecuación A y tiene como objetivo garantizar que las páginas web no contengan nada que parpadee más de 3 veces por segundo y que los destellos brillantes están por debajo de los límites definidos por los destellos generales y/o rojos.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/three-flashes-or-below-threshold.html>

### **CC-2.3.2. Tres destellos**

Pertenece al nivel de adecuación AAA y tiene como objetivo que las páginas web no contengan nada que parpadee más de tres veces en un segundo. Cabe destacar que, mientras que el Criterio de Conformidad 2.3.1 permite más de 3 secuencias de parpadeo (de 3 destellos por segundo), este no.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/three-flashes.html>

### **CC-2.3.3. Animación de las Interacciones**

Pertenece al nivel de adecuación AAA y tiene como objetivo asegurar que las animaciones producidas por la interacción del usuario se pueden deshabilitar a no ser que dichas animaciones sean indispensables para ejecutar la funcionalidad o transmitir la información.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/animation-from-interactions.html>

### **CC-2.4.1. Evitar bloques**

Pertenece al nivel de adecuación A y su objetivo es permitir que las personas que navegan de forma secuencial por el contenido mediante lectores de pantalla teclado ir directamente al contenido principal.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/bypass-blocks.html>

### **CC-2.4.2. Título de la página**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar que todas las páginas web contienen títulos representativos que describen el tema o propósito.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/page-titled.html>

### **CC-2.4.3. Orden del foco**

Pertenece al nivel de adecuación A y tiene como objetivo garantizar que, cuando los usuarios naveguen de forma secuencial a través de contenidos, obtengan la información en un orden que es compatible con el sentido del contenido y puede ser operado a través del teclado.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/focus-order.html>

### **CC-2.4.4. Propósito de los enlaces (en su contexto)**

Pertenece al nivel de adecuación A y su objetivo es ayudar a los usuarios a entender el propósito de cada enlace para que puedan decidir si quieren seguirlo.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/link-purpose-in-context.html>

### **CC-2.4.5. Múltiples formas**

Pertenece al nivel de adecuación AA y tiene como objetivo hacer posible que los usuarios localicen el contenido de la manera que mejor se adapte a sus necesidades proporcionando más de un camino para localizar los contenidos, excepto cuando son resultados, pasos intermedios o procesos.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/multiple-ways.html>

### **CC-2.4.6. Encabezados y etiquetas**

Pertenece al nivel de adecuación AA y su objetivo es ayudar a los usuarios a entender qué información está contenida en las páginas y cómo se organiza toda la información.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/headings-and-labels.html>

#### **CC-2.4.7. Visibilidad del foco**

Pertenece al nivel de adecuación AA y su objetivo es ayudar a los usuarios a saber qué elemento, entre los múltiples que hay en la página, tiene el foco del teclado de manera visual y predecible.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/focus-visible.html>

#### **CC-2.4.8. Ubicación**

Pertenece al nivel de adecuación AAA y tiene como objetivo proporcionar una manera para que el usuario pueda orientarse en una web o web app de forma que encuentre la información relacionada a través de migas de pan, sitemaps y otras técnicas adicionales.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/location.html>

#### **CC-2.4.9. Propósito de los enlaces**

Pertenece al nivel de adecuación AAA y tiene como objetivo ayudar a los usuarios a entender el propósito de cada enlace en el contenido, para que puedan decidir si quieren seguirlo o no.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/link-purpose-link-only.html>

#### **CC-2.4.10. Encabezados de sección**

Pertenece al nivel de adecuación AAA y tiene como objetivo proporcionar encabezados en las secciones de una página Web, cuando ésta se encuentra organizada por secciones.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/section-headings.html>

#### **CC-2.5.1. Gestos del puntero**

Pertenece al nivel de adecuación A y tiene como objetivo garantizar que el contenido pueda controlarse con múltiples dispositivos señaladores diferentes, habilidades y tecnologías de asistencia.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/pointer-gestures.html>

#### **CC-2.5.2. Cancelación del puntero**

Pertenece al nivel de adecuación A y su objetivo es asegurar que los usuarios pueden evitar la entrada accidental o errónea del ratón o dispositivo señalador.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/pointer-cancellation.html>

#### **CC-2.5.3. Etiquetas con nombre**

Pertenece al nivel de adecuación A y su objetivo es asegurar que las palabras que etiquetan los componentes sean también las palabras asociadas al componente.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/label-in-name.html>

#### **CC-2.5.4. Actuación por movimiento**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar que todas las funciones que sean activadas por un movimiento físico del dispositivo puedan ser utilizadas a través de otros componentes de la interfaz más comunes o convencionales.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/motion-actuation.html>

#### **CC-2.5.5. Tamaño del objetivo**

Pertenece al nivel de adecuación AAA y tiene como objetivo asegurar que el tamaño de los objetivos sean lo suficientemente grandes como para que los usuarios puedan activarlos fácilmente.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/target-size.html>

#### **CC-2.5.6. Mecanismos de entrada concurrentes**

Pertenece al nivel de adecuación AAA y tiene como objetivo asegurar que las personas puedan utilizar y seleccionar diferentes modos de entrada al interactuar con el contenido.

*Más información en:*

[www.w3.org/WAI/WCAG21/Understanding/concurrent-input-mechanisms.html](https://www.w3.org/WAI/WCAG21/Understanding/concurrent-input-mechanisms.html)

### **COMPRENSIBLE**

#### **CC-3.1.1. Idioma de la página**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar que se está informando del idioma utilizado a los agentes de usuario para que ellos puedan actuar en consecuencia y cambiar reglas de pronunciación, agregar subtítulos, etcétera.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/language-of-page.html>

#### **CC-3.1.2. Idioma de las partes**

Pertenece al nivel de adecuación AA y tiene como objetivo asegurar que los agentes de usuario puedan presentar correctamente el contenido escrito en otros idiomas a excepción de los nombres propios, términos técnicos, palabras del lenguaje indeterminado y expresiones que se han heredado o convertido en nativas como parte del texto inmediatamente circundante.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/language-of-parts.html>

#### **CC-3.1.3. Palabras raras o inusuales**

Pertenece al nivel de adecuación AAA y tiene como objetivo proporcionar una definición de cualquier palabra que se use de una manera inusual o restringida.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/unusual-words.html>

#### **CC-3.1.4. Abreviaturas**

Pertenece al nivel de adecuación AAA y tiene como objetivo proporcionar la forma expandida de las abreviaturas, especialmente si éstas tienen varias interpretaciones posibles.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/abbreviations.html>

#### **CC-3.1.5. Nivel de lectura**

Pertenece al nivel de adecuación AAA y su objetivo es ayudar a las personas con discapacidad de lectura intentando proporcionar una visión más simplificada del texto que requiere un nivel avanzado de conocimiento.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/reading-level.html>

#### **CC-3.1.6. Pronunciación**

Pertenece al nivel de adecuación AAA y tiene como objetivo ayudar a las personas ciegas, visión deficiente y/o con discapacidad de lectura a comprender el contenido en los casos en que, el significado, depende de la pronunciación.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/pronunciation.html>

#### **CC-3.2.1. Al recibir el foco**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar que la navegación por las páginas es predecible y no se producen cambios de contexto, aunque el foco desencadene eventos asociados.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/on-focus.html>

#### **CC-3.2.2. Al recibir una entrada de datos**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar que la introducción de datos o la selección de un control de formulario tiene efectos predecibles.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/on-input.html>

#### **CC-3.2.3. Navegación consistente**

Pertenece al nivel de adecuación AA y tiene como objetivo fomentar que la presentación, el formato y la localización del contenido es coherente para los usuarios que interactúan entre las diferentes páginas del sitio.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/consistent-navigation.html>

#### **CC-3.2.4. Identificación consistente**

Pertenece al nivel de adecuación AA y tiene como objetivo asegurar que los componentes que tienen la misma funcionalidad, dentro de un conjunto de páginas, se identifican de manera consistente.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/consistent-identification.html>

#### **CC-3.2.5. Cambio a petición**

Pertenece al nivel de adecuación AAA y tiene como objetivo evitar la posible confusión que puede ser causada por los cambios de contexto, tales como las ventanas de diálogo emergentes o popups, el envío automático de formularios tras seleccionar un elemento de una lista, etc.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/change-on-request.html>

#### **CC-3.3.1. Identificación de errores**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar que los usuarios son conscientes de que se ha producido un error informándoles con mensajes claros para que puedan saber lo que sucede y dónde.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/error-identification.html>

#### **CC-3.3.2. Etiquetas o instrucciones**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar que se proporcionan etiquetas o instrucciones cuando se requiere la intervención del usuario.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/labels-or-instructions.html>

#### **CC-3.3.3. Sugerencias de error**

Pertenece al nivel de adecuación AA y tiene como objetivo asegurar que los usuarios reciben unas sugerencias apropiadas o adecuadas para poder realizar la corrección de los errores, si es posible.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/error-suggestion.html>

#### **CC-3.3.4. Prevención de errores (legales, financieros, de datos)**

Pertenece al nivel de adecuación AA y su objetivo es ayudar a los usuarios con discapacidad a evitar graves consecuencias como resultado de un error al realizar una acción que no se puede revertir.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/error-prevention-legal-financial-data.html>

#### **CC-3.3.5. Ayuda**

Pertenece al nivel de adecuación AAA y tiene como objetivo evitar que los usuarios cometan errores a través de “ayudas sensibles al contexto”, asistentes de ayuda, corrección ortográfica y sugerencias para la introducción del texto o instrucciones “textuales”.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/help.html>

### **CC-3.3.6. Prevención de errores (todos)**

Pertenece al nivel de adecuación AAA y tiene como objetivo ayudar a los usuarios con discapacidad a evitar las consecuencias que puedan derivarse al cometer un error en el proceso de envío de datos de un formulario.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/error-prevention-all.html>

## **ROBUSTO**

### **CC-4.1.1. Análisis**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar que los agentes de usuario, incluyendo las ayudas técnicas, pueden interpretar y analizar con precisión el contenido.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/parsing.html>

### **CC-4.1.2. Nombre, función, valor**

Pertenece al nivel de adecuación A y tiene como objetivo asegurar que el nombre y función de todos los componentes de la interfaz de usuario (incluidos los elementos de formulario, enlaces y componentes generados por scripts) puedan determinarse a través de código.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/name-role-value.html>

### **CC-4.1.3. Mensajes de estado**

Pertenece al nivel de adecuación AA y tiene como objetivo asegurar que los mensajes de estado implementados por los lenguajes de marcado como HTML puedan ser determinados a través de propiedades de modo o roles.

*Más información en:*

<https://www.w3.org/WAI/WCAG21/Understanding/status-messages.html>

## **Proceso de la conformidad en accesibilidad web**

El proceso de la conformidad en accesibilidad web podríamos decir que se conforma de dos pasos.

### **EXAMEN PRELIMINAR**

El primero de ellos es realizar un examen preliminar para verificar el grado de cumplimiento actual que tiene el sitio web. Este examen no tiene por qué realizarse en todas las páginas, puede hacer sólo en las más representativas o visitadas.

Para conocer un poco el estado actual, se debe comprobar lo siguiente:

- Si el tamaño de fuente influye en la legibilidad de las páginas, es decir, si al aumentar el tamaño de los textos se pierde la estructuración y compresión de los contenidos.
- Comprobar que los contenidos textuales, como las imágenes, presentan una alternativa textual que lo describa.
- Verificar que no aparece la barra de desplazamiento horizontal en ninguna de las resoluciones típicas o populares, exceptuando si el diseño de la web es en horizontal. Es este último caso, se debe comprobar que no hay desplazamiento vertical en ninguna de las resoluciones.

- Comprobar que se puede realizar el manejo y navegación de todos los botones y enlaces de la página a través de teclado mediante el uso del tabulador, atajos de teclado, barra espaciadora, etcétera.
- Si la página tiene video o audio, verificar que dispone de transcripciones textuales y/o subtítulos.
- Examinar las páginas con algún lector de pantalla y navegador de sólo texto para verificar que los contenidos están presentados con legibilidad y en el orden correcto.
- Verificar la validez del código a través de alguna herramienta validadora y alguna herramienta de evaluación automática.

Al final de este análisis, se puede realizar un informe que describa el estado actual. Se debe tratar de exponer una idea global que muestre todos los aspectos que se han encontrado, es decir, tanto lo que sí se cumple, como lo que no.

Cuando se describa cada punto se deberá especificar su tipo, severidad, nivel de conformidad, criterio de conformidad y el método por el cual se encontró. Además, se pueden incluir comentarios, recomendaciones, sugerencias e, incluso, la realización de un análisis más exhaustivo.

## **IMPLANTACIÓN, DESARROLLO Y MANTENIMIENTO**

Una vez que ya sabemos el estado del sistema o interfaz y antes de empezar la implantación o proceso de mejora de accesibilidad web, se debe establecer el nivel de adecuación que se desea aplicar, esto es A, AA o AAA y un calendario de revisión.

Seguidamente se debe seleccionar a una persona que cumpla con el rol de experto en accesibilidad para que realice los seguimientos oportunos.

Después, durante el desarrollo y/o mantenimiento, se debe proporcionar a los desarrolladores y diseñadores toda la información necesaria sobre posibles métodos de evaluación para que puedan ir avanzando sin cometer demasiados errores y no tengan que trabajar más de lo debido.

De manera adicional, de vez en cuando, se pueden realizar verificaciones con herramientas de evaluación automática para ir comprobando que todo lleva un flujo y enfoque adecuado.

## **HERRAMIENTAS PARA LA VALIDACIÓN DE LA ACCESIBILIDAD WEB**

### **Basadas en navegador**

Existen muchos complementos para ayudar a validar la accesibilidad de los sitios web, no obstante, en muchas ocasiones estos complementos no son compatibles con todos los navegadores y hay que recurrir a componentes específicos de cada uno de ellos.

Entre los más conocidos podemos encontrar:

#### **Extensiones / AddOns para Chrome**

- **WAVE Evaluation Tool** permite realizar la evaluación de una página web y proporciona comentarios visuales sobre la accesibilidad de su contenido web mediante la inyección de iconos e indicadores.
- **WCAG Contrast Checker** evalúa el contraste en todos los elementos de la página considerando su estilo calculado para las propiedades CSS de color y color de fondo. En el caso de que estos colores se definan con valores RGBA, también considera la opacidad para deducir el color real que finalmente se muestra.
- **Accessibility monitor** es una herramienta que audita las páginas para detectar errores de accesibilidad a medida que se van utilizando.

- **Axe Web Accessibility Testing** permite verificar la accesibilidad para WCAG 2 y Sección 508, aunque sólo prueba los problemas de accesibilidad que pueden detectarse con precisión a través de la automatización, y solo prueba los componentes que realmente existen en la página o aplicación que está probando.
- **Total validator** permite realizar múltiples validaciones, desde HTML y CSS, hasta evaluaciones de accesibilidad web WCAG2 o US-508. Además, tiene corrector de enlaces rotos y corrector ortográfico en varios idiomas.
- **Color Contrast Analyzer** permite analizar una página web o una parte de una página web para cumplir con los requisitos de contraste de color de WCAG 2.0.

## **Extensiones / AddOns para Chrome**

- **WAVE Accessibility Extension** permite realizar la evaluación de una página web y proporciona comentarios visuales sobre la accesibilidad de su contenido web mediante la inyección de iconos e indicadores.
- **WCAG Contrast Checker** evalúa el contraste en todos los elementos de la página considerando su estilo calculado para las propiedades CSS de color y color de fondo. En el caso de que estos colores se definan con valores RGBA, también considera la opacidad para deducir el color real que finalmente se muestra.
- **Axe Developer Tools** permite verificar la accesibilidad para WCAG 2 y Sección 508, aunque sólo prueba los problemas de accesibilidad que pueden detectarse con precisión a través de la automatización, y solo prueba los componentes que realmente existen en la página o aplicación que está probando.
- **Visual Aria** es una herramienta que permite a los ingenieros, evaluadores, educadores y estudiantes observar físicamente el uso de ARIA dentro de las tecnologías web, incluidas las funciones estructurales, de región en vivo y de widgets de ARIA 1.1, la anidación adecuada y la gestión del enfoque.
- **Tota11y Accessibility Toolkit** es una herramienta que muestra el rendimiento de las páginas con tecnologías de asistencia y permite observar problemas de tipo encabezado, contraste de color, textos de enlaces, etiquetas o alternativas de texto en imágenes.
- **Totally Automated Accessibility Scanner** es una herramienta que detecta e informa automáticamente las infracciones de accesibilidad comunes de las páginas.
- **Firebug** es un conjunto de herramientas con las que se puede revisar el CSS, HTML y JavaScript, comprobar la velocidad de carga, consultar la estructura del DOM, monitorizar y depurar el código fuente, entre otras funcionalidades.
- **HTML Validator** es una herramienta que agrega un monitor que valida el HTML y muestra el número de errores encontrados en forma de ícono.

## **Extensiones / AddOns para Internet Explorer**

- **Web Accessibility Toolbar** es una herramienta que ayuda a la evaluación de las páginas de forma manual para comprobar que se está cumpliendo con las Pautas de Accesibilidad para el Contenido Web 2.0.

## **Extensiones / AddOns para Safari**

- **a11yTools - Accessibility Web** es una colección de herramientas para verificar la accesibilidad web HTML de forma rápida y fácil. Además de ejecutar la prueba de accesibilidad permite tomar una captura de pantalla que muestra el error "a11y".

## **Mediante aplicaciones de escritorio**

### **HERRAMIENTAS DE ACCESIBILIDAD WEB DE WINDOWS**

Windows dispone de múltiples herramientas que pueden ayudar a las personas con discapacidad y mejorar un poco la accesibilidad web. En esta sección sólo presentaremos las “más conocidas”, aunque hay bastantes más, todas ellas explicadas en la URL <https://www.microsoft.com/es-es/accessibility/windows>.

#### **Magnificador pantalla de Windows**

Windows dispone de un magnificador de pantalla denominado LUPA. Esta herramienta puede aumentar distintas partes de la pantalla y dispone de tres modos diferentes (a pantalla completa, modo lente y modo acoplado).

Para activarla sólo hay que escribir lupa en el menú de inicio de Windows. Si se desea salir o del software de ampliación de pantalla, sólo se debe pulsar la combinación de  + Esc.

#### **Teclado en pantalla de Windows**

El teclado visual de Windows es una aplicación independiente que contiene todas las teclas estándar que, un teclado físico puede utilizar.

Sólo se necesita escribir “teclado en pantalla” en el menú de inicio y mostrará la aplicación para su utilización. Esta aplicación permite, además, su manejo a través de un ratón o dispositivo señalador.

#### **Narrador**

Lee en voz alta el texto de la pantalla y describe algunos eventos y mensajes de error que se producen mientras se utiliza el sistema.

Para poder utilizarlo se debe configurar una voz y realizar algunos ajustes adicionales.

#### **Reconocimiento de voz**

Permite interactuar con el sistema utilizando únicamente la voz.

Para poder utilizarlo se deben realizar algunos ajustes como seleccionar el tipo de micrófono que se va a usar y pruebas de voz.

#### **Métodos abreviados de teclado**

Los métodos abreviados o atajos de teclado son combinaciones de dos o más teclas del teclado que, cuando se presionan, realizan una tarea que normalmente requiere un ratón u otro dispositivo señalador.

En Internet existen gran cantidad de fuentes dónde se describen, uno a uno, todos los atajos más recurrentes e, incluso, algunos desconocidos bastante útiles.

### **PHOTOSENSITIVE EPILEPSY ANALYSIS TOOL / PEAT**

**URL:** <https://trace.umd.edu/peat>



Photosensitive Epilepsy Analysis Tool, también conocida como PEAT, es una herramienta que permite determinar si las animaciones o videos que se presentan o descargan en los sitios web y/o aplicaciones pueden causar convulsiones

fotosensibles, es decir, si contiene parpadeos o transiciones rápidas entre los colores de fondo claros y oscuros.

## **JOB ACCESS WITH SPEECH / JAWS**

**URL:** <https://www.freedomscientific.com/products/software/jaws/>



Job Access With Speech, más conocido como JAWS, es un lector de pantalla para Windows que permite a todas las personas con discapacidad visual total o parcial leer la pantalla con una salida de texto a voz o a través de una pantalla de Braille actualizable.

## **Mediante servicios web externos**

### **MARKUP VALIDATION SERVICE**

**URL:** <https://validator.w3.org/>



Markup Validation Service es servicio gratuito creado por la W3C que permite comprobar la validez del marcado en códigos HTML, XHTML, SMIL y MathML, entre otros, en contextos de Web Semántica, Accesibilidad y Usabilidad Web.

### **CSS VALIDATION SERVICE**

**URL:** <http://jigsaw.w3.org/css-validator/>



CSS Validation Service es un software creado por la W3C y pensado para diseñadores y desarrolladores web que permite revisar las hojas de estilo en cascada (CSS) y los documentos XHTML con hojas de estilo. Puede utilizarse mediante su servicio gratuito vía web, o puede descargarse para ser usado bien como un programa Java, o como un servlet Java en un servidor Web.

## **LINK CHECKER**

**URL:** <https://validator.w3.org/checklink>



Link Checker es una herramienta web que busca problemas en vínculos, anclajes y otros objetos referenciados en una página web, hojas de estilo CSS o, recursivamente, en todo el sitio Web completo.

## **ACCESSIBILITY CHECKER**

**URL:** <https://achecker.ca/checker/>



Accessibility Checker (AChecker) es una herramienta web que permite detectar e informar de las infracciones de accesibilidad en los contenidos, al mismo tiempo que proporciona recomendaciones sobre cómo repararlas. No obstante, sólo verifica los problemas de accesibilidad para la normativa WCAG 2.0.

## **TAW**

**URL:** <https://www.tawdis.net/>



TAW es una herramienta web que permite evaluar de forma automática la accesibilidad de una página web, utilizando algunas de las técnicas recomendadas por las WCAG 2.0. Además, permite la intervención del evaluador/a para confirmar o descartar de manera manual.

## **Evolución de la accesibilidad. Nuevas tendencias**

Si hay una tendencia web que está ganando terreno a pasos agigantados, es la accesibilidad web, porque además de permitir el acceso a los contenidos a un número mayor de usuarios, vuelve los sistemas e interfaces más usables. De hecho, el concepto de accesibilidad web ya no podría considerarse una tendencia, ahora se podría decir que es una realidad., aunque sólo sea por la responsabilidad social corporativa.

En lo referente a cómo evoluciona la accesibilidad, se podrían añadir muchos comentarios al respecto, porque está en constante desarrollo. Por ejemplo, si bien la accesibilidad busca, entre otras cosas, lo simple, los diseños minimalistas Flat Design y Material Design son buenas pruebas de ello.

Si la accesibilidad web busca que todas las personas puedan acceder a todos los contenidos de la red, buscar formas de conseguir que perciban lo mismo en todos los casos puede ser una gran aventura. Según avanzan las cosas, la accesibilidad

irá evolucionando más a la robótica y a la inteligencia artificial porque son dos elementos que pueden ayudar, si se utilizan bien, a gran parte de los discapacitados, exceptuando casos muy concretos.

En estos dos elementos, también entrará en juego una mejora de la interacción persona ordenador para se vuelva más coherente y accesible y, la semántica, para que los contenidos y su estructura sean mucho más legibles y compresibles.

## PRÁCTICA 13: VERIFICANDO Y MEJORANDO LA ACCESIBILIDAD WEB DE UNIVERSES

### Código del ejemplo

<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-13>



### Objetivo de la práctica

Verificar y solucionar los posibles problemas de accesibilidad web de UniversES.

### Resolución

Si, en el capítulo anterior, vimos cómo evaluar y mejorar la Usabilidad Web de nuestra página de inicio, ahora es el momento de evaluar y mejorar la Accesibilidad Web.

Bien, podríamos hacer una revisión manualmente y, probablemente, puede que incluso fuese más efectiva, no obstante, para realizar una revisión manual de la Accesibilidad Web de un sitio se requiere tiempo, atención y paciencia. Por ello, para que este proceso sea algo más ágil, nos valdremos de herramientas externas que nos ayuden a encontrar, al menos, los errores más evidentes.

Una de estas herramientas bien podría ser la denominada AXE, disponible como extensión de los navegadores Chrome y Firefox. Para instalarla, sólo hace falta ingresar en el site de extensiones o complementos, buscar AXE y presionar en el botón de instalar.

Una vez instalado el complemento del navegador, accedemos a la consola del navegador presionando F12 (o la combinación de teclas que proceda), seleccionamos la pestaña con su nombre y presionamos ANALIZE.

Tras el análisis, debería aparecer una pantalla similar la siguiente:

A screenshot of the AXE browser extension interface. The main window shows a list of accessibility issues found on the UniversES website. There are 18 issues listed, including: 'Document must not have more than one source (favicon)', 'Elements must have unique IDs', 'Elements must have sufficient color contrast' (with 14 instances), 'Links with the same name must have a different href', 'Elements must have sufficient color contrast' (with 3 instances), and 'Links with the same name must have a different href'. Below the list, there's a detailed description of the first issue, a 'Bar chart' section showing the distribution of issues by severity, and a summary at the bottom stating 'We are not sure this is an issue, because:'. The overall interface is dark-themed with white text and icons.

Aunque en la imagen no se percibe con claridad, uno de los errores encontrados es que, el elemento HEADER > .LOGOTIPO no posee el contraste adecuado entre texto y fondo.

Element location

```
header > .logotipo
```

Element source

```
<span class="logotipo">UniversES</span>
```

Este es un buen ejemplo para nuestra práctica porque no todos los errores que encuentran las herramientas de evaluación de accesibilidad son errores realmente.

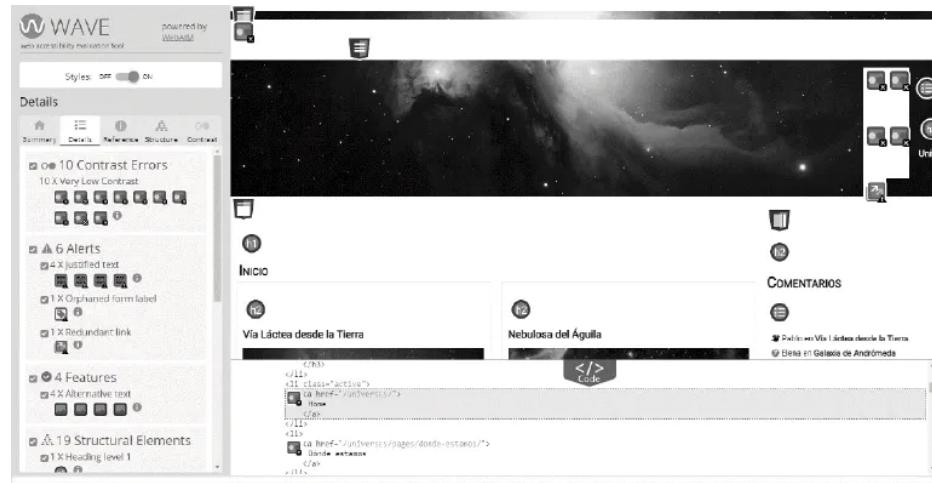
De hecho, este error es porque el elemento que representa el logotipo es un texto en vez de una imagen y, aunque el texto es blanco, el fondo del elemento es transparente, por lo que el sistema advierte que no hay un contraste asegurado. No obstante, como podemos apreciar en la captura inicial, el texto sí que posee un contraste suficiente puesto que el fondo de la imagen es negro.

Uno podría pensar que la herramienta de AXE es mala, pero no, sólo está desactualizada. Por este motivo, lo que haremos es cambiar de herramienta antes de continuar con el proceso de evaluación de Accesibilidad Web.

La siguiente herramienta que puede ayudarnos a mejorar la accesibilidad es la denominada como WAVE, disponible como extensión de los navegadores Chrome y Firefox. Para instalarla, al igual que la que acabamos de utilizar, sólo hace falta ingresar en el site de extensiones o complementos, buscar AXE y presionar en el botón de instalar.

Una vez instalado el complemento del navegador, sólo hay que pulsar en el icono con forma de onda o uve doble redondeada que representa a la extensión, normalmente ubicado al lado de la barra de direcciones del navegador.

Tras el análisis, debería aparecer una pantalla similar la siguiente:



## MEJORANDO EL CONTRASTE

Como se puede apreciar en la imagen, lo primero que nos indican es que se han encontrado 10 errores de contraste. Este error, lejos de ser un problema difícil de solucionar, sí que puede ser un problema grave, puesto que los usuarios pueden presentar una discapacidad o incapacidad visual parcial y, un bajo contraste, puede convertirse en un impedimento en la legibilidad.

No obstante, aunque en la información de referencia que presenta la herramienta dice que se aumente el contraste entre el texto y el fondo, este aumento, no tiene por qué hacerse en el propio elemento objeto del análisis. Es decir, el contraste puede venir determinado por el color del texto y el color de fondo de uno de sus ancestros.

Por ello, para solucionar este problema, lo que haremos es modificar el color de fondo y la altura del elemento referenciado por la regla de CSS BODY > HEADER definida dentro de nuestra hoja de estilos (archivo STYLES.CSS). El fondo será

#000000 y la altura 44 píxeles.

Este cambio, provocará que se solucionen ocho, de los diez errores encontrados, puesto que todos ellos están dentro del elemento HEADER y un fondo de color negro cumple el requisito de tener un contraste igual o superior a 7 sobre 1.

Para los otros dos, la solución también pasa por establecer un color de fondo negro, sin embargo, este cambio puede ser una trampa porque, aunque el fondo de la imagen sea negro, dicho color de fondo nunca será visible ya que la imagen carece de transparencias y está ajustada al tamaño del contenedor.

Pensemos que podría darse la circunstancia de que los textos implicados se posicionasen exactamente en la zona más iluminada de la imagen y eso provocaría que los textos se volviesen ilegibles. Un ejemplo de esta situación la podemos ver a continuación:



Por tanto, la solución real a este problema pasa por establecer un color de fondo a los textos objeto del análisis. Es decir, modificar las reglas de estilo de los elementos que tienen los textos posicionándolos de manera absoluta en el centro de la pantalla y estableciéndoles un color de fondo negro con un borde fino y claro y un ancho de 360 píxeles.

Ahora sí. Con esta solución aseguramos que los textos poseen un contraste suficiente y que los usuarios lo podrán leer sin problemas.

## MEJORANDO LA LEGIBILIDAD

Un problema que parece estar muy presente en Internet es que los textos estén justificados. La razón de justificarlos es porque muchos autores piensan que una buena “imagen” provoca una mejor experiencia de usuario y, como los textos en este formato suelen generar una mayor sensación de orden, creen que el usuario obtendrá una mayor satisfacción.

Nada más lejos. No olvidemos que, un buen diseño, es aquel que consigue dar una buena experiencia de usuario a todas las personas que la utilizan por igual, sea cual sea la limitación, discapacidad o incapacidad que posea.

Por ello, si hay personas que puedan presentar dificultades en la lectura, lo que debemos hacer es cambiar nuestro diseño para que todas ellas perciban la misma experiencia de usuario.

La solución, entonces, pasa por establecer la alineación a la izquierda para todos los textos de la página.

## SOLUCIÓN A LOS ELEMENTOS HUÉRFANOS

Otro problema muy frecuente en los diseños es la utilización de elementos LABEL para presentar textos. Esto no sería un problema si no fuese porque este elemento está muy vinculado a los formularios y, las herramientas de accesibilidad, pueden asociarlos a problemas de pérdida de asociación.

Recordemos que, para que un campo de entrada sea accesible, o bien el elemento INPUT está asociado con un elemento LABEL a través del atributo FOR, o bien el elemento INPUT está dentro del propio LABEL.

Por esta razón, si durante el proceso del análisis la herramienta de validación encuentra un elemento LABEL sin el atributo FOR, y sin un elemento INPUT en su interior, puede deducir que está huérfano y mostrar un mensaje de error.

Para solucionarlo, lo que se puede hacer es, simplemente, cambiar el elemento por otro de uso general. En nuestro caso, habrá que cambiar el elemento LABEL de nuestro componente LOADER por un elemento SPAN, tanto en el HTML, como en el CSS.

## SOLUCIÓN A LOS ENLACES REDUNDANTES

Este problema no suele darse con frecuencia, sin embargo, puede ser causa de varios quebraderos de cabeza.

Si comprobamos el código que nos muestra la consola del navegador, o la herramienta WAVE, veremos que, en el menú de navegación, hay dos enlaces con el mismo vínculo.

Ambos enlaces están anulados a través de la instrucción JAVASCRIPT:VOID(0), que se supone que evita el vínculo y corta la navegación, sin embargo, el sistema nos comunica que hay un problema de enlace redundante.

Para solucionar este problema, lo que haremos es establecer, en cada atributo HREF, un texto o una función diferente. En nuestro caso, estableceremos llamadas a las funciones de JavaScript FONTSIZEINCREASE y FONTSIZEDECREASE desde un elemento A, situado debajo del elemento UL de nuestro elemento NAV.

Estas funciones, básicamente, lo que harán es recuperar el tamaño de la letra asignada al elemento BODY de la página e incrementarla o restarle uno en función del enlace pulsado.

Sin embargo, todavía no hemos terminado puesto que, ahora, nos ha aparecido otro problema. Al añadir dos nuevos botones, necesitamos asegurar, todos los puntos anteriormente explicados, es decir, el contraste, legibilidad, etcétera. Por ello, se definirá una nueva regla CSS denominada BODY > HEADER > NAV > A que establecerá los enlaces a la derecha del elemento NAV, con color de texto blanco, entre otras opciones.

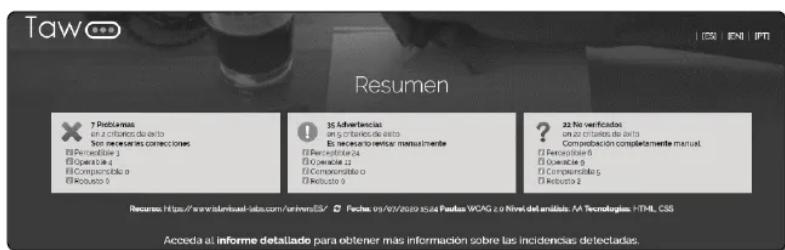
## VERIFICANDO LOS CAMBIOS

Si ahora, recargamos la página y volvemos a presionar el botón asociado al componente WAVE, se nos mostrará un resumen donde podremos comprobar el número de elementos estructurales, el número de características verificadas, el número de etiquetas ARIA encontradas y, lo más importante, que no hay errores generales, ni de contraste, ni de alertas. Además, se nos mostrará un mensaje de felicitación y otro recordándonos que el test manual es requerido para asegurar que la accesibilidad web está cubierta.

## OTRAS CONSIDERACIONES

Una vez que hemos terminado de verificar la accesibilidad a nivel local, si lo deseamos, podemos adquirir otro informe algo más detallado. No obstante, para conseguir este informe, lo único que necesitamos es tener nuestra web accesible desde Internet a través de algún servicio de hosting.

Si esto sucede, el siguiente paso puede ser irnos a TAWDIS.NET e introducir la URL para que el sistema analice la web de forma remota. Tras haber introducido la url de nuestra página, el sistema nos debería mostrar un resumen en pantalla con la posibilidad de enviar el informe detallado a una dirección de correo.



En el informe que se recibe a través del servicio de correo, se nos muestran, una a una, todas las incidencias ordenadas y categorizadas por pauta de accesibilidad y Criterio de Conformidad. Además, se adjunta el código fuente con cada una de las líneas en donde se nos indica el error que se produjo.

The screenshot shows a detailed accessibility audit report from the TAW tool. At the top, there's a message encouraging users to support the project. Below that, the 'Información del análisis' section provides basic details like the URL, date, and WCAG level. The main content area is titled 'Perceivable' and contains a sub-section for '1.4.1-Contenido no textual'. This section lists two types of non-textual content: 'Imágenes' and 'Navegación'. For 'Imágenes', it says they require descriptive text. For 'Navegación', it says they have consecutive text and image links. A table summarizes the results for each type, showing counts of 4 and 3 respectively, along with various icons indicating technical details.

Tras analizar este informe, lo que vemos es que, entre otras cosas, se están detallando errores claros de maquetación. Por ejemplo, uno de los primeros errores que nos muestra es que, según el Criterio de Conformidad de Contenido no textual “[hay enlaces consecutivos de texto e imagen al mismo recurso](#)”. Esto es, principalmente, porque todos nuestros enlaces tienen el símbolo de almohadilla establecido en los atributos HREF. En cuanto estos enlaces apunten a direcciones o recursos reales, estos errores desaparecerán.

Y lo mismo pasa con el error que está asociado al Criterio de Conformidad “[2.4.4 Propósito de los enlaces \(en contexto\)](#)”. En cuanto los enlaces tengan una definición adecuada que haga que puedan diferenciarse de manera automática, este error desaparecerá.

Por último, sólo hay que destacar que, esta herramienta automática, no cubre toda la accesibilidad web, pero es uno de los mejores informes que se pueden encontrar. Recordemos que es importante verificar manualmente la accesibilidad web, como comentamos antes porque, de no ser así, no podremos asegurar que nuestra página es totalmente accesible.

# 13

## DISEÑO DE EMAILS BASADOS EN HTML

### USO DE JAVASCRIPT

Cuando se trata de crear emails, se deben tener en cuenta ciertas directrices puesto que los clientes de correo electrónico presentan muy diversas limitaciones.

En la dirección <https://www.caniemail.com> es posible averiguar todas y cada una de las limitaciones que presentan los elementos y atributos de HTML y las propiedades de CSS.

Para empezar, el uso de JavaScript y todas sus variaciones como jQuery, Vue JS, ReactJS, NodeJS, Angular, etcétera, pueden ser muy potentes y todo lo que se quiera, sin embargo, no pueden ser utilizados para este objetivo.

La razón de que no se pueda incrustar código alguno escrito en estos lenguajes o frameworks es porque pueden generar agujeros de seguridad y permitir el acceso a zonas sensibles a través de las propiedades del agente de usuario.

No obstante, sí que se pueden insertar funcionalidades adicionales se puede recurrir a otros métodos. Por ejemplo, si lo que se desea es conocer el ratio de visualizaciones o cuántos usuarios al accedido a una landing page a partir de ese email, se puede recurrir a lo que se denominan píxeles de seguimiento.

Para entendernos, los píxeles de seguimiento son una imagen de 1x1 píxeles que produce una descarga cuando un usuario visita una página web o abre un email. Suelen implementarse a través de servicios de terceros y son utilizados para monitorizar acciones específicas de los usuarios como pueda ser la obtención de información estadística sobre analítica web o marketing online.

## TIPO DE DOCUMENTO A UTILIZAR

Para diseñar un email lo más compatible posible, se debe encontrar un DOCTYPE adecuado. No obstante, encontrar un DOCTYPE que funcione en todos los clientes de correo resulta ser una misión imposible porque, o bien usan el suyo propio, o bien lo eliminan.

Según algunos estudios, hoy en día, lo más compatible es utilizar el DOCTYPE HTML5 puesto que los clientes de correo electrónico de Apple Mail, Outlook, Gmail, Yahoo, Outlook 365 y Orange lo soportan. Sin embargo, si utilizamos alguna versión antigua de estos clientes de correo, lo mejor es recurrir a una estructura de documento de tipo XHTML1.0 TRANSICIONAL.

En lo que casi todos están de acuerdo es en utilizar la etiqueta HTML con declaración de idioma y soporte VML. Esta declaración garantizará que el correo electrónico se muestre en todos los clientes de Outlook, además de permitir la utilización de atributos y propiedades que, de otro modo, serían incompatibles, como la propiedad BACKGROUND de CSS.

A continuación, se muestra una tabla con los tipos de documento que pueden manejar los clientes de correo:

Nombre	dESKTOP wEBMAILS	mOBILE	aNDROID	ios
		wEBMAILS	aPPS	aPPS
AOL	NO USA	HTML5	X	X
Apple mail	SE RESPETA	X	X	SE RESPETA
Gmail	HTML5	HTML5	HTML5	HTML5
Inbox Gmail	HTML5	HTML5	HTML5	HTML5
Outlook	HTML5	HTML5	SE RESPETA	SE RESPETA
Orange	HTML5	NO USA	SE RESPETA	NO USA
Yahoo	HTML5	HTML5	SE RESPETA	HTML5

Si vemos la tabla, el término “SE RESPETA” indica que se respetará el tipo de documento que venga definido en el email, tanto si es XHTML1.0 transicional, como si es HTML5. El término “HTML5” indica que se eliminará el DOCTYPE que tenga establecido el email en su definición y lo convertirá a un tipo de documento de HTML5. El término “NO USA” indica que no utiliza DOCTYPE alguno y, el término “X”, indica que no dispone de esa posibilidad o versión.

## PECULIARIDADES DE LOS CLIENTES DE CORREO

A continuación, se comentan algunas de las peculiaridades más notables que presentan los clientes de correo más conocidos.

- **AOL:** Permite los estilos en línea y la adhesión de reglas, sin embargo, no es del todo compatible con el uso de LINK, no admite la personalización de fuentes y tiene algunos problemas con la definición de imágenes de fondo a través de CSS.
- **APPLE MAIL:** Permite los estilos en línea y la personalización de fuentes, pero tiene algunos problemas con HTML5.
- **GMAIL:** Permite el diseño responsive y los estilos en línea, pero renombra las clases CSS añadiendo un prefijo, por lo que es complicado proporcionar estilos comunes en algunas situaciones, como el cambio de modo oscuro.
- **OUTLOOK:** Permite los estilos en línea y elimina las clases CSS, pero añade atributos personalizados como DATA-OGSB y DATA-OGSC, los cuales pueden utilizarse para afinar en los diseños. Normalmente, el atributo DATA-OGSB lleva asociado un color de fondo y el atributo DATA-OGSC lleva asociado un color de texto.
- **YAHOO:** Permite los estilos en línea y la adhesión de reglas, sin embargo, para poder referenciarlas se requiere que haga referencia a la clase YAHOO que añade al elemento BODY.

## EL MODO OSCURO

Hasta no hace tanto, todos los clientes de correo estaban diseñados con tonos claros, si no blancos. Aunque no se le daba toda la importancia que debía, estos diseños generaban fatiga visual y un mayor gasto energético.

Entonces, llegó el modo oscuro, que ahorraba batería en los dispositivos móviles y mejoraba el cansancio de los ojos, pero aumentaron los problemas en lo que a diseño y marketing se refiere.

En principio, el modo oscuro, no es más que una inversión de los colores que tiene definido el email. No obstante, no todos los clientes de correo admiten el modo oscuro y no todos los profesionales están dispuestos a aceptar los cambios que produce el cambio de modo. Por ello, HTML y CSS ofrecen una manera de “controlar” el modo en el que se encuentra el cliente de correo.

Para saber si el dispositivo tiene habilitado el modo oscuro o, por el contrario, el modo claro, se puede añadir en la sección de cabecera el siguiente código:

```
<meta name="color-scheme" content="light dark">
<meta name="supported-color-schemes" content="light dark">
<style type="text/css">
:root {
Color-scheme: light dark;
supported-color-schemes:light dark;
}
</stlye>
```

Una vez que hayamos declarado estos metadatos en nuestro documento, ya podremos manipular el modo a través de una simple consulta de medios.

```
@media (prefiere-color-esquema: oscuro) {
body { background-color: #121212! important; }
/* Este y otros posibles estilos pueden definirse aquí*/
}
```

## SECCIÓN DE CABECERA

En la sección de cabecera se deberá especificar el tipo de codificación que se utiliza, la vista de compatibilidad con la última versión de Internet Explorer y el área visible de la ventana para dispositivos móviles, entre otras cosas. Y

como cabía esperar, en esta sección de cabecera, también se podrán especificar fuentes vectoriales a través de servicios de terceros como un CDN (Content Delivery Network), consultas de medios, reglas de estilo y otros efectos como las animaciones CSS que se vayan a utilizar en todo el documento.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"https://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="es"
xmlns="http://www.w3.org/1999/xhtml"
xmlns:v="urn:schemas-microsoft-com:vml"
xmlns:o="urn:schemas-microsoft-com:office:office">
<head>
<!--[if gte mso 9 | (IE)]>
<xml>
<o:OfficeDocumentSettings>
<o:AllowPNG/>
<o:PixelsPerInch>96</o:PixelsPerInch>
</o:OfficeDocumentSettings>
</xml>
<![endif]-->
<title>Ejemplo de email</title>
<meta http-equiv="Content-Type"
content="text/html; charset=UTF-8" />
<meta http-equiv="X-UA-Compatible"
content="IE=edge" />
<meta name="viewport"
content="width=device-width, initial-scale=1.0 " />
<!--[if !mso]><!-->
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,600,700">
<!--<![endif]-->
<style>
<!-- CSS code (if any) --->
</style>
</head>
</html>
```

Si nos fijamos en el código declarado justo debajo de la etiqueta HEAD, veremos que se ha definido una instrucción condicional que lo que hace es averiguar si el correo está siendo mostrado por un Outlook versión 9 o superior, es decir, desde Outlook 2000 en adelante, o por un Outlook 2000-2003 que usa Internet Explorer.

Su código interior, es algo complicado de entender, pero, básicamente, permite forzar los DPI a un nivel inferior para los clientes de Outlook. Su objetivo, ayudar a prevenir el escalado de la imagen y evitar problemas de representación.

## SECCIÓN DEL CUERPO

En la sección del cuerpo, se deberán definir la estructura y los contenidos del email, sin embargo, debemos tener en cuenta algunos detalles:

- No es posible utilizar los elementos HTML5. De hecho, si utilizamos elementos como SECTION, ARTICLE, ASIDE, etcétera, lo más probable es que se produzcan problemas de formato, de presentación, o algo peor.
- Tampoco está permitido el uso de IFRAME, formularios ni elementos basados en cajas o capas para estructurar las diferentes partes o secciones, por lo que se deben evitar los elementos como DIV o SPAN.
- Los correos electrónicos no deben superar los 600 píxeles de ancho. Esto sucede porque todavía hay muchos clientes de correo como Microsoft Outlook que trabajan con resoluciones pequeñas e, incluso, algunos de estos

clientes de correo tienen 10 años o más. No obstante, esto no tardará en cambiar puesto que los dispositivos actuales cuentan con una mayor resolución y será cuestión de tiempo que los usuarios terminen por adaptarse.

- En la etiqueta BODY se deben resetear los márgenes internos y externos y establecer el color de fondo del contenido.
- Por norma general, después de la etiqueta BODY, se suele añadir un elemento CENTER que actúa como portador de propiedades globales, ya que la etiqueta BODY puede no ser correctamente interpretada por los clientes de correo. Además, ayuda a que, todo el contenido, esté centrado.
- Seguidamente, y como primer descendiente directo del elemento CENTER, se puede añadir un DIV que tenga definida la clase WEBKIT puesto que los clientes de Apple Mail 6 o inferiores y Outlook 2011 lo requieren para su correcta visualización.
- Se debe evitar todo uso de elementos BR, HR y del atributo HEIGHT. Esto se debe, fundamentalmente, a que los clientes de correo establecen su propia altura de línea y sobrescriben los estilos asociados que puedan tener.
- La estructura principal de un correo debe delimitarse a través de estructuras TABLE, es decir, que cada sección diferenciada del email deberá ser una celda definida a través de elementos TD. Para la personalización de esta tabla, lo mejor es recurrir a las propiedades HTML siempre que se pueda. Es decir, se debe recurrir a atributos como BGCOLOR, CELLPADDING, ALIGN, VALIGN, y WIDTH para personalizar las celdas y tablas.

```
<body class="em_body" style="margin:0px; padding:0px;" bgcolor="#efefef">
<center class="wrapper">
<table align="center"
border="0"
cellspacing="0"
cellpadding="0"
class="main"
width="600"
style="width:600px;">
<tr>
<td style="padding:15px;"
align="center"
bgcolor="#efefef"
class="em_padd"
valign="top">
<!-- Aquí podemos definir más tablas y más celdas, -->
<!--tantas como se deseen -->
</td>
</tr>
</center>
</body>
```

## LÍMITES Y LIMITACIONES DE CSS

Sobre este tema, existen algunas diferencias según quién opine, pero, en general, podríamos afirmar que:

- No se pueden utilizar frameworks de CSS como Bootstrap, Materialize, PureCSS, Bulma o Foundation.
- Se debe evitar, siempre que se pueda, el uso de CSS3. Además, algunas propiedades como FLOAT o MARGIN deben utilizarse con precaución puesto que algunos clientes de correo como GMail y Outlook pueden no entenderlas o, incluso, ignorarlas.
- Aunque muchos clientes de correo permiten la adhesión de consultas de medios y definición reglas en la sección de cabecera, se debe realizar la definición de estilos a través de estilos en línea. Esto es así, porque

algunos clientes ignoran la etiqueta STYLE de HTML y otros sólo permiten la definición de esta etiqueta siempre y cuando esté declarada dentro del cuerpo del documento.

- En lo referente a fuentes de texto, su uso está muy limitado ya que la propiedad @FONT-FACE puede tomarse como invasiva y los clientes de correo pueden llegar a anular su declaración.
- El uso de la etiqueta LINK de HTML y la propiedad @IMPORT pueden también ser eliminadas o ignoradas, por lo que, una de las mejores opciones cuando se quieren usar fuentes personalizadas es recurrir una mezcla entre las fuentes personalizadas y las fuentes estándar de toda la vida como Arial o Times New Roman. No obstante, como decíamos antes, esto no es una imposición, sólo una recomendación.
- Se debe evitar el uso de propiedades relacionadas con el diseño como puedan ser POSITION, CLEAR, VISIBILITY, etcétera y las propiedades compuestas como FONT. Además, se deben definir sólo declaraciones de color en hexadecimal evitando la abreviatura de tres dígitos y la utilización de pseudo-clases y pseudo-elementos como ACTIVE o BEFORE.

Por último, me gustaría comentar que hay varios sitios web que nos proveen de información sobre la compatibilidad de cada una de las características que aceptan los clientes de correo. Un ejemplo, es <https://www.campaignmonitor.com/css/>.

## USO DE IMÁGENES Y VÍDEOS

Dado que no se dispone de una estructura de directorios como sucede en los sitios web, la adición de imágenes debe ser a través de servidores remotos, preferiblemente, mediante algún CDN.

Siempre que se pueda, se debe recurrir a imágenes en formato JPEG y GIF y evitar las imágenes rasterizadas en formato PNG.

Siempre hay que establecer las dimensiones de las imágenes. Esto es porque, algunos clientes de correo las pueden modificar a su antojo si no vienen definidas. Además, es importante que las imágenes tengan definidos los atributos ALT y BORDER.

También es una buena idea establecer en la cabecera el modo de visualización de bloque para las imágenes. Esto es, sobre todo, para que Outlook no añada márgenes internos a las imágenes.

Los vídeos, por ejemplo, sólo son compatibles con los clientes de correo electrónico de Apple y Mozilla Thunderbird, por lo que, la mejor solución ante esta situación es la inserción de una imagen con un enlace externo al video.

## HERRAMIENTAS PARA LA CREACIÓN DE EMAILS

Existen multitud de servicios y recursos que ofrecen la posibilidad de crear plantillas de email, si no del todo, prácticamente compatibles con la inmensa mayoría de clientes de correo. A continuación, se presentan algunos de estos recursos compatibles con los principales clientes de correo (Gmail, Inbox, Outlook, Apple Mail o Yahoo).

### Recurso

<https://github.com/rodriguezcommaj/salted>



*Para qué este recurso*

SALTED es un repositorio de GitHub que ofrece un modelo de plantilla utilizado por varios proveedores.

### **Recurso**

<https://github.com/TedGoas/Cerberus>



*Para qué este recurso*

CERBERUS es un repositorio de GitHub que ofrece varias plantillas compatibles con la gran mayoría de clientes de correo con algunos patrones de diseño sencillos separados por bloques y compartimentados para facilitar su reutilización.

### **Recurso**

<https://github.com/charlesmudy/Responsive-HTML-Email-Template>



*Para qué este recurso*

RESPMAIL es un repositorio de GitHub, similar a SALTED que ofrece un modelo de plantilla que funciona con los principales clientes de correo.

### **Recurso**

<https://mosaico.io/>



### *Para qué este recurso*

MOSAICO es un editor de plantillas de correo electrónico de código abierto que ofrece dos plantillas gratuitas y manipulables a través de tecnología Drag & Drop.

### **Recurso**

<https://www.campaignmonitor.com/email-templates/responsive/>



### *Para qué este recurso*

CAMPAIGN MONITOR es un servicio online que ofrece un editor de plantillas multitud de variaciones compatibles con los principales clientes de correo.

### **Recurso**

<https://putsmail.com/>



### *Para qué este recurso*

PUTSMAIL es un servicio online que ofrece la posibilidad de enviar los correos y su previsualización en los principales proveedores, entre otras funciones.

## **PRÁCTICA 14: NEWSLETTER DE UNIVERSES**

### **Código del ejemplo**

<https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-14>



### **Objetivo de la práctica**

Crear un email corporativo compatible con la mayoría de los clientes de correo.

## **Resultado**

## Se acaban las oportunidades de ver el cometa C/2020 F3 Neowise.

Sólo durante el mes de Julio de 2020 se podrá contemplar este gran fenomeno, tan asombroso, como inaudito. Aunque se encuentra a más de 103 millones de kilémetros, desde el 3 de Julio y mirando por debajo de la Osa Mayor a unos 15 grados al noroeste, se podrá ver con una magnitud aparente de unos +0.9 en el perihelio.

[Cómo y cuándo verlo](#)

Síguenos en: [Facebook](#) | [Twitter](#) | [Instagram](#)

[Política de Privacidad](#) | [Condiciones de uso](#) | ©2020 UniversES. Todos los derechos reservados.

Este mensaje no es SPAM. Ha sido enviado porque has dado tu consentimiento a UniversES a través del registro de <https://universes.com> o mediante la opción de newsletter. Si este correo lo has recibido por error, o no deseas recibir más correos electrónicos de UniversES, pulsa en [darse de baja](#)

Para poder realizar este ejemplo, lo único que se debe hacer descargar la plantilla de CERBERUS-FLUID y modificar los diferentes grupos de elementos. Seguidamente, añadiremos una imagen representativa del tema sobre el que se va a informar y le añadiremos una sección de enlaces a las redes sociales y el típico pie con textos informativos indicando que no es un mensaje de SPAM y la posibilidad de darse de baja del servicio de newsletter.

También, en el contenido del propio email, insertaremos un botón con un enlace a la web, la cual, aportará más información a quién le interese.

Cabe destacar que, aunque el logo debería ser una imagen, en esta práctica, está representado por un texto con una fuente vinculada a Google Fonts.

## Recursos para hacer la práctica

### Recurso

[https://commons.wikimedia.org/wiki/File:Comet\\_2020\\_F3\\_\(NEOWISE\)\\_on\\_Jul\\_14\\_2020\\_aligned\\_to\\_stars.jpg](https://commons.wikimedia.org/wiki/File:Comet_2020_F3_(NEOWISE)_on_Jul_14_2020_aligned_to_stars.jpg)



### Para qué este recurso

Esta es la dirección de la imagen para la imagen para el correo electrónico. Se puede escoger la versión que se desee, no obstante, como el ancho máximo de un email se ha establecido es de 600 píxeles, descargaremos la versión de 640x441. Su texto alternativo será el “Comet 2020 F3 (NEOWISE) on Jul 14 2020 aligned to stars”.

### Resolución

En nuestro caso, la solución más rápida y efectiva puede ser tomar una de las plantillas de CERBERUS y modificar los elementos para que se ajuste a nuestras necesidades.

Para más detalles consultar el resultado de esta práctica.

14

# DIRECTRICES PARA LA CREACIÓN DE GUÍAS DE ESTILOS

Una guía de estilos es un documento que representa, de forma visual y emotiva, la imagen corporativa de una organización o empresa.

Aunque pueden codificarse en formato de documentos electrónicos como PDF, PhotoShop o, incluso, Microsoft Word, lo más frecuente y recomendable es que estén codificadas en HTML5 y CSS3 para poder reutilizar código.

Cabe destacar que, antes de empezar a diseñar una guía de estilos web, primero se debe estudiar la marca y tener muy clara la imagen que quiere proporcionar teniendo en cuenta los ideales y valores que se desean transmitir sin perder el objetivo final ni fracturar o desprestigiar su historia.

En lo referente a su contenido y estructura, no se debe entrar en detalles ni justificaciones ya que es un documento informativo y no, un libro educativo. Tampoco suelen seguir un estándar, no obstante, sí que presentan unos puntos comunes que definen diferentes aspectos.

A continuación, se muestran las directrices que se deben cumplir para crear una guía de estilo de diseño web que cubra todas las necesidades de los desarrolladores. Aunque algunas de ellas pueden ser secciones o titulares, no todos están descritos con ese fin.

Aspecto	Descripción
<b>Tipografía</b>	La elección de la tipografía debe ir acorde a la imagen de la empresa u organización y no debe presentar características que puedan hacerla perder legibilidad. Entre los datos que debe proporcionar, se deben incluir el nombre de las fuentes a utilizar, todos los posibles niveles de jerarquía (H1...H6), todas las posibles variaciones de texto incluyendo cursiva y negrita, y todas las características referentes al tamaño, espaciado interlineal, espaciado entre caracteres y, la alineación y color, si procede.
<b>Paleta de colores</b>	Se deben describir todos los colores utilizados, incluyendo su contexto. Deben proporcionar una identidad de marca y estar bien combinados con colores neutros para facilitar la lectura. Además, se debe de intentar evitar la inclusión de más de múltiples de colores dominantes, aunque, si se necesita para definir la interfaz, pueden incluirse colores terciarios.
<b>El tono y la jerga</b>	El tono y la dialéctica empleadas en la página o aplicación deben transmitir el espíritu de la marca y ser compresibles para todo su público real y potencial. Piénsese que, no es lo mismo un sitio web para profesionales informáticos, que una web de compras. Por este motivo, hay que elegir una terminología y tono adecuado a los usuarios finales
<b>Iconografía</b>	Si la página utiliza iconos, que es lo más frecuente, se deben especificar todos y cada uno de ellos incluyendo su ámbito de aplicación y/o significado. En lo referente a su elección, se debe intentar basarse en alguna fuente vectorial que permita su fácil manipulación y transmite, inequívocamente, el concepto que se desea percibir sin ningún lugar a dudas. Además, se debe tratar de que sigan una línea gráfica adecuada para el sector y la empresa u organización. Para estos casos en los que, los iconos pertenecen a una fuente vectorial como pueda ser Material Design Icons o FontAweSome, se también es muy interesante especificar el nombre de la misma y el identificador del ícono (nombre de clase CSS o código hexadecimal asignado).
<b>Imágenes</b>	Con respecto a las imágenes, debe especificarse el estilo o características especiales que deban tener. Además, se debe indicar dónde se deben guardar y en qué formato. Si existe un CDN o banco de imágenes dónde extraer los posibles elementos gráficos, también es importante dejarlo claro para que todos los desarrolladores sigan la misma línea de diseño.
<b>Formularios y acciones</b>	Se debe asegurar la jerarquía e incluir los mensajes e identificadores más frecuentes y destacados. Esto es, mensajes típicos de error, advertencia y éxito y posibles etiquetas recurrentes o frecuentes.
<b>Navegación</b>	Se debe indicar dónde debe estar localizado el menú de navegación principal y poder distinguir los diferentes estados de los menús de navegación y enlaces. Por ejemplo, se debe indicar si el elemento cambia de color y/o clase CSS al estar seleccionado o activo.

	Además, si uno de estos elementos puede tener, a su vez, elementos adicionales como un valor numérico, se debe establecer su significado y un ejemplo.
<b>Botones</b>	Los botones deben poder distinguirse por forma, color y nombre de la acción. Es decir, debe mostrarse un ejemplo de todos y cada uno de los botones con sus acciones y nombres específicos. Si, además, utilizan una regla CSS o deben estar ubicados en alguna zona de la pantalla o componente, también debe especificarse.
<b>Espaciado</b>	El espaciado entre textos, imágenes, botones, titulares y demás elementos debe ser tenido en cuenta. Es decir, si un elemento de título debe llevar asociado un margen de cualquier tipo, o un espaciado entre caracteres, debe especificarse de forma gráfica con un ejemplo claro de uso.
<b>Especiales</b>	Si hay componentes o comportamientos predefinidos, se debe especificar cómo implementar con un ejemplo claro. Por ejemplo, si se utiliza un componente de pestañas, se debe proveer un ejemplo con todas las posibles casuísticas.
<b>Qué hacer y qué no hacer</b>	Es muy importante que se indiquen todas las directrices para que no se comentan errores de adaptación o creatividad. Es decir, se debe indicar si al logo se le pueden cambiar los colores del isotipo o fondo, las combinaciones permitidas de color y jerarquías, las palabras prohibidas si las hay, etcétera.

## EJEMPLOS DE GUÍAS DE ESTILO



## REFERENCIAS

- Casado, P. E. (2018). *Usabilidad Web Teoría y Uso*. RA-MA.
- Casado, P. E. (2020). *Domine JavaScript 4<sup>a</sup> Edición*. RA-MA.
- Commons Wikipedia. (2017). *Wikipedia*. Obtenido de <https://es.wikipedia.org/>
- HTMLQuick. (2020). *HTML Quick - Tutoriales y referencia sobre HTML*. Obtenido de <https://www.htmlquick.com/es/>
- Mozilla.org. (06 de 2020). <https://developer.mozilla.org/es/docs/Web/>. Obtenido de <https://developer.mozilla.org/es/docs/Web/>.
- W3C Schools. (2019, 06). *HTML The language for building web pages*. Retrieved from HTML The language for building web pages: <https://www.w3schools.com/>
- World Wide Web Consortium. (2018). *W3C*. Retrieved from <https://www.w3.org>