

UNIDAD 4. CREACIÓN DE INTERFACES WEB UTILIZANDO ESTILOS

1. INTRODUCCIÓN A CSS.....	3
1.1. AÑADIR ESTILOS A UN DOCUMENTO CON CSS.....	3
2. CREAR Y VINCULAR HOJAS DE ESTILO.....	4
2.1. CREACIÓN DE HOJAS DE ESTILO.....	6
2.1. CREAR Y VINCULAR HOJAS DE ESTILO EN CASCADA EXTERNA.....	6
2.3. CREACIÓN, VINCULACIÓN E IMPORTACIÓN MÚLTIPLE DE REGLAS CSS.....	7
3. APLICACIÓN DE ESTILOS.....	9
3.1. SELECTORES BÁSICOS.....	9
3.1.1. <i>Selector universal</i>	9
3.1.2. <i>Selector de tipo o etiqueta</i>	9
3.1.3. <i>Selector descendente</i>	10
3.1.4. <i>Selector de clase</i>	11
3.1.5. <i>Selector por identificador</i>	12
3.2. SELECTORES AVANZADOS.....	13
3.2.1. <i>Selector de hijos</i>	13
3.2.2. <i>Selector adyacente</i>	13
3.2.3. <i>Selector de atributos</i>	14
3.3. LA HERENCIA EN LA ANIDACIÓN DE ELEMENTOS.....	15
3.4. UNIDADES DE MEDIDA.....	17
3.5. COLORES.....	19
3.6. TEXTO.....	21
3.6.1. <i>Tipografía</i>	21
3.6.2. <i>Apariencia del texto</i>	24
Textos y espacios.....	24
Alineaciones.....	27
Variaciones.....	29
3.6.3. <i>Las fuentes incrustadas</i>	30
Los formatos de fuente.....	30
La regla @font-face.....	31
El nombre de las fuentes.....	32
Las fuentes incrustadas locales.....	33
Indicar varios formatos.....	33
Los estilos tipográficos.....	33
3.6.4. <i>Las fuentes en línea</i>	34
Las Google Fonts.....	34
3.7. ENLACES.....	37
3.8. IMÁGENES.....	38
3.9. LISTAS.....	38
3.10. TABLAS.....	41
3.11. FORMULARIOS.....	42
3.11.1. <i>Dar formato</i>	43
3.11.2. <i>Redimensionar un campo</i>	43
3.11.3. <i>Pseudoclases para formularios</i>	45
3.11.4. <i>Los campos requeridos y los opcionales</i>	47
3.11.5. <i>Dar formato al «foco»</i>	48
3.11.6. <i>Validar la información introducida</i>	49
3.12. CURSOR.....	52
4. PROPIEDADES CSS3.....	52
4.1. <i>Los módulos CSS3</i>	52

4.2. Etapas en la concepción de las CSS3.....	53
4.3. Los prefijos de los navegadores.....	56
4.4. El sitio Can I use.....	58

5. HERRAMIENTAS Y TEST DE VERIFICACIÓN.....61

1. INTRODUCCIÓN A CSS

Las hojas de estilo en cascada (CSS - Cascading Style Sheets) son un estándar W3C que define la presentación de los documentos Web, es decir, el modo en el que se muestra un documento en pantalla o se suministra al usuario, ya sea por el monitor, en la pantalla del teléfono móvil o leído por un lector de pantalla. Lo más importante es que con CSS se mantienen las instrucciones de presentación separadas del contenido del documento XHTML.

CSS es un lenguaje que trabaja junto con HTML para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo, bordes, etc...A pesar de que cada navegador garantiza estilos por defecto para cada uno de los elementos HTML, estos estilos no necesariamente satisfacen los requerimientos de cada diseñador. Normalmente se encuentran muy distanciados de lo que queremos para nuestros sitios webs. Diseñadores y desarrolladores a menudo deben aplicar sus propios estilos para obtener la organización y el efecto visual que realmente desean.

Las hojas de estilo en cascada como hoy las conocemos, comenzaron cuando Håkon Lie publicó su primer borrador de hojas de estilo HTML en cascada, al que pronto se le unió Bert Bos, gran impulsor de este estándar.

- CSS Nivel 2, Revisión 1 que es simplemente una recomendación que realiza unos ajustes menores a CSS2 basándose en la experiencia de trabajo con él entre 1998 y 2004.
- CSS Nivel 3, basada en módulos, añade soporte para texto flotante vertical, mejor manejo de tablas, idiomas internacionales y una mejor integración con otras tecnologías XML como SVG, MathM

El CSS WG también está trabajando en conjuntos CSS especiales orientados a medios específicos como: CSS Mobile, CSS Print y CSS TV.

1.1. AÑADIR ESTILOS A UN DOCUMENTO CON CSS.

Sin duda, no existe ninguna desventaja por utilizar CSS en la maquetación de páginas web, las ventajas que podemos destacar son las siguientes:

- **Mayor control en el diseño de las páginas:** Se puede llegar a diseños fuera del alcance de HTML.
- **Menos trabajo:** Se puede cambiar el estilo de todo un sitio con la modificación de un único archivo.

- **Documentos más pequeños:** Las etiquetas `` y la gran cantidad de tablas empleadas para dar una buena apariencia a los sitios web desaparecen ahora, por lo que se ahorra código en la configuración de la presentación del sitio.
- **Documentos mucho más estructurados:** Los documentos bien estructurados son accesibles a más dispositivos y usuarios.
- **El HTML de presentación está en vías de desaparecer:** Todos los elementos y atributos de presentación de las especificaciones HTML y XHTML fueron declarados obsoletos por el W3C.
- **Tiene buen soporte:** En este momento, casi todos los navegadores soportan casi toda la especificación CSS1 y la mayoría también las recomendaciones de nivel 2 y 3.

El proceso de funcionamiento de las hojas de estilo en cascada podemos resumirlo en tres pasos:

1. Hay que comenzar con un documento XHTML (o HTML). En teoría, el documento tendrá una estructura lógica y un significado semántico a través de los elementos XHTML adecuados. Con XHTML se crea la estructura de la página web.
2. Luego hay que escribir las reglas de estilo para definir el aspecto ideal de todos los elementos. Las reglas seleccionan el elemento en cuestión por su nombre y, a continuación, listan las propiedades (fuente, color, etc.) y los valores que se le van a aplicar.
3. Por último, hay que vincular los estilos al documento. Las reglas de estilo pueden reunirse en un documento independiente y aplicarse a todo el sitio, o pueden aparecer en la cabecera y aplicarse solo a ese documento.

A la hora de aplicar las reglas de estilo a un documento (X)HTML, debes tener en cuenta que existen tres modos distintos:

- Estilos en línea.
- Hojas de estilos incrustados.
- Hojas de estilos externas: vinculadas o importadas.

2. CREAR Y VINCULAR HOJAS DE ESTILO

Las hojas de estilo se componen de **reglas**, cada una de las cuales incluye un **selector** y, entre llaves, una **declaración**. El selector indica el elemento o elementos HTML al que se aplica la regla CSS y la declaración especifica los estilos que se aplican a los elementos indicados en el selector.

Cada declaración está compuesta por una o más **propiedades** y sus **valores** correspondientes. Una propiedad permite modificar el aspecto de una característica del elemento y un valor indica el nuevo valor de la característica modificada en el elemento.

He aquí una regla CSS muy simple:

```
.miEstilo{color: red; font-style: italic;}
```

Estos son los elementos de la sintaxis:

`.miEstilo` es el nombre del selector.

`{color: red; font-style: italic;}` es la declaración, situada entre llaves. Esta declaración utiliza las propiedades `color` y `font-style`, separadas por un punto y coma, con los valores `red` e `italic`, respectivamente.

Este estilo simplemente **permite escribir texto en rojo y cursiva**.

Cuando una declaración comporta varias propiedades, **cada propiedad se separa por un punto y coma**:

```
.miEstilo{color: red; font-size: 12pt; text-align: center}
```

Esta sintaxis es perfectamente válida, pero se acostumbra a poner cada par propiedad-valor en una línea para que el código resulte más legible. En ese caso, la última línea de propiedad también termina con un punto y coma.

He aquí el ejemplo anterior con una sintaxis más legible:

```
.miEstilo{  
    color: red;  
  
    font-size: 12pt;  
  
    text-align: center;  
}
```

Las reglas CSS deben respetar reglas de nomenclatura:

- Dé prioridad a las minúsculas cuando escriba reglas CSS.
- El nombre de los selectores no debe comenzar por un guion ni por un número.
- El nombre de los selectores no debe incluir espacios, caracteres de puntuación, especiales ni acentuados.

Una de las características más importantes de CSS es la herencia, de modo que, en general, las propiedades de los elementos se heredan de los elementos que los contienen, salvo si se les especifica una propiedad particular. De todas formas, la interpretación de la herencia puede variar según el navegador.

2.1. CREACIÓN DE HOJAS DE ESTILO

Existen tres formas de incluir hojas de estilo en un documento XHTML

- Utilizando la **etiqueta <style>** dentro de la cabecera <head>. Por ejemplo:

```
<head>
...
  <style type="text/css">
    p {color: red; font-family: sans-serif;}
  </style>
...
</head>
```

- En las propias etiquetas HTML, **utilizando el atributo style="definición de estilo"**. Por ejemplo:

```
<body>
...
<p style = "color: red; font-family: sans-serif;">Este texto es de color rojo y
fuente sans-serif.</p>
...
</body>
```

- **Utilizando un archivo** externo y enlazándolo mediante la etiqueta link:

```
<link rel="stylesheet" type = "text/css" href = "url archivo css" media =
"screen"/>
```

- También se puede **importar el archivo externo desde la etiqueta <style>** poniendo, como primera línea:

```
<style>          se modifica el contenido de la pag web, metiendo en el style todo el contenido de la url. Reglas incrustadas,
@import "url";   preferencia sobre link
... resto de definición de estilo...
```

2.1. CREAR Y VINCULAR HOJAS DE ESTILO EN CASCADA EXTERNA

Las hojas de estilo son **documentos de texto** con, por lo menos, una regla. Estos archivos no contienen ninguna etiqueta HTML. Al igual que en los documentos HTML, en las hojas de estilo se pueden incluir **comentarios** pero, en este caso, se escriben del siguiente modo: **/* Este es un comentario */**

CSS2 introduce la posibilidad de orientar las hojas de estilo a medios de presentación específicos. Para ello se emplea el atributo **media** del elemento **link**.

```
<link rel="stylesheet" type = "text/css" href = "url archivo css" media =
"screen"/>
```

Medios de presentación	Valor atributo media
all	Todos los medios definidos.
braille	Dispositivos táctiles que emplean el sistema Braille.
embosed	Impresoras que emplean el sistema Braille.
handheld	Dispositivos de mano: móviles, PDA , etcétera.
print	Impresoras y navegadores en el modo "vista previa para imprimir".
projection	Proyectores y dispositivos para presentaciones.
screen	Pantallas de ordenador.
speech	Sintetizadores para navegadores de voz empleados por personas discapacitadas.
tty	Dispositivos textuales limitados, como teletipos y terminales de texto.
tv	Televisores y dispositivos con resolución baja.

En el ejemplo siguiente se muestra cómo se pueden emplear en la función @media de la misma forma que hacíamos con la función @import.

```
<style type="text/css">
<!--
@import url(http://estilos/miestilo.css);
@media print
{
    body{ font-size: 10pt; } /* Establece el tamaño de fuente para impresión */
}
@media screen
{
    body { font-size: 13px } /* Establece el tamaño de fuente para visualización
    */
}
p {font-face: Verdana;}
-->
</style>
```

2.3. CREACIÓN, VINCULACIÓN E IMPORTACIÓN MÚLTIPLE DE REGLAS CSS

Cuando ha creado varias hojas de estilos CSS, puede vincularlas e importarlas a una misma página HTML, así como importarlas entre ellas. También puede añadir estilos CSS creándolos en la página HTML y en los elementos.

Para vincular varios archivos CSS a una página HTML, utilice tantas veces como sea preciso <link>:

```
<head>
...
<link href="mis-estilos.css" rel="stylesheet" media="screen">
<link href="principal.css" rel="stylesheet" media="screen">
...
</head>
```

Para importar varios archivos CSS en una página HTML, use tantas veces como sea preciso la regla `@import` en el elemento `<style>`. Recuerde que deben colocarse al principio del elemento `<style>`, antes de crear las reglas de estilos CSS internas en la página:

```
<style type="text/css" media="screen">
  @import url("marketing.css");
  @import url("principal.css");
  .miEstilo{
    color: red;
  }
  p {
    background-color: yellow;
  }
  ...
</style>
```

Tenga en cuenta desde ahora que el orden de aparición en el código de los vínculos y las importaciones es relevante. Lo veremos en los párrafos dedicados a la cascada.

He aquí un **ejemplo** de código en el que encontrará vínculos a archivos `.css`, importaciones con la regla `@`, estilos CSS creados en la página y estilos CSS creados directamente en un elemento HTML:

```
<head>
  ...
  <link href="mis-estilos.css" rel="stylesheet" media="screen">
  <link href="principal.css" rel="stylesheet" media="screen">
  <style type="text/css" media="screen">
    @import url("marketing.css");
    @import url("principal.css");
    .miEstilo{
      color: red;
    }
    p.amarillo {
      background-color: yellow;
    }
  </style>
  ...
</head>
<body>
  ...
  <h3 style="color: green">Mi título</h3>
  <p class="amarillo">Lorem ipsum...</p>
  ...
</body>
```


3. APLICACIÓN DE ESTILOS

Como vimos anteriormente, una regla CSS está formada por una parte llamada “**selector**” y otra parte llamada “**declaración**”. La declaración indica “qué hay que hacer” y el selector indica “qué elementos van a ser afectados”.

A un mismo elemento HTML se le pueden asignar varias reglas CSS y cada regla CSS puede aplicarse a varios elementos. Es decir, una misma regla se puede aplicar a diferentes selectores y un mismo selector se puede utilizar en varias reglas.

El estándar CSS 2.1 incluye doce tipos diferentes de selectores que permiten seleccionar de forma muy precisa elementos individuales o conjuntos de elementos dentro de una página web. Sin embargo, la mayoría de los sitios web se pueden diseñar utilizando solamente los cinco selectores básicos.

3.1. SELECTORES BÁSICOS.

3.1.1. SELECTOR UNIVERSAL.

Se utiliza para seleccionar todos los elementos de la página. Se indica mediante un asterisco y, a pesar de su sencillez, no se utiliza habitualmente ya que es raro que un mismo estilo se pueda aplicar a todos los elementos de una página.

El siguiente ejemplo elimina el margen y el relleno de todos los elementos HTML:

```
*{
margin: 0;
padding: 0;
}
```

4.1.2. SELECTOR DE TIPO O ETIQUETA.

Selecciona todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector. Para utilizar este selector, solamente es necesario indicar el nombre de una etiqueta HTML (sin los caracteres < y >) correspondiente a los elementos que se quieren seleccionar.

El siguiente ejemplo selecciona todos los párrafos de la página:

```
p {
    ... declaraciones...
}
```

Si se quiere aplicar los mismos estilos a dos etiquetas diferentes, se puede aplicar un selector múltiple a una sola regla. Para esto, se incluyen todos los selectores, separados por una coma. Por ejemplo:

```
h1, h2, h3 {
    color: #8A8E27;
    font-weight: normal;
    font-family: Arial, Helvetica, sans-serif;
}
```

En las hojas de estilos complejas es habitual agrupar las propiedades comunes de varios elementos en una única regla CSS y, posteriormente, definir las propiedades específicas de esos mismos elementos. Por ejemplo, tras el código anterior, podemos escribir:

```
h1 {font-size: 2em}
h2 {font-size: 1.5em}
h3 {font-size: 1.2em}
```

4.1.3. SELECTOR DESCENDENTE.

Recordemos que un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y cierre de otro elemento. Pues bien, un selector descendiente es aquél que permite seleccionar elementos que se encuentran dentro de otros elementos. En CSS, se indica mediante el término `span`.

El selector del ejemplo siguiente selecciona todos los elementos `` de la página que se encuentren dentro de un elemento `<p>`:

```
p span { color: red; }
```

Si éste se aplica al siguiente código HTML de una página:

```
<p>
    <span>texto1</span>
    ...
    <a href=" " >... <span> texto2 </span></a>
</p>
```

El selector `p span` seleccionará tanto `texto1` como `texto2`. El motivo es que en el selector descendiente, un elemento no tiene que ser “hijo directo” de otro, sino que basta con que se encuentre dentro de él.

Al resto de los elementos `` de la página que no están dentro de un elemento `<p>`, no se les aplica la regla CSS anterior.

La sintaxis formal del selector descendiente se muestra a continuación:

elemento1 elemento2 elemento3 ... elementoN

donde elementoN indica el elemento sobre el que se aplican los estilos y el resto indica el lugar en que se tiene que encontrar elementoN para que los estilos se apliquen realmente.

En el siguiente ejemplo, el selector descendiente contiene cuatro elementos:

```
p a span em {text-decoration: underline; }
```

El estilo subrayado de esta regla se aplicará a los elementos de tipo `` que se encuentren dentro de elementos de tipo `` que, a su vez, estén

contenidos en elementos de tipo `<a>` que estén dentro de elementos de tipo `<p>`.

4.1.4. SELECTOR DE CLASE.

Este selector permite aplicar estilo a uno o varios elementos de una página. Por ejemplo, si en el siguiente código HTML queremos que los textos del primer y tercer párrafo se muestren en rojo:

```
<body>
  <p>Lorem ipsum dolor sit amet...</p>
  <p>Nunc sed lacus et est adipiscing accumsan...</p>
  <p>Class aptent taciti sociosqu ad litora...</p>
</body>
```

Ninguno de los selectores vistos hasta el momento nos lo permiten. Sin embargo, existe una solución que consiste en:

Utilizar el atributo `class` de HTML sobre el elemento para indicar directamente la regla CSS que se le debe aplicar:

```
<body>
  <p class="destacado">Lorem ipsum dolor sit amet...</p>
  <p>Nunc sed lacus et est adipiscing accumsan...</p>
  <p class="destacado">Class aptent taciti sociosqu ad litora...</p>
</body>
```

Crear, en el archivo CSS, una nueva regla cuyo selector es el valor del atributo `class`, precedido por el carácter punto y con todos los estilos que se quieren asignar. En nuestro ejemplo:

```
.destacado {color: red; }
```

El selector `.destacado` se interpreta como “cualquier elemento de la página cuyo atributo `class` sea igual a `destacado`”.

La principal característica de este selector es que, en una misma página HTML, varios elementos diferentes pueden utilizar el mismo valor en el atributo `class`. En el siguiente ejemplo, se aplica el estilo `destacado` a varios elementos:

```
<body>
  <p class="destacado">Lorem ipsum dolor sit amet...</p>
  <p>Nunc sed lacus et<a href="#" class="destacado">est adipiscing
  </a>accumsan...</p>
  <p>Class aptent taciti<em class="destacado"> sociosqu ad </3m>litora...</p>
</body>
```

Por el contrario, a veces, es necesario restringir el alcance del selector de clase. Para ello, escribiremos el selector de tipo o etiqueta y el de clase. Supongamos

que en el ejemplo anterior, sólo queremos aplicar el estilo al párrafo que lleve el atributo class con valor destacado. Para ello, utilizaremos el selector:

```
p.destacado {color: red; }
```

Hay que distinguir, por tanto, entre estas expresiones:

p.destacado {...}	Todos los elementos de tipo “p” con atributo class=“destacado” <small>Solo habra un elemento con ese id</small>
p .destacado {...}	Todos los elementos con atributo class=“destacado” que estén dentro de cualquier elemento de tipo “p”
p, .destacado {...}	Todos los elementos “p” de la página y todos los elementos con atributo class=“destacado” de la página

4.1.5. SELECTOR POR IDENTIFICADOR.

En ocasiones, es necesario aplicar estilos a un único elemento de la página. Aunque puede utilizarse el selector de clase para aplicar estilos a un único elemento, existe otro selector más eficiente en este caso.

El selector por identificador selecciona un único elemento a través del valor de su atributo id, cuyo valor nunca se puede repetir en dos elementos diferentes de una misma página.

La sintaxis de los selectores por identificador es análoga a la de los selectores de clase, salvo porque se utiliza el símbolo almohadilla (#), en vez del punto (.), como prefijo del nombre de la regla CSS.

En el siguiente ejemplo, el selector #destacado solamente selecciona el segundo párrafo:

```
#destacado {color: red; }
<p>Primer párrafo</p>
<p id="destacado">Segundo párrafo</p>
<p>Tercer párrafo</p>
```

En general, se recomienda usar el selector por identificador cuando se quiere aplicar un estilo a un solo elemento específico de una página y el selector de clase cuando se quiere aplicar un estilo a varios elementos diferentes de la misma.

Cuando se trabaja sobre varias páginas suele resultar útil restringir el alcance de un selector por identificador. En el siguiente ejemplo se aplica la regla CSS sólo al elemento de tipo <p> que tenga un atributo id igual al indicado:

```
p#destacado {color: red; }
```

En este caso, algunas páginas pueden disponer de elementos con un atributo id igual a destacado y que no sean párrafos, por lo que la regla anterior no se aplicará sobre esos elementos.

No debe confundirse el selector por identificador con los selectores anteriores:

p#destacado {...}	Todos los elementos de tipo "p" con atributo id="destacado"
p #destacado {...}	Todos los elementos con atributo id="destacado" que estén dentro de cualquier elemento de tipo "p"
p, #destacado{ ...}	Todos los elementos "p" de la página y todos los elementos con atributo id="destacado" de la página

3.2. SELECTORES AVANZADOS.

Aunque con los selectores básicos es posible diseñar prácticamente cualquier página web, tenemos la opción de utilizar los selectores avanzados para simplificar las hojas de estilos.

El navegador Internet Explorer 6 y sus versiones anteriores no soportan este tipo de selectores. Se puede consultar la lista de los selectores que soporta cada versión de cada navegador en:

<http://dev.l-c-n.com/CSS3-selectors/browser-support.php>

4.2.1. SELECTOR DE HIJOS.

Se utiliza para seleccionar un elemento que es **hijo directo** de otro elemento y se indica mediante el signo mayor que (>). Veamos un ejemplo:

```
p > span {color: blue; }
```

```
<p><span>texto1</span></p>
<p><a href="#"><span>Texto2</span></a></p>
```

El selector p > span se interpreta como "cualquier elemento que sea hijo directo de un elemento <p>". Por ello, el primer elemento cumple la condición del selector, pero el segundo no la cumple porque es descendiente pero no es hijo directo de <p>.

4.2.2. SELECTOR ADYACENTE.

El selector adyacente utiliza el **signo +** y su sintaxis es la siguiente:

elemento1 + elemento2 { ...}

Con él se seleccionan todos los elementos de tipo elemento2 que cumplan las dos siguientes condiciones:

- **elemento1 y elemento2 deben ser hermanos**, por lo que su padre debe ser el mismo.
- **elemento2 debe aparecer inmediatamente después de elemento1** en el código HTML de la página.

En el siguiente ejemplo:

```
h1 + h2 {color: red;}

<body>
  <h1>Título1</h1>
  <h2>Subtítulo</h2>
  ...
  <h2>Otro subtítulo</h2>
</body>
```

Los estilos del selector h1 + h2 se aplican al primer elemento h2 de la página, pero no al segundo <h2>, ya que:

El elemento padre de <h1> es <body>, el mismo que el de los dos elementos <h2>. Así, los dos elementos <h2> cumplen la primera condición del selector adyacente.

El primer elemento <h2> aparece en el código HTML, justo después del elemento <h1>, por lo que este elemento <h2> también cumple la segunda condición del selector adyacente.

Por el contrario, el segundo elemento <h2> no aparece justo después del elemento <h1>, por lo que no cumple la segunda condición del selector adyacente y, por tanto, no se aplican los estilos h1 + h2.

4.2.3. SELECTOR DE ATRIBUTOS.

Permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos.

Existen estos cuatro tipos de selectores de atributos:

- **[nombre_atributo]**, selecciona los elementos, que tienen establecido el atributo llamado nombre_atributo, independientemente de su valor. Ejemplo: a[class] {color: blue;}
- **[nombre_atributo=valor]**, selecciona los elementos que tienen establecido un atributo llamado nombre_atributo con valor igual a valor. Ejemplo: a[class="externo"] {color: blue;}

- **[nombre_atributo~=valor]**, selecciona los elementos que tienen establecido un atributo llamado nombre_atributo con valor igual a valor, siendo valor un texto de entre varios que tiene el atributo. Ejemplo: a[title~=“externo”] {color: blue;}
- **[nombre_atributo^=valor]**, aplica un estilo a un elemento cuyo atributo atributo comienza exactamente por valor.
- **[nombre_atributo\$=valor]**, aplica un estilo a un elemento cuyo atributo atributo termina exactamente por valor.
- **[nombre_atributo*=valor]**, aplica un estilo a un elemento cuyo atributo atributo contiene al menos una vez valor.

3.3. LA HERENCIA EN LA ANIDACIÓN DE ELEMENTOS

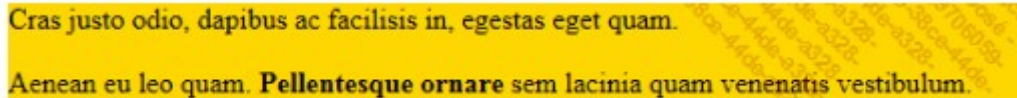
Debe saber que, cuando anida un elemento HTML en otro, el elemento incluido hereda las propiedades CSS del elemento de nivel superior. El elemento hijo hereda las propiedades CSS de su padre.

Veamos este ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>La herencia padre-hijo</title>
  <meta charset="UTF-8" />
  <style>
    .dorado {
      background-color: gold;
    }
  </style>
</head>
<body>
<div class="dorado">
  <p>Cras justo odio, dapibus ac facilisis in, egestas eget quam.</p>
  <p>Aenean eu leo quam. <strong>Pellentesque ornare</strong> sem lacinia
quam venenatis vestibulum.</p>
</div>
</body>
</html>
```

El elemento padre <div> utiliza la clase .dorado, que muestra el texto sobre un fondo dorado. Los elementos incluidos, los párrafos <p>, heredan esta propiedad y, por lo tanto, se mostrarán con un fondo dorado. En el segundo párrafo <p>, el elemento hijo aparecerá asimismo sobre un fondo dorado, también por herencia.

Esto es lo que se muestra:



Cras justo odio, dapibus ac facilisis in, egestas eget quam.

Aenean eu leo quam. **Pellentesque ornare sem lacinia quam venenatis vestibulum.**

Modifiquemos ahora nuestro código:

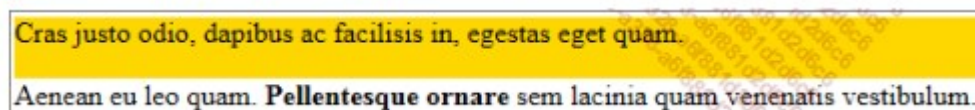
```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>La herencia padre-hijo</title>
  <meta charset="UTF-8" />
  <style>
    .dorado {
      background-color: gold;
    }
    .blanco {
      background-color: white;
    }
  </style>
</head>
<body>
<div class="dorado">
  <p>Cras justo odio, dapibus ac facilisis in, egestas eget quam.</p>
  <p class="blanco">Aenean eu leo quam.
<strong>Pellentesque ornare</strong> sem lacinia quam venenatis
vestibulum.</p>
</div>
</body>
</html>
```

Hemos añadido una clase, .blanco, que muestra un fondo blanco. Esta clase se aplica al segundo párrafo.

El primer párrafo conserva su herencia, por lo que se muestra sobre un fondo dorado.

El segundo párrafo, con su clase .blanco, no hereda el fondo dorado, sino que utiliza la clase que se le ha aplicado. Por lo tanto, muestra un fondo blanco, igual que el elemento , que hereda esta clase.

Esto es lo que se muestra:



Cras justo odio, dapibus ac facilisis in, egestas eget quam.

Aenean eu leo quam. **Pellentesque ornare sem lacinia quam venenatis vestibulum.**

Preste atención: un determinado número de propiedades CSS no se heredan, como las propiedades margin y padding de los párrafos, por ejemplo. He aquí el ejemplo de la propiedad margin en la recomendación de las CSS del W3C (<http://www.w3.org/TR/CSS2/box.html#margin-properties>): **Inherited: no.**

'margin'	
Value:	<margin-width>{1,4} inherit
Initial:	see individual properties
Applies to:	all elements except elements with table display types other than table-caption, table and inline-table
Inherited:	no
Percentages:	refer to width of containing block
Media:	visual
Computed value:	see individual properties

3.4. UNIDADES DE MEDIDA.

Las medidas en CSS se emplean, entre otras cosas, para definir la altura, anchura y márgenes de los elementos y para establecer el tamaño de letra del texto. Todas las medidas se expresan con un valor numérico entero o decimal seguido de una unidad de medida, sin ningún espacio en blanco entre ellos.

Las unidades de medida CSS pueden ser de dos tipos:

- **Relativas:** definen su valor en relación con otra medida por lo que, para obtener su valor, se debe realizar alguna operación con el valor indicado.
- **Absolutas:** establecen de forma completa el valor de una medida, por lo que su valor real es exactamente el especificado.

Unidades relativas.

Estas unidades son más flexibles que las unidades absolutas porque se adaptan más fácilmente a los diferentes medios. Son las siguientes:

- **em**, relativa al tamaño de letra empleado. Aunque no es una definición exacta, el valor 1em se puede aproximar por la altura de la letra “m” del tipo de letra que se esté empleando. Por ejemplo, `h1 { font-size: 2em; }` indica que el tamaño de letra para el texto referenciado por `<h1>` será el doble del tamaño que se esté usando. 1 em corresponde al tamaño de fuente utilizado. Si ha definido un tamaño de fuente de 12 puntos, entonces un 1 em vale 12 puntos. Se trata, por lo tanto, de una unidad relativa en función del tamaño de letra usado. Debe saber que todos los navegadores utilizan una fuente de 16 píxeles como base de referencia para la unidad em, de lo que se deduce que 1 em = 16 píxeles. Recuerde que la cascada se aplica también a las unidades de medida para calcular el tamaño de las fuentes.
- **El rem.** Esta nueva unidad se refiere a la raíz de una página (root em), al elemento `<html>`. Podemos indicar un valor en em para la raíz de la página y los demás valores en rem, es decir, relativos a la raíz definida; de este modo, se evita la cascada en el cálculo del tamaño de las fuentes.
- **ex**, también es relativa al tamaño de la letra empleada pero, esta vez, respecto a la altura de la letra “x”.
- **px**, píxel, relativa respecto a la pantalla del usuario.
- **%**, porcentaje, hace referencia a otra medida.

Estas medidas se pueden utilizar en diferentes elementos de una misma página, como se puede ver en el siguiente ejemplo:

```
body { font-size: 10px; }
```

```
h1 { font-size: 2.5em; }
```

En la primera línea se establece un tamaño de letra base de 10 pixels para toda la página. En la segunda, se asigna el tamaño 2.5em al elemento <h1>, por lo que su tamaño de letra real será de $2.5 \times 10\text{px} = 25\text{px}$.

Unidades absolutas.

CSS permite utilizar las siguientes unidades absolutas:

- **in**, pulgadas (en inglés, “inches”). Una pulgada equivale a 2.54 centímetros.
- **cm**, centímetros.
- **mm**, milímetros.
- **pt**, puntos. Un punto equivale a 1/72 pulgadas, es decir, a unos 0.35 milímetros.
- **pc**, picas. Una pica equivale a 1/12 puntos, es decir, unos 4.23 milímetros.

He aquí algunos ejemplos de utilización de estas unidades:

```
body { margin: 0.5in; }  
h1 {line-height: 2cm; }  
p {word-spacing: 4mm; }  
a { font-size: 12 pt; }  
span { font-size: 1pc; }
```

Otras unidades

También puede trabajar con estas otras unidades:

- El **grado**, representado por deg.
- El **gradiente**, representado por grad.
- El **radián**, representado por rad.
- **turn** corresponde al giro completo de un círculo.
- El **segundo**, representado por s.
- El **milisegundo**, representado por ms.
- El **Dot Per Inch** (puntos por pulgada), representado por dpi, para indicar la resolución de una imagen.

- El **Dot Per Centimeter** (puntos por centímetro), representado por dpcm, para indicar la resolución de una imagen.
- El **Dot Per Pixel** (puntos por píxel), representado por dpx, para indicar la resolución de una imagen.

3.5. COLORES.

Los colores en CSS se pueden indicar de cinco modos diferentes: palabras clave, colores del sistema, RGB hexadecimal, RGB numérico y RGB porcentual. De ellos, el más habitual es el RGB hexadecimal.

Palabras clave.

CSS define 17 palabras clave para referirse a los colores básicos, que corresponden al nombre en inglés de cada color: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow.

Aunque es una forma muy sencilla de referirse a los colores básicos, este método

maroon #800000	red #ff0000	orange #ffa500	yellow #ffff00	olive #808000
purple #800080	fuchsia #ff00ff	white #ffffff	lime #00ff00	green #008000
navy #000080	blue #0000ff	aqua #00ffff	teal #008080	
black #000000	silver #c0c0c0	gray #808080		

apenas se utiliza porque permite representar una gama de colores muy limitada.

Además de esta lista básica, los navegadores modernos soportan muchos otros nombres de colores. La lista completa se puede ver en: https://es.wikipedia.org/wiki/Colores_web.

RGB decimal.

El modelo RGB permite definir un color indicando las cantidades de las componentes rojo (Red), verde (Green) y azul (Blue) que se deben mezclar para obtenerlo. Estas cantidades pueden ir desde cero a un valor máximo. En concreto, en CSS, las componentes de los colores definidos mediante RGB decimal pueden tomar valores entre 0 y 255

Si el valor de cada componente es cero, el color creado será el negro y, si es el máximo, se obtendrá el blanco. Por otro lado, para crear el color rojo puro se utilizará el máximo valor para la cantidad de rojo y valor cero para la cantidad de verde y azul. La sintaxis para indicar un color consiste en escribir `rgb()` y, dentro del paréntesis, los valores de las tres componentes separados por comas. En el siguiente ejemplo se establece el color del texto de un párrafo:

```
p {color: rgb(71, 98, 176); }
```

Las CSS3 permiten usar un cuarto parámetro para indicar el grado de transparencia del color; se trata del parámetro *a*, que corresponde al canal alfa y que gestiona la transparencia. Este valor va de 0, totalmente transparente, a 1, opaco. Por ejemplo: `rgba(0,0,255,0.5)`: un azul semitransparente (0,5=50 % de transparencia).

RGB porcentual.

El funcionamiento y la sintaxis de este método es el mismo que el del RGB decimal, excepto porque el valor de las componentes está comprendido entre 0% y 100%. El ejemplo anterior también se podría escribir así:

```
p {color: rgb(27%, 38%, 69%); }
```

RGB hexadecimal.

En este caso, el valor de cada componente se expresa mediante un número hexadecimal comprendido entre 0 y FF. Sin embargo, la sintaxis para indicar un color concreto varía ligeramente puesto que simplemente se escribirá el símbolo *#* seguido de dos dígitos hexadecimales para cada una de las componentes. El ejemplo de las secciones anteriores se escribe así:

```
P {color: #4762B0; }
```

Una ventaja de este formato es que permite comprimir sus valores cuando las tres componentes contienen un mismo dígito repetido. Por ejemplo: El color `#AA0066` puede expresarse como `#A06`.

En el siguiente ejemplo se establece el color de fondo de la página a blanco, el del texto a negro y el de los titulares a rojo:

```
body {background-color: #FFF; color: #000; }  
h1, h2, h3, h4, h5, h6 {color: #C00; }
```

La notación HSL

Otra posibilidad es indicar un color especificando los valores de los tres componentes HSL: **matiz** (hue), **saturación** (saturation) y **brillo** (lightness). El componente del matiz va de 0 (rojo), a 60 (amarillo), a 120 (verde), a 180 (turquesa), a 240 (azul), a 300 (magenta). La unidad es un grado en la rueda cromática. El componente de saturación va de 0 (color desaturado) a 100 (color saturado). El componente del brillo va de 0 (negro) a 100 (blanco). Las unidades para la saturación y el brillo son porcentajes.

Ejemplos:

- `hsl(100,75,90)` es un verde claro.
- `hsl(328,81,51)` es un violeta oscuro.

Como en el caso anterior, es posible añadir un grado de transparencia con la propiedad `hsla(328,81,51,0.3)`.

3.6. TEXTO.

Comencemos por el uso de fuentes tipográficas «**locales**». El término «local» no es nomenclatura estándar; simplemente designa las fuentes instaladas en el equipo del internauta. No cuesta mucho entender que esta solución resulta muy limitada para los diseñadores, ya que los restringe a la utilización de fuentes de aspecto muy similar que se suministran por defecto con todos los sistemas operativos de los internautas (Windows, Mac y Linux-Unix). Estas fuentes pueden clasificarse usualmente en tres familias genéricas:

- Fuentes con remates o romanas: Georgia, Times New Roman, Times y serif.
- Fuentes sin remates o de palo seco: Verdana, Arial, Helvetica y sans-serif.
- Fuentes de ancho fijo: Courier New, Courier y monoespaciadas.

3.6.1. TIPOGRAFÍA.

CSS define numerosas propiedades para modificar la apariencia del texto. De ellas, la propiedad básica es **color**, que permite establecer el color de la letra.

Color	Color del texto
Valores	<code><color></code> <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	Depende del navegador
Descripción	Establece el color de letra utilizado para el texto

Aunque el color por defecto del texto depende del navegador, en general, los navegadores principales utilizan el color negro. Para establecer el color la letra de un texto se pueden utilizar cualquiera de las cinco formas que incluye CSS para definir un color. Veamos un ejemplo:

```
h1 {color: #148; }  
p {color: pink; }  
a, span {color: #4163E; }  
div {color: rgb(34, 75, 93); }
```

Otra propiedad básica que define CSS se denomina **font-family** y se utiliza para indicar el tipo de letra con el que se muestra el texto.

font-family	Tipo de letra
Valores	(<nombre_familia> <familia_genérica>) (,<nombre_familia> <familia_genérica>*) inherit
Se aplica a	Todos los elementos
Valor inicial	Depende del navegador
Descripción	Establece el tipo de letra utilizado para el texto

El tipo de letra se puede indicar de dos formas:

- **Mediante el nombre de una familia tipográfica**, es decir, mediante el nombre del tipo de letra, como Arial, Verdana, Garamond, etc.
- **Mediante el nombre genérico de una familia tipográfica**, es decir, el nombre que designa el estilo del tipo de letra, como serif (similar a Times New Romans), sans-serif (tipo Arial), cursive (tipo ComicSans), fantasy (tipo Impact) y monospace (tipo Courier New).

Para evitar el problema común de que el usuario no tenga instalada la fuente que quiere usar el diseñador, CSS permite indicar en la propiedad font-family más de un tipo de letra. El navegador probará en primer lugar con el primer tipo de letra indicado. Si el usuario la tiene instalada, el texto se muestra con ese tipo de letra. Si no, el navegador irá probando con el resto hasta que encuentre alguna fuente instalada en el ordenador del usuario.

Una vez seleccionado el tipo de letra, se puede modificar su tamaño mediante la propiedad **font-size**.

font-size	Tamaño de letra
Valores	<tamaño_absoluto> <tamaño_relativo> <medida> <familia_genérica> <porcentaje> inherit
Se aplica a	Todos los elementos
Valor inicial	medium
Descripción	Establece el tamaño de letra utilizado para el texto

CSS permite utilizar una serie de palabras clave para indicar el tamaño de la letra del texto:

- **Tamaño_absoluto:** xx-small, x-smal, small, medium, large, x-large, xx-large.
- **Tamaño_relativo:** larger y smaller, que toman como referencia el tamaño de la letra del elemento padre.

La propiedad que controla la anchura de la letra es **font-weight**:

font-weight	Anchura de letra
Valores	normal bold bolder lighter 100 200 300 400 500 600 700 800 900 inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece la anchura de letra utilizada para el texto

CSS permite establecer el estilo de la letra de un texto con la propiedad **font-style**:

font-style	Anchura de letra
Valores	normal italic oblique inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el estilo de letra utilizado para el texto

CSS permite otra variación en el estilo del tipo de letra, controlado mediante la **propiedad font-variant**:

font-variant	Anchura de letra
Valores	normal small-caps inherit
Se aplica a	Todos los elementos
Valor inicial	normal

Descripción	Establece el estilo alternativo de letra utilizado para el texto
-------------	--

Por último, CSS proporciona una propiedad tipo “**shorthand**” denominada **font** y que permite indicar de forma directa algunas o todas las propiedades de la tipografía del texto:

font	Tipografía
Valores	((<font-style> <font-variant>) (<font-weight>)? <font-size>(/<line-height>)? caption icon menú message-box small-caption status-bar inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Permite indicar de forma directa todas las propiedades de la tipografía de un texto

3.6.2. APARIENCIA DEL TEXTO.

TEXTOS Y ESPACIOS

CSS también dispone de ciertas propiedades relacionadas con el texto, con objetivos de alineación o tratamiento de espaciados. Veamos algunas de ellas:

Propiedad	Valor	Significado
letter-spacing:	normal TAMAÑO	Espacio entre letras (tracking)
word-spacing:	normal TAMAÑO	Espacio entre palabras
line-height:	normal TAMAÑO	Altura de una línea (interlineado)
text-	TAMAÑO	Indentación de texto

indent:		(sangría)
white-space:	normal nowrap pre pre-line pre-wrap	Comportamiento de los espacios
tab-size:	<u>número de espacios</u> TAMAÑO	Ancho de las tabulaciones
direction:	ltr rtl	Dirección del texto

Las tres primeras propiedades, determinan el espacio en diferentes zonas del texto. Por ejemplo, la primera de ellas, **letter-spacing**, especifica el espacio de separación que hay entre cada letra de un texto, denominado comúnmente **interletraje** o **tracking**.



La propiedad **line-height** especifica la **altura** que tendrá cada línea de texto, una característica que puede facilitar muchísimo la lectura, puesto que un interlineado excesivo puede desorientar al lector, mientras que uno insuficiente puede hacer perder al visitante el foco en el texto.

La propiedad **word-spacing** permite establecer el espacio que hay entre una palabra y otra en un texto determinado, lo que puede facilitar la legibilidad de los textos de una página web y da flexibilidad y control sobre ciertas tipografías.

La propiedad **text-indent** establece un tamaño de indentación (*por defecto, 0*), o lo que es lo mismo, hace un sangrado, desplazando el texto la longitud especificada hacia la derecha (*o izquierda en cantidades negativas*).

Al utilizar **white-space** podemos indicar el comportamiento que tendrán los espacios en blanco en una página web. Por defecto, el valor

es **normal** (*transforma múltiples espacios en blanco en un solo espacio consecutivo*), pero tiene otras opciones posibles:

Valor	Espacios en blanco consecutivos	Contenido
normal	Los espacios se transforman en uno solo.	Se ajusta al contenedor.
nowrap	Los espacios se transforman en uno solo.	Ignora saltos de línea.
Pre	Respetar literalmente los espacios.	Ignora saltos de línea.
pre-wrap	Respetar literalmente los espacios.	Se ajusta al contenedor.
pre-line	Respetar literalmente los espacios y suprime los espacios del final.	Se ajusta al contenedor.

Nota: La diferencia entre **pre-wrap** y **pre-line** es que este último respeta literalmente los espacios que están antes del texto, mientras que si sobran después del texto, los suprime.

Probablemente, a medida que realices diferentes diseños, te encontrarás con la desagradable situación en la que un texto concreto (*por ejemplo, un enlace demasiado largo*) no cabe dentro de un contenedor, por lo que el texto se desborda y provoca efectos no deseados como salirse de su lugar.

Propiedad	Valor	Significado
hyphens:	manual none auto	Indica si se debe dividir las palabras por guiones.

overflow-wrap:	normal break-word anywhere	Dividir palabras para evitar desbordamientos.
line-break:	auto loose normal strict anywhere	Determina como dividir líneas.
word-break:	normal keep-all break-all	Establece si está permitido partir palabras.

OJO: La propiedad **overflow-wrap** sólo funciona cuando **white-space** está establecida a valores que respeten espacios. Además, la propiedad **word-wrap** es un alias de **overflow-wrap** por temas de retrocompatibilidad.

Existen formas de mitigar este problema, como la propiedad **word-break**, **word-wrap** o la propiedad **hyphens**, sin embargo, aún están en fase de desarrollo y su soporte está poco extendido en la actualidad. Aún así, si quieres probar una combinación de varias propiedades que suele dar resultado para paliar este comportamiento, puedes probar lo siguiente:

```
.container {
  hyphens: auto;
  word-wrap: break-word;
  word-break: break-word;
}
```

Por otra parte, la propiedad **tab-size** permite establecer el número de espacios que se mostrarán en el cliente o navegador al representar el carácter de un TAB (*tabulador*), que generalmente se convierten a un espacio en blanco, pero sin embargo son visibles en elementos HTML como **<textarea>** o **<pre>**.

Por último, la propiedad **direction** permite establecer la dirección del texto: de izquierda a derecha (*ltr*, *left to right*) o de derecha a izquierda (*rtl*, *right to left*).

ALINEACIONES

También existen varias propiedades CSS que permiten modificar las diferentes alineaciones de los textos en su conjunto. Veamos un resumen:

Propiedad	Valor	Significado
text-align:	left center right justify	Justificación del texto
text-justify	auto inter-word inter-character none	Método de justificación de textos
text-overflow	clip ellipsis <i>texto</i>	Comportamiento cuando el texto no cabe «[...]»

En el primer caso, se puede establecer los valores **left**, **right**, **center** o **justify** a la propiedad **text-align** para alinear horizontalmente el texto a la izquierda, a la derecha, en el centro o justificar el texto, respectivamente, de la misma forma que lo hacemos en un procesador de texto.

En la propiedad **text-justify** indicamos el tipo de justificación de texto que el navegador realizará: automática (*el navegador elige*), ajustar el espacio entre palabras (*el resultado de ajustar con la propiedad word-spacing*), ajustar el espacio entre par de caracteres (*el resultado de ajustar con la propiedad letter-spacing*) y justificación desactivada.

Por su parte, la propiedad **text-overflow** cambia el comportamiento del navegador cuando detecta que un texto no cabe y se desborda. En ella podemos utilizar los valores **clip**, desbordar el contenedor (*comportamiento por defecto*), **ellipsis**, que muestra el texto «...» cuando no cabe más texto y por último indicar el texto que queremos utilizar en lugar de «...».

Al igual que existe **text-align** para alinear horizontalmente, también existe la propiedad **vertical-align**, que se encarga de la alineación vertical de un elemento, pudiendo establecer como valor las siguientes opciones:

Valor	¿Cómo hace la alineación?
-------	---------------------------

baseline	La base del elemento con la base del elemento padre.
Sub	El elemento como un subíndice.
super	El elemento como un superíndice.
Top	La parte superior del elemento con la parte superior del elemento más alto de la línea.
middle	El elemento en la mitad del elemento padre.
bottom	La parte inferior del elemento con la parte inferior del elemento más bajo de esa línea.
text-top	La parte superior del elemento con la parte superior del texto padre.
text-bottom	La parte inferior del elemento con la parte inferior del texto padre.
<u>tamaño</u>	Sube o baja un elemento el tamaño o porcentaje especificado.

Consejo: Cuidado con **vertical-align**. Esta propiedad puede querer utilizarse para centrar verticalmente un elemento, sin embargo, su utilización es un poco menos intuitiva de lo que en un principio se cree, ya que se debe utilizar para alinear textos respecto a elementos. Para alinear bloques de contenido o crear estructuras de diseño, véase [Flexbox](#).

VARIACIONES

Por último, existen varias propiedades aplicables a los textos para variar su naturaleza. Echemos un vistazo:

Propiedad	Valor	Significado
text-decoration:	none underline overline line-through	Indica el tipo de subrayado (decoración)
text-transform:	none capitalize uppercase lowercase	Transforma a mayús/minús o texto capitalizado

La propiedad **text-decoration** permite establecer **subrayados** (*underline*), subrayados por encima del texto (*overline*) y tachados (*line-through*). Indicando el valor **none** se puede eliminar cualquiera de los formatos anteriores. Es muy utilizado, por ejemplo, para eliminar el subrayado de los textos que tienen un enlace o hipervínculo.

Por último, la propiedad **text-transform** es muy útil para convertir textos a mayúsculas (*uppercase*) o minúsculas (*lowercase*), o incluso capitalizar el texto (*capitalize*), es decir, poner sólo la primera letra en mayúscula, independientemente de como esté escrito en el documento HTML.

3.6.3. LAS FUENTES INCRUSTADAS

LOS FORMATOS DE FUENTE

La regla @font-face de las CSS3 permite «incrustar» fuentes tipográficas en las páginas web. De este modo, los diseñadores no quedan limitados al uso de fuentes genéricas. No obstante, tenga siempre en cuenta estos tres grandes principios:

- La mayoría de las fuentes «profesionales» están sujetas a derechos de uso y difusión.
- Cuando «incrusta» una fuente tipográfica, se incorpora al archivo todo el juego de caracteres, lo que puede hacer que sus páginas pesen considerablemente más.
- Por lo general, el antialiasing (un método que evita que los caracteres se vean pixelados) no se aplica a las páginas web.

Ahora hemos de enfrentarnos al problema de la compatibilidad de los formatos de fuentes tipográficas con las distintas versiones de los navegadores.

He aquí los formatos de fuentes reconocidos por los navegadores:

- **TrueType**: extensión .ttf.
- **OpenType**: extensión .otf.
- **Web Open Font**: extensión .woff.
- **SVG Font**: extensión .svg y .svgz.
- **Embed OpenType**: extensión .eot. Pero cuidado: es un formato propietario de Microsoft y, por lo tanto, solo compatible con Internet Explorer.

Estas son las compatibilidades con los formatos .ttf y .otf (<http://caniuse.com/#feat=ttf>) según el sitio **Can I use**:



Y estas son las compatibilidades con el formato .woff (<http://caniuse.com/#feat=woff>):

LA REGLA @FONT-FACE

Esta es la sintaxis de la regla @font-face:

```
@font-face {
  font-family: "Skia";
  src: url('Skia Regular.ttf');
}
```

Una vez se ha hecho la declaración de la regla @font-face, puede indicar el nombre que quiera para la fuente que se ha de incrustar mediante la propiedad font-family. A continuación, con la propiedad src, especificará la ruta de acceso al archivo de la fuente.

Para aplicarla, basta con indicar al selector deseado el nombre de la fuente mediante la propiedad `font-family`:

```
h1, h2 {
  font-family: Skia;
}
```

He aquí el código completo de este ejemplo (**03_01.html**):

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Fuentes incrustadas</title>
  <meta charset="UTF-8" />
  <style>
    @font-face {
      font-family: "Skia";
      src: url('Skia.ttf');
    }
    h1, h2 {
      font-family: Skia;
    }
  </style>
</head>
<body>
  <h1>Título de mi página</h1>
  <h2>Subtítulo</h2>
  <p>Donec ullamcorper nulla...</p>
</body>
</html>
```

Y he aquí lo que obtenemos:



EL NOMBRE DE LAS FUENTES

Observe que el nombre indicado en la propiedad `font-family` es arbitrario; no tiene por qué corresponder estrictamente al nombre del archivo de la fuente. Se trata de una etiqueta que usted da a esta fuente. Lo que es importante es el nombre del archivo de origen de la fuente en `src`.

He aquí una sintaxis correcta que funciona sin ningún problema:

```
@font-face {
  font-family: "Fuente titulo";
  src: url('Skia.ttf');
```



```
}  
h1, h2 {  
  font-family: 'Fuente titulo', Arial, Helvetica, sans-serif;  
}
```

LAS FUENTES INCRUSTADAS LOCALES

Con la sintaxis anterior, las fuentes se descargan desde el servidor en el equipo del internauta. Pero podemos suponer que algunos internautas dispondrán de esta fuente de forma local, en su ordenador. Vamos a pedir que se utilice la fuente local si dicha fuente se halla en el equipo, especificando local en src.

```
@font-face {  
  font-family: "Skia";  
  src: local("Skia"), url('Skia.ttf');  
}
```

El nombre de la fuente indicado en local debe ser el nombre del archivo en el sistema.

Dicha sintaxis puede interpretarse de la siguiente manera: usar una fuente denominada skia por el diseñador, utilizarla localmente si existe una fuente con este nombre en el sistema del usuario (Skia); en caso contrario, descargar la fuente cuyo nombre de archivo es Skia.ttf.

INDICAR VARIOS FORMATOS

También puede indicar varios formatos de fuentes (si existen) para que la compatibilidad sea más elevada:

```
@font-face {  
  font-family: "Skia";  
  src: url('Skia.ttf') format("truetype"),  
       url('Skia.woff') format("woff");  
}
```

Por supuesto, puede indicar otras fuentes para tener mayor compatibilidad con navegadores antiguos:

```
h1, h2 {  
  font-family: Skia, Arial, Helvetica, sans-serif;  
}
```

LOS ESTILOS TIPOGRÁFICOS

En la declaración de la regla @font-face, puede precisar algunos estilos tipográficos con las propiedades font-weight, font-style y font-variant. Pero no todos los navegadores las reconocen.

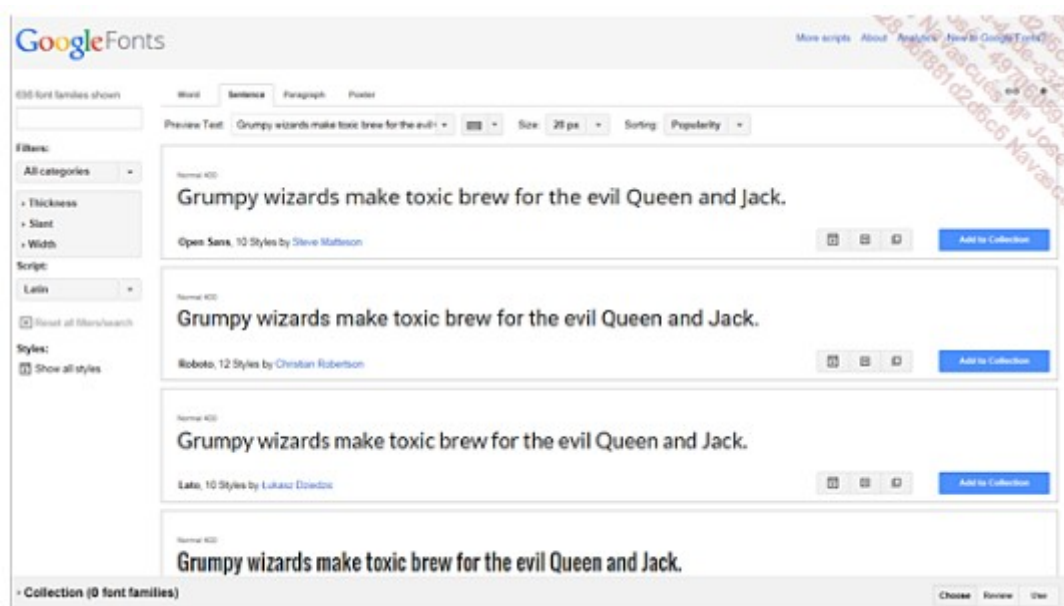
```
@font-face {  
  font-family: "Fuente titulo";  
  src: url('Skia.ttf');
```

```
font-weight: bold;
}
```

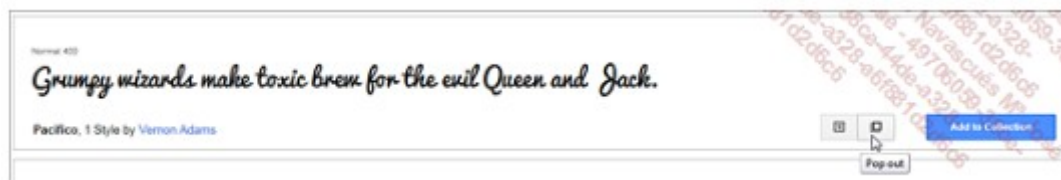
3.6.4. LAS FUENTES EN LÍNEA

LAS GOOGLE FONTS

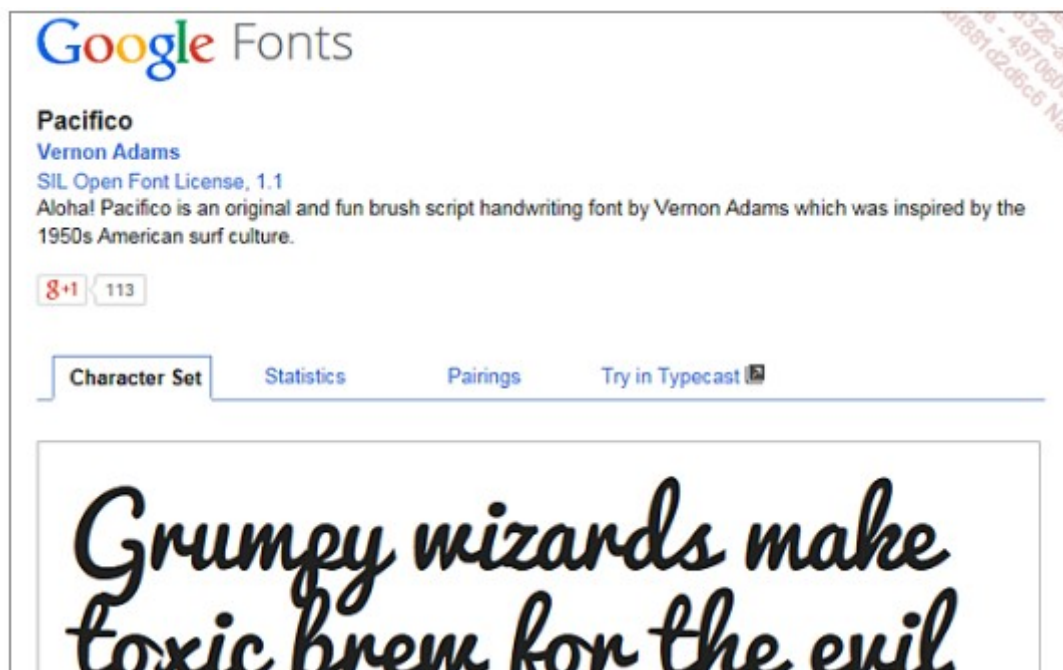
Existen numerosos servicios en línea que ofrecen la posibilidad de utilizar fuentes tipográficas para el Web. El servicio **Google Fonts** es uno de los más conocidos: <http://www.google.com/fonts/>. Google ofrecía 698 fuentes en el momento de escribir este libro (junio de 2015).



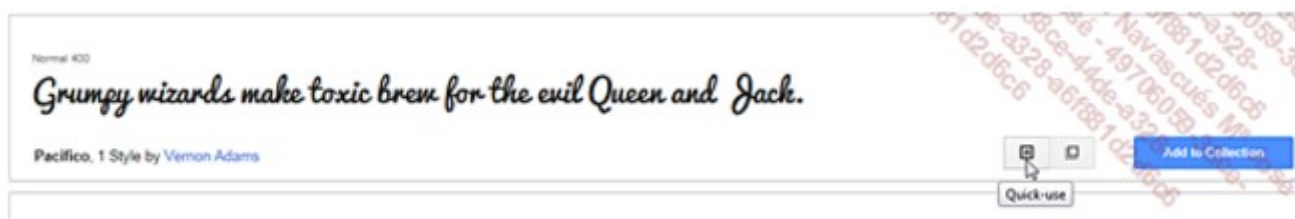
Puede utilizar los filtros que aparecen a la izquierda de la pantalla para elegir una fuente. Si desea ampliar la información sobre una fuente en concreto, haga clic en el botón **Pop out**.



Podrá acceder entonces a toda la información útil:



Para usar la fuente escogida, en la pantalla anterior haga clic en el botón **Quick-use**:

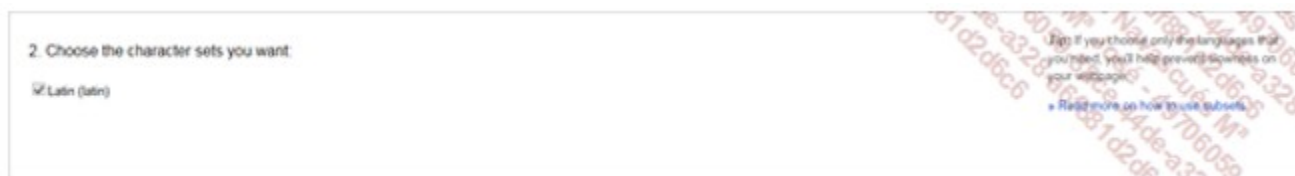


Aparecen cuatro recuadros.

El primero le indica el impacto que tendrá esta fuente cuando se carguen las páginas:



El segundo recuadro permite elegir el alfabeto que se desea utilizar, en caso de que la fuente lo permita. En este ejemplo, la fuente dispone de un solo alfabeto:



El tercer recuadro indica la sintaxis que se ha de utilizar para enlazar esta fuente tipográfica a las páginas web:



El último recuadro indica la sintaxis que se ha de usar en las reglas CSS:



Este es el código completo de una página (**03_02.html**) que utiliza esta fuente tipográfica de Google:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Las fuentes de Google</title>
  <meta charset="UTF-8" />
  <link href='http://fonts.googleapis.com/css?family=Pacifico'
rel='stylesheet' type='text/css'>
  <style>
    h1, h2 {
      font-family: 'Pacifico', cursive;
    }
  </style>
</head>
<body>
  <h1>Título de mi página</h1>
  <h2>Subtítulo</h2>
  <p>Donec ullamcorper nulla...</p>
</body>
</html>
```

Este es el resultado que se obtiene:



3.7. ENLACES.

CSS permite aplicar diferentes estilos a un mismo enlace en función de su estado. De esta forma, es posible cambiar el aspecto de un enlace cuando, por ejemplo, el usuario pasa el ratón por encima o cuando el usuario pincha sobre ese enlace.

Con los atributos `id` o `class` no es posible aplicar distintos estilos a un mismo elemento en función de su estado. Por ello, CSS introduce un nuevo concepto llamado pseudo-clases y, en concreto, define cuatro diferentes:

:link, aplica estilos a los enlaces que apuntan a páginas o recursos que aún no han sido visitados por el usuario.

:visited, aplica estilos a los enlaces que apuntan a recursos que han sido visitados anteriormente por el usuario.

:hover, aplica estilos al enlace sobre el que el usuario ha posicionado el puntero del ratón.

:active, aplica estilos al enlace que está pinchando el usuario. Los estilos sólo se aplican desde que el usuario pincha el botón del ratón hasta que lo suelta, por lo que suelen ser unas pocas décimas de segundo.

El siguiente ejemplo muestra cómo ocultar el subrayado cuando el usuario pasa el ratón por encima de cualquier enlace de la página:

```
a:hover {text-decoration: none; }
```

Si se definen varias pseudo-clases diferentes sobre un mismo enlace, habrá que hacerlo en el orden en que se han descrito para que no se produzcan colisiones de estilos.

3.8. IMÁGENES

CSS permite establecer la anchura y la altura de una imagen mediante las propiedades `width` y `height`, respectivamente, independientemente de su anchura y altura real. Veamos un ejemplo:

```
#destacada {  
width: 120px;  
height: 250px;  
}  

```

Usar anchuras y alturas diferentes a las reales produce deformaciones en las imágenes y el resultado estético es muy desagradable.

Por otra parte, establecer la anchura y altura para cada imagen mediante CSS es una práctica poco recomendable puesto que produce una sobrecarga de estilos. Por este motivo, aunque es una solución que no respeta la separación entre contenidos y presentación, se recomienda establecer la anchura y la altura de las imágenes mediante atributos de la etiqueta ``.

Cuando una imagen forma parte de un enlace, los navegadores muestran por defecto un borde grueso azul alrededor de ella. Por tanto, una de las reglas más utilizadas en los archivos CSS es la que elimina los bordes de las imágenes con enlace:

```
img { border: none; }
```

3.9. LISTAS

Viñetas personalizadas.

Por defecto, los navegadores muestran los elementos de las listas no ordenadas con una viñeta formada por un pequeño círculo de color negro y los elementos de las listas ordenadas con la numeración decimal utilizada por la mayoría de los países.

CSS define la propiedad `list-style-type` para controlar el tipo de viñeta que muestra una lista:

list-style-type	Tipo de viñeta
Valores	disc circle square decimal decimal-leading-zero lower-roman upper-roman lower-greek lower-latin upper-latin armenian georgian lower-alpha upper-alpha none inherit
Se aplica a	Elementos de una lista
Valor inicial	Disc

Descripción	Permite establecer el tipo de viñeta mostrada para una lista

El valor none permite mostrar una lista sin viñetas, números o letras, por lo que es un valor muy utilizado en los menús de navegación de las páginas.

El resto de los valores se dividen en tres grupos:

- **Valores gráficos:** muestran las viñetas como círculos rellenos (circle), círculos vacíos (disc) o cuadrados (square).
- **Valores numéricos:** pueden ser decimal, decimal-leading-zero, lower-roman, upper-roman, armenian y georgian.
- **Valores alfanuméricos:** se controlan mediante lower-latin, upper-latin, lower-alpha, upper-alpha y lower-greek.

La propiedad **list-style-position** permite controlar la colocación de las viñetas:

list-style-position	Posición de la viñeta
Valores	inside outside inherit
Se aplica a	Elementos de una lista
Valor inicial	outside
Descripción	Permite establecer la posición de la viñeta de cada elemento de una lista

Para personalizar el aspecto de las viñetas se emplea la propiedad **list-style-image** que permite mostrar una imagen propia en vez de una viñeta automática.

list-style-image	Imagen de la viñeta
Valores	<url> none inherit
Se aplica a	Elementos de una lista
Valor inicial	none
Descripción	Permite reemplazar las viñetas automáticas por

	una imagen personalizada
--	--------------------------

CSS define una propiedad de tipo “shorthand” que permite establecer todas las propiedades de una lista de forma directa. El siguiente ejemplo indica que no se debe mostrar ni viñetas automáticas ni personalizadas: `ul { list-style: none; }`

Menús.

Las listas HTML se suelen emplear, además de para su función natural, para la creación de menús de navegación verticales y horizontales.

A continuación se muestra una transformación de una lista sencilla de enlaces de un menú de navegación:

```
<ul class="menu">
  <li><a href="#" >Elemento 1</a></li>
  <li><a href="#" >Elemento 2</a></li>
  <li><a href="#" >Elemento 3</a></li>
</ul>
```

La transformación de la lista en un menú se podría realizar de esta forma:

`ul.menu {`

```
width: 180px;
list-style: none;
margin: 0;
padding: 0;
border: 1px solid
```

Definir anchura

Eliminar viñetas, márgenes y espaciados

Enmarcar la lista

`#3A3A3A;`

`ul.menu li{`

```
border-bottom: 1px solid
border-top: 1px solid #BBB;
background: #F4F4F4;
}
```

Establecer color de fondo y bordes a cada elemento del menú

`#3A3A3A;`

`ul.menu li a{`

```
padding: .2em 0 .2em .5em;
display: block;
text-decoration: none;
color: #222;
}
```

Aplicar estilos a los enlaces:

añadir relleno.
mostrarlos como bloques
modificar colores y decoración

3.10. TABLAS.

CSS permite seleccionar el modelo de borde de las celdas de una tabla mediante la propiedad **border-collapse**:

border-collapse	Fusión de bordes
Valores	collapse separate inherit
Se aplica a	Todas las tablas
Valor inicial	separate
Descripción	Define el mecanismo de fusión de los bordes de las celdas adyacentes de una tabla

El modelo collapse fusiona de forma automática los bordes de las celdas adyacentes, mientras que el modelo separate fuerza a que cada celda muestre sus cuatro bordes.

Si se opta por el modelo **separate**, se puede utilizar la propiedad **border-spacing** para controlar la separación entre los bordes de cada celda.

border-spacing	Espaciado entre bordes
Valores	<medida><medida>? inherit
Se aplica a	Todas las tablas
Valor inicial	0
Descripción	Establece la separación entre los bordes de las celdas adyacentes de una tabla

Si solamente se indica una medida, se asignará a la separación horizontal y vertical. Si se indican dos medidas, la primera corresponderá a la separación horizontal y la segunda a la separación vertical.

Cuando se usa el modelo de bordes **separate**, se puede establecer el tratamiento que reciben las celdas vacías de una tabla mediante la propiedad **empty-cells**:

empty-cells	Tratamiento de las celdas vacías
Valores	show hide inherit
Se aplica a	Celdas de una tabla
Valor inicial	Show
Descripción	Define el mecanismo utilizado para el tratamiento de las celdas vacías de una tabla

Una celda vacía es aquella que no tiene ningún contenido, ni siquiera un espacio en blanco o un ` `. El valor `hide` indica que las celdas vacías no se deben mostrar.

El título de las tablas se establece mediante el elemento `<caption>` que, por defecto, se muestra encima de los contenidos de la tabla. La propiedad `caption-side` permite controlar la posición del título de la tabla:

caption-side	Posición del título de la tabla
Valores	top bottom inherit
Se aplica a	Los elementos <code>caption</code>
Valor inicial	Top
Descripción	Establece la posición del título de la tabla

El valor `bottom` indica que el título de la tabla se debe mostrar después de los contenidos de la misma. Su alineación horizontal se controla mediante la propiedad `text-align`.

Se puede aplicar la pseudo-clase `:hover` para que el color de una fila varíe cuando el usuario pasa el ratón por encima de ella. En el ejemplo se muestra cómo se haría:

```
table tr:hover{ background: #FFFF66; }
```

Desafortunadamente, Internet Explorer 6 y sus versiones anteriores no soportan esta pseudo-clase en elementos que no sean enlaces.

3.11. FORMULARIOS

CSS permite modificar el aspecto de los controles de un formulario. Por ejemplo, podemos cambiar el estilo de un botón:

```
.boton {
```

```
border: 0;
padding: 0;
background-color: transparent;
color: red;
border-bottom: 1px solid red;
}
```

Por defecto, los campos de texto de los formularios no incluyen ningún espacio de relleno, por lo que el texto introducido por el usuario aparece pegado a los bordes del cuadro del texto. Añadiendo un pequeño padding a cada elemento `<input>`, se mejora notablemente el aspecto del formulario. Por ejemplo, podríamos escribir:

```
form.elegante input { padding: .2em; }
```

CSS permite resaltar el campo en el que el usuario está introduciendo datos mediante la pseudo-clase `:focus`. A continuación, se muestra un ejemplo:

```
input :focus
{
    border: 2px solid #000;
    background: #F3F3F3;
}
```

4.11.1. DAR FORMATO

Para dar formato a los formularios, puede utilizar las propiedades CSS aplicables al texto, así como las propiedades de los contenedores (ancho, contorno, color de fondo...). No existe ninguna propiedad ni ningún módulo específicamente dedicado a los formularios.

Vamos a utilizar, sencillamente, selectores, propiedades y atributos que se adaptan bien a los formularios.

4.11.2. REDIMENSIONAR UN CAMPO

La propiedad `resize` forma parte del módulo **Basic User Interface Module Level 3 (CSS3 UI)**, que está como **Candidate Recommendation, 7 July 2015** (<http://www.w3.org/TR/css3-ui/>). Puede usarse, por ejemplo, en un campo de formulario cuando el diseñador desee que el usuario pueda modificar el tamaño de dicho campo.

Propiedad: `resize`

Valor: `none` | `both` | `horizontal` | `vertical`

Valor inicial: `none`

Se aplica a: los elementos visibles (propiedad overflow)

Herencia: no

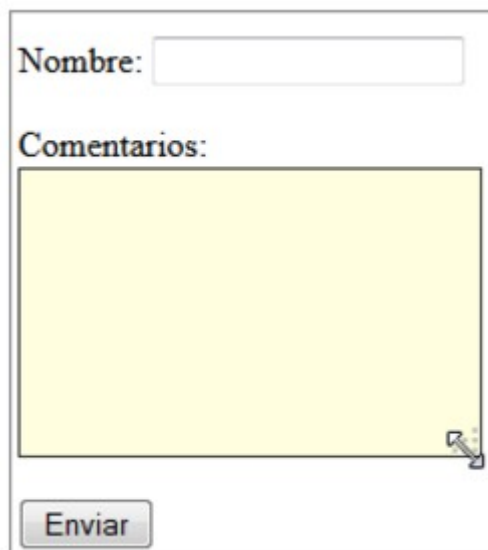
Los valores que se pueden utilizar:

- **none:** sin posibilidad de redimensionamiento.
- **both:** redimensionable horizontal y verticalmente.
- **horizontal:** redimensionable horizontalmente.
- **vertical:** redimensionable verticalmente.

He aquí un ejemplo sencillo (**06_03.html**):

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Redimensionar un campo</title>
  <meta charset="UTF-8" />
  <style>
    #comentario {
      width: 200px;
      height: 90px;
      border: 1px solid #333;
      background-color: lightyellow;
      resize: both;
    }
  </style>
</head>
<body>
<form id="test" method="#" action="#">
  <p>
    <label for="nombre">Nombre: </label>
    <input type="text" id="nombre" />
  </p>
  <p>
    <label for="comentario">Comentarios: </label>
    <br />
    <textarea name="comentario" id="comentario"></textarea>
  </p>
  <p>
    <input type="submit" name="envio" id="envio" value="Enviar" />
  </p>
</form>
</body>
</html>
```

Esto es lo que obtenemos: el usuario puede cambiar las dimensiones del campo **Comentarios**.

Un formulario web con un campo de texto etiquetado como 'Nombre:', un área de texto grande etiquetada como 'Comentarios:' y un botón 'Enviar'.

4.11.3. PSEUDOCASES PARA FORMULARIOS

Las CSS3 proponen una serie de pseudocases muy útiles para los objetos de formulario; permiten resaltar los objetos activos, desactivados y marcados en un formulario. De este modo, el usuario puede saber qué ha hecho él, qué está activo y qué inactivo.

Estas nuevas pseudocases son: **:enabled**, **:disabled** y **:checked**.

En este ejemplo (**06_04.html**), resaltamos el uso de diferentes campos de formulario.

El primer selector se dirige a la etiqueta (label) de la casilla de verificación (**#acuerdo**) para saber si está marcada (checked). Lo que queremos resaltar es, efectivamente, la etiqueta de la casilla de verificación, y no la casilla de verificación propiamente dicha; de ahí el uso de un selector adyacente.

El segundo selector se dirige al botón de opción marcado, también en este caso para detectar que lo está.

Los otros dos estilos simplemente sirven para atribuir un fondo de color según el estado del campo.

Este es el código de la página:

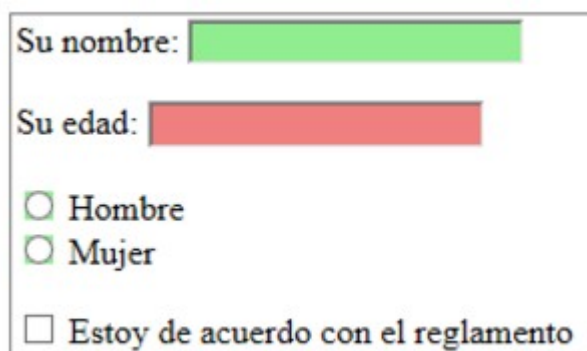
```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Pseudocases para los formularios</title>
  <meta charset="UTF-8" />
  <style>
    #acuerdo:checked+label {
      background-color: gold;
      font-weight: bold;
    }
  </style>
</head>
<body>
  <div>
    <input checked="" type="checkbox"/>
    <label>Acuerdo</label>
  </div>
  <div>
    <input checked="" type="radio"/>
    <label>Opción marcada</label>
  </div>
  <div>
    <input type="text" value="Nombre:">
  </div>
  <div>
    <div>Comentarios:</div>
    <div></div>
  </div>
  <div>
    <input type="button" value="Enviar"/>
  </div>
</body>
</html>
```

```

        #hombre:checked+label, #mujer:checked+label {
            font-style: italic;
        }
        :enabled {
            background-color: lightgreen;
        }
        :disabled {
            background-color: lightcoral;
        }
    }
</style>
</head>
<body>
<form id="form1" method="#" action="#">
<p>
    <label for="nombre">Su nombre: </label>
    <input type="text" id="nombre" />
</p>
<p>
    <label for="edad">Su edad: </label>
    <input type="text" id="edad" disabled="disabled"/>
</p>
<p>
    <input type="radio" name="sexo" id="hombre" value="hombre" />
    <label for="hombre">Hombre</label><br/>
    <input type="radio" name="sexo" id="mujer" value="mujer" />
    <label for="mujer">Mujer</label>
</p>
<p>
    <input type="checkbox" id="acuerdo" />
    <label for="acuerdo">Estoy de acuerdo con el reglamento</label>
</p>
</form>
</body>
</html>

```

Esto es lo que aparece cuando se carga la página:



Su nombre:

Su edad:

☒ Hombre

☐ Mujer

☐ Estoy de acuerdo con el reglamento

Esto es lo que se ve cuando el usuario marca el botón de opción **Hombre**.

Su nombre:

Su edad:

☐ *Hombre*

☐ *Mujer*

☐ *Estoy de acuerdo con el reglamento*

Y esto es lo que se obtiene cuando el usuario marca la casilla de verificación.

Su nombre:

Su edad:

☐ *Hombre*

☐ *Mujer*

☒ **Estoy de acuerdo con el reglamento**

Aún existe otro estado para los botones de opción y las casillas de verificación: se trata del estado «indeterminado»: `:indeterminate`, que indica que el objeto no está ni marcado ni desmarcado.

4.11.4. LOS CAMPOS REQUERIDOS Y LOS OPCIONALES

Si lo desea, puede resaltar los campos que sean obligatorios u opcionales con las pseudoclasas **`:required`** y **`:optional`**.

Los campos que son obligatorios, es decir, aquellos en los que se exige introducir o marcar algo, deben disponer del atributo booleano `required`. Los que no sean obligatorios serán opcionales de forma predeterminada, sin que resulte necesario especificarlo.

He aquí un ejemplo (**06_05.html**):

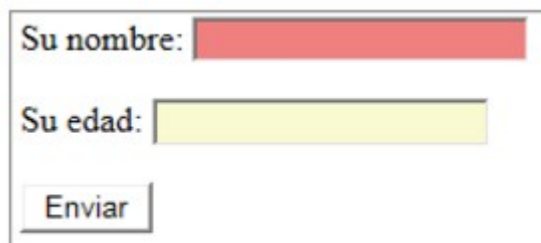
```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Los campos requeridos y los opcionales</title>
  <meta charset="UTF-8" />
  <style>
    input:required {
      background-color: lightcoral;
    }
    input:optional {
      background-color: lightgoldenrodyellow;
    }
    input#envio {
```

```

        background: none;
    }
</style>
</head>
<body>
<form id="form1" method="#" action="#">
    <p>
        <label for="nombre">Su nombre: </label>
        <input type="text" name="nombre" id="nombre" required />
    </p>
    <p>
        <label for="edad">Su edad: </label>
        <input type="text" name="edad" id="edad" />
    </p>
    <p>
        <input type="submit" id="envio" value="Enviar" />
    </p>
</form>
</body>
</html>

```

Esto es lo que se obtiene:



4.11.5. DAR FORMATO AL «FOCO»

La pseudoclase `:focus` sirve para resaltar el campo que se está usando mediante la adición de un contorno suplementario que proporciona la propiedad `outline`. Las CSS3 añaden una nueva propiedad: `outline-offset`, que define la distancia entre el límite de la caja y el contorno de resalte (módulo **Basic User Interface Module Level 3 (CSS3 UI)**, en **Candidate Recommendation, 7 July 2015**).

He aquí un ejemplo muy sencillo (**06_06.html**):

```

<!DOCTYPE html>
<html lang="es">
<head>
    <title>El foco</title>
    <meta charset="UTF-8" />
    <style>
        input:focus {
            outline: solid 3px green;
            outline-offset: 2px;
        }
        input#envio {
            background: none;
        }
    </style>

```

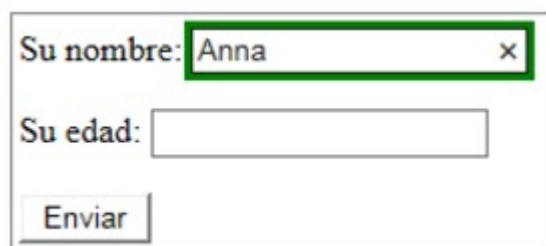


```

</head>
<body>
<form id="form1" method="#" action="#">
  <p>
    <label for="nombre">Su nombre: </label>
    <input type="text" name="nombre" id="nombre" required />
  </p>
  <p>
    <label for="edad">Su edad: </label>
    <input type="text" name="edad" id="edad" />
  </p>
  <p>
    <input type="submit" id="envio" value="Enviar" />
  </p>
</form>
</body>
</html>

```

He aquí lo que aparece cuando el usuario hace clic en un campo:



4.11.6. VALIDAR LA INFORMACIÓN INTRODUCIDA

Se trata de resaltar la validación de los campos en un formulario. En este ejemplo (**06_07.html**), los dos primeros campos son obligatorios debido al uso del atributo `required`. El tercer campo incorpora un patrón que deben seguir los datos que se introduzcan (`pattern`). En estos tres campos se mostrará un círculo rojo cuando la información introducida no sea válida, y un círculo verde cuando sí lo sea.

Veamos el código de este ejemplo:

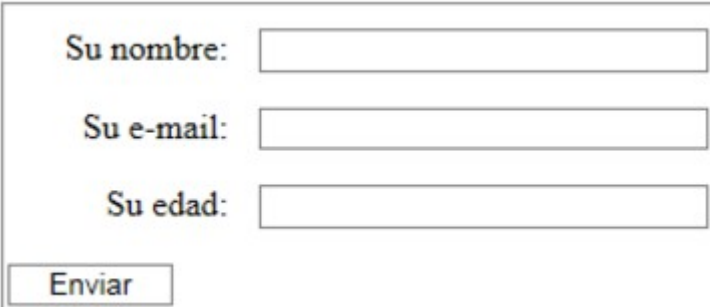
```

<!DOCTYPE html>
<html lang="es">
<head>
  <title>Validación de un formulario</title>
  <meta charset="UTF-8" />
  <style>
    input {
      border: solid 1px gray;
      width: 200px;
    }
    label, input{
      display: inline-block;
    }
    label {
      width: 100px;
      text-align: right;
    }

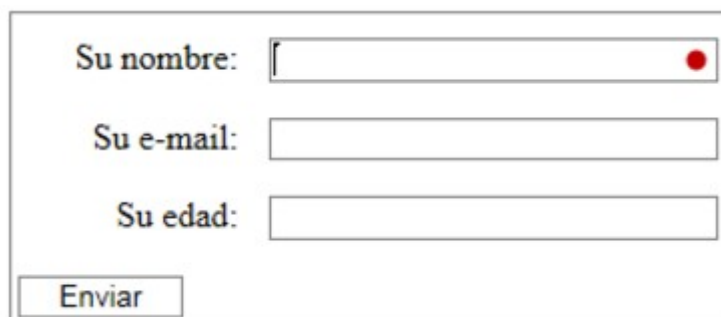
```

```
        margin-right: 10px;
    }
    input[type=submit]{
        width: 75px;
        background: none;
    }
    input:focus:invalid {
        background: url(rojo.png) no-repeat 98% center;
    }
    input:focus:valid {
        background: url(verde.png) no-repeat 98% center;
    }
}
</style>
</head>
<body>
<form id="inscription" method="#" action="#">
    <p>
        <label for="nombre">Su nombre: </label>
        <input type="text" id="nombre" required />
    </p>
    <p>
        <label for="email">Su e-mail: </label>
        <input type="email" id="email" required />
    </p>
    <p>
        <label for="edad">Su edad: </label>
        <input type="text" id="edad" pattern="\d{2}" />
    </p>
    <p>
        <input type="submit" id="envio" value="Enviar" />
    </p>
</form>
</body>
</html>
```

Esto es lo que aparece cuando se carga la página:

A screenshot of a web form. It contains three rows of labels and input fields: 'Su nombre:' followed by a text input field, 'Su e-mail:' followed by an email input field, and 'Su edad:' followed by a text input field with a pattern attribute. At the bottom left, there is a button labeled 'Enviar'.

Esto es lo que se muestra cuando el usuario hace clic en el primer campo, **Su nombre:**



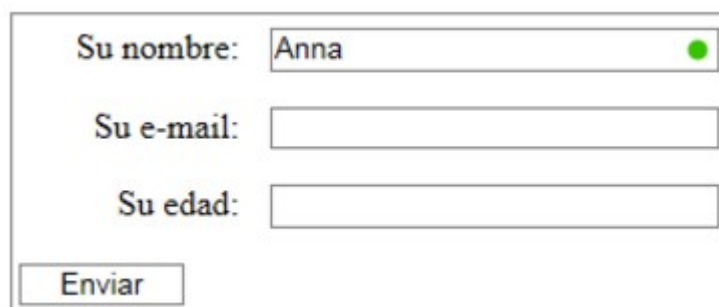
Su nombre: ●

Su e-mail:

Su edad:

El campo no es válido, ya que no se ha introducido nada y es obligatorio rellenar este campo.

Veamos lo que aparece cuando el usuario escribe algo en el primer campo, **Su nombre**:



Su nombre: ●

Su e-mail:

Su edad:

El campo tiene contenido, por lo que la validación es correcta.

Esto es lo que aparece cuando el usuario únicamente introduce una cifra en el campo **Su edad**:



Su nombre:

Su e-mail:

Su edad: ●

El dato no es válido, ya que tiene únicamente una cifra.

Y esto es lo que aparece cuando el usuario introduce dos cifras en el campo **Su edad**:

Formulario de registro:

Su nombre:

Su e-mail:

Su edad: ●

El dato es válido, ya que tiene dos cifras.

3.12. CURSOR

CSS no permite modificar los elementos propios del navegador o de la interfaz de usuario del sistema operativo. Sin embargo, el puntero del ratón es una excepción, ya que se puede modificar mediante la propiedad cursor:

Cursor	Puntero del ratón
Valores	((<url> ,)* (auto crosshair default pointer move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text wait help progress)) inherit
Se aplica a	Todos los elementos
Valor inicial	auto
Descripción	Permite personalizar el puntero del ratón

Se pueden indicar varias URL para que CSS intente cargar varias imágenes. Si la primera de ellas no se carga o no la soporta el navegador, se pasa a la siguiente y, así, sucesivamente, hasta que se pueda cargar alguna imagen.

Se puede ver un ejemplo de cada uno de los punteros y la compatibilidad con los diferentes navegadores en la siguiente página: <http://www.echoecho.com/csscursors.htm>.

4. PROPIEDADES CSS3

4.1. LOS MÓDULOS CSS3

Las CSS tienen como objetivo separar la estructura de la página (con su contenido, creado con HTML) del diseño y el formato de la propia página.

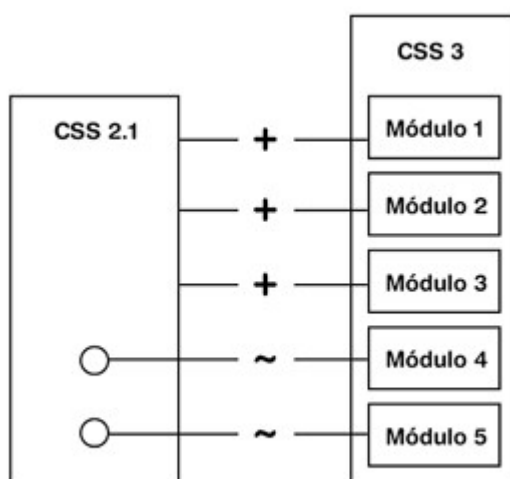
Las Cascading Style Sheets Level 1 se publicaron como Recommendation (documento finalizado por el W3C) el 17 de diciembre de 1996 y se revisaron el 11 abril de 2008: <http://www.w3.org/TR/CSS1/>. Las CSS 2.1 se han publicado como Recommendation el 7 de junio de 2011: <http://www.w3.org/TR/CSS21/>.

Cada recomendación de CSS (1, 2 y 2.1) implicó un enorme trabajo, desde los borradores (Working Draft) hasta el documento finalizado (Recommendation). La recomendación final es un documento único con un número impresionante de páginas.

Para la versión siguiente de las CSS, la versión 3, el W3C decidió cambiar su modo de trabajo.

Primer punto importante: en lugar de hacer una recomendación «enorme», monolítica y de difícil mantenimiento, el W3C ha trabajado de manera modular. Las propiedades se han reagrupado en módulos funcionales independientes que evolucionan a su propio ritmo. De este modo, los navegadores han podido probar e implementar las nuevas propiedades de forma progresiva, módulo a módulo.

Segundo punto, cuya comprensión resulta muy importante: las CSS3 no reemplazan las CSS 2.1. Las CSS3 son una continuación, un complemento, una prolongación de las CSS 2.1. Las CSS 2.1 se mantienen plenamente vigentes y usables. Lo único que hacen las CSS3 es añadir nuevas propiedades y redefinir algunas de las que ya existían desde las CSS 2.1.



4.2. ETAPAS EN LA CONCEPCIÓN DE LAS CSS3

Cada módulo pasa por una serie de etapas perfectamente estandarizadas por el W3C.

La primera etapa se centra en el borrador de trabajo, el **Working Draft (WD)**. En este documento se recogen las propuestas de propiedades para que toda la

comunidad (público, empresas, navegadores...) puedan probar su implementación.

A continuación, se entra en la etapa **Last Call (LC)**. El W3C considera que su trabajo es técnicamente satisfactorio, anuncia la fecha de finalización de las pruebas y toma en cuenta los comentarios de los grupos de trabajo.

Después viene la **Candidate Recommendation (CR)**. El W3C ha revisado prolongadamente su copia, de acuerdo con los grupos de trabajo y con las pruebas de puesta en práctica. El documento casi está finalizado.

La última etapa es la de **Recommendation (REC)**. El W3C publica el documento, que desempeña el papel de estándar y al cual deben referirse todos los navegadores si quieren respetar estos estándares.

Además, es preciso entender que las CSS3 están «vivas», evolucionan, pasan del Estado **Working Draft** a la **Recommendation**. Cada módulo evolucionará a su propia velocidad. Por lo tanto, conviene referirse con frecuencia a la página de la evolución de las CSS3 en el sitio del W3C, la página **Current Work**: <http://www.w3.org/Style/CSS/current-work>

En la zona **Table of specifications**, puede observar la evolución de los diversos módulos, organizados en varias categorías. Este es el estado que presentaban en mayo de 2015:

- **Completed**: módulos terminados, en **Recommendation**.

Completed	Current	Upcoming	Notes	
CSS Snapshot 2010	NOTE		Latest stable CSS	10
CSS Snapshot 2007	NOTE			10
CSS Color Level 3	REC	REC	See Errata	10
CSS Namespaces	REC	REC		10
Selectors Level 3	REC	REC		10
CSS Level 2 Revision 1	REC	REC	See Errata	10
CSS Level 1	REC		Unmaintained, see Snapshot	10
CSS Print Profile	NOTE			10
Media Queries	REC	REC		10
CSS Style Attributes	REC	REC		10

- **Testing**: módulos en estado **Candidate Recommendation**.

Testing	Current	Upcoming	Notes
CSS Backgrounds and Borders Level 3	CR	PR	
CSS Conditional Rules Level 3	CR	CR	
CSS Image Values and Replaced Content Level 3	CR	PR	
CSS Multi-column Layout	CR	CR	
CSS Speech	CR	PR	
CSS Values and Units Level 3	CR	PR	
CSS Flexible Box Layout	LC	CR	
CSS Text Decoration Module Level 3	CR	PR	
CSS Cascading and Inheritance Level 3	CR	PR	
CSS Fonts Level 3	CR	PR	
CSS Writing Modes Level 3	CR	PR	
CSS Shapes	CR	PR	
CSS Masking	CR	PR	
CSS Mobile Profile 2.0	CR	NOTE	Status unknown

- **Refining:** módulos que se están redefiniendo.

Refining	Current	Upcoming	Notes
CSS Animations	WD	WD	
Web Animations 1.0	WD	WD	
CSS Counter Styles Level 3	CR	PR	
CSS Text Level 3	LC	CR	
CSS Fragmentation Level 3	WD	WD	
CSS Transforms	WD	WD	
CSS Transitions	WD	LC	
Cascading Variables	WD	LC	
Compositing and Blending	CR	PR	
CSS Syntax Level 3	CR	PR	
CSS Basic User Interface Level 3	WD	CR	
CSS Will Change	FPWD	WD	

- **Revising:** módulos que se están reestructurando.

Revising	Current	Upcoming	Notes
CSS Box Alignment Module Level 3	WD	WD	
CSS Grid Layout	WD	WD	
CSS Paged Media Level 3	WD	LC	
CSSOM View	WD	WD	
Selectors Level 4	WD	WD	

- **Exploring:** módulos que aún están en desarrollo.

Exploring	Current	Upcoming	Notes	
CSS Backgrounds and Borders Level 4		WD		IO
CSS Device Adaptation	FPWD	WD		IO
CSS Exclusions	WD	WD		IO
Filter Effects	WD	WD		IO
CSS Generated Content for Paged Media	WD	WD		IO
CSS Page Floats		WD		IO
CSS Template Layout	NOTE	NOTE		IO
CSS Intrinsic & Extrinsic Sizing Module Level 3	FPWD	WD		IO
CSS Line Grid	WD	WD		IO
CSS Lists Level 3	WD	WD		IO
CSS Positioned Layout Level 3	WD	WD		IO
CSS Regions	WD	WD		IO
CSS Ruby	WD	WD		IO
CSS Tables Level 3		WD	Inactive	IO
CSS Object Model	WD	WD		IO
CSS Overflow	FPWD	WD		IO
CSS Font Loading	WD	WD		IO
CSS Display	WD	WD		IO
CSS Scoping	FPWD	WD		IO
Media Queries level 4	FPWD	WD		IO
Non-element Selectors	FPWD	WD		IO
Geometry Interfaces Module Level 1	WD	WD		IO
CSS Inline Layout Module Level 3	FPWD	WD		IO
CSS Pseudo-Elements Module Level 4	FPWD	WD		IO
Motion Path Module Level 1	FPWD	WD		IO
CSS Cascading and Inheritance Level 4	FPWD	WD		IO
CSS Scroll Snap Points Module Level 1	FPWD	WD		IO

- **Rewriting:** módulos que se están reescribiendo por completo.

Rewriting	Current	Upcoming	Notes	
CSS Basic Box Model Level 3	WD	WD	Dangerously outdated; see CSS 2.1	IO
CSS Generated Content Level 3	FPWD	WD	Severely outdated	IO
CSS Line Layout Level 3	FPWD	WD	Severely outdated	IO

- **Abandoned:** módulos que sencillamente se han abandonado.

Abandoned	Current	Upcoming	Notes	
The CSS 'Reader' Media Type	NOTE			IO
CSS Presentation Levels	NOTE			IO
CSS TV Profile 1.0	NOTE			IO
CSS Marquee	NOTE			IO
Behavioral Extensions to CSS	NOTE			IO
CSS Hyperlink Presentation	NOTE			IO
Fullscreen	NOTE			IO

Esta es la razón de que sea imprescindible consultar de forma regular este sitio, para estar al corriente de la evolución de los módulos CSS3.

4.3.LOS PREFIJOS DE LOS NAVEGADORES

La concepción de las CSS3 en módulos independientes supone para los navegadores un trabajo considerable de implementación de las nuevas propiedades en los motores de renderizado. Y, además, mientras los módulos no pertenezcan a la categoría **Candidate Recommendation**, las propiedades pueden cambiar, lo que complica todavía más el trabajo de los navegadores.

Con objeto de trabajar a mayor velocidad, el W3C propone que los navegadores utilicen un prefijo específico para cada uno de ellos, con la finalidad de probar la implementación de las nuevas propiedades. En cuanto la especificación llegue al nivel **CR, Candidate Recommendation**, los prefijos resultaran inútiles.

He aquí estos prefijos (designación según proveedor):

- -moz-: para el motor de renderizado Gecko que utiliza Mozilla Firefox.
- -webkit-: para el motor de renderizado WebKit que utilizan Apple Safari y Google Chrome, con sus bifurcaciones.
- -o-: para el motor de renderizado de Opera.
- -ms-: para el motor de renderizado de Microsoft Internet Explorer.
- -khtml-: para el motor de renderizado KHTML que usan varios navegadores bajo Linux.

Tomemos un ejemplo preciso. Supongamos que queremos utilizar la propiedad border-radius, que permite crear ángulos redondeados en las etiquetas <div>, por ejemplo.

He aquí la sintaxis que se debe utilizar para que la reconozcan todos los navegadores:

```
header {  
  -moz-border-radius: 10px;  
  -webkit-border-radius: 10px;  
  -o-border-radius: 10px;  
  -ms-border-radius: 10px;  
  -khtml-border-radius: 10px;  
  border-radius: 10px;  
}
```

Las primeras líneas se han escrito respectivamente para cada motor de renderizado que hemos mencionado en la lista previa. La última línea corresponde al uso de la propiedad estándar para todos los navegadores que reconocerán más adelante esta propiedad, en el momento en que el W3C la considere finalizada.

El orden de las líneas es importante. Al ubicar la propiedad «estándar» en la última posición, se asegura de que esta se impone a las líneas precedentes. Es decir, primero debe indicar las propiedades con los prefijos (las cuales pueden evolucionar) y después finalizar con la propiedad «oficial».

Esta sintaxis puede parecer pesada, pero facilita la mejora de la evolución y la portabilidad de las propiedades que aún no son oficiales.

Para ayudarle a gestionar fácilmente los prefijos, existen numerosos servicios en línea:

- **Net Tuts Prefixr:** <http://prefixr.com/>
- **prefixMyCSS:** <http://prefixmycss.com/>
- **Online CSS Prefixer:** <http://www.css-prefix.com>

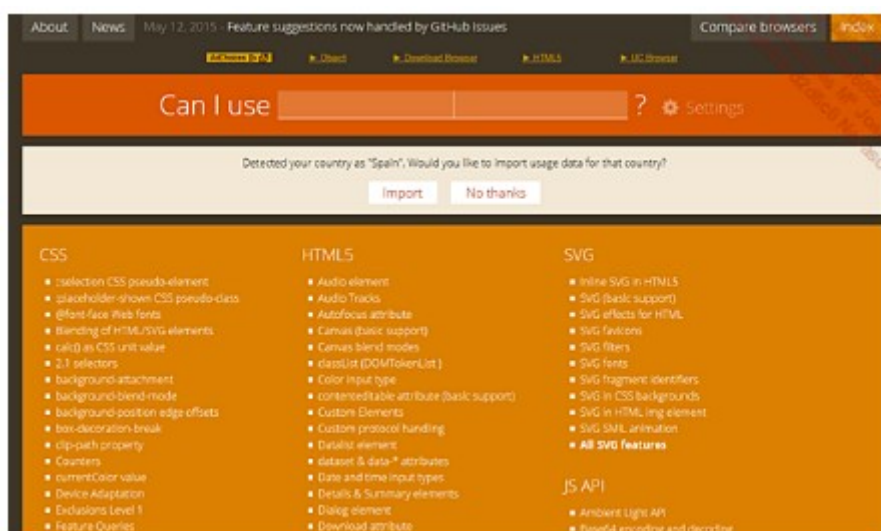
También puede utilizar un prefijador en JavaScript: <http://leaverou.github.io/prefixfree/>

Pero, aun en este caso, debe realizar un trabajo de verificación. Le corresponde a usted comprobar si los prefijos deben usarse con una propiedad en concreto. Para ello, consulte el sitio **Current Work** del W3C, <http://www.w3.org/Style/CSS/current-work#CSS3>, y el sitio **Can I use**, que vamos a ver a continuación.

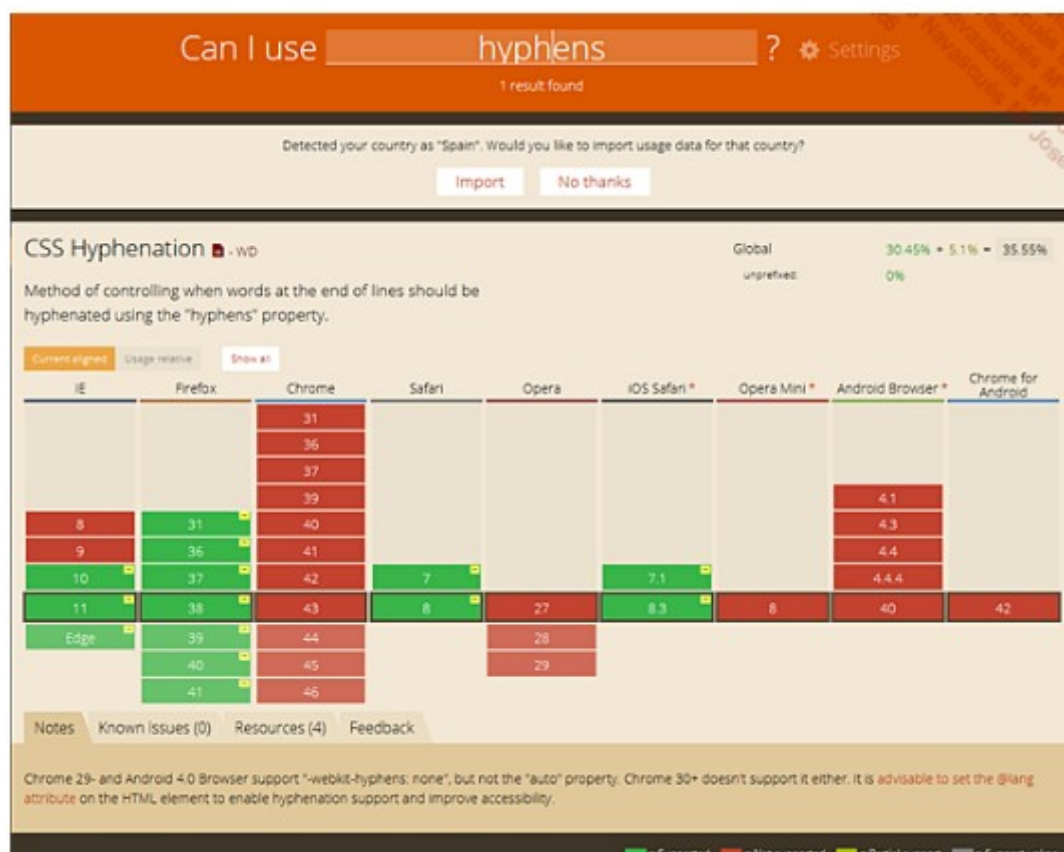
4.4. EL SITIO CAN I USE

Como acabamos de ver, el sitio **Current Work** del W3C permite seguir la evolución de los módulos que componen las CSS3. Sin embargo, no puede visualizar la implementación de las propiedades en los navegadores, es decir, no visualiza las propiedades que pueden usarse en el instante «t». Para ello, debe consultar, también regularmente, el sitio **Can I use**: <http://caniuse.com>

Este sitio muestra las compatibilidades entre los navegadores y los lenguajes HTML5 y CSS3 (y también SVG, JavaScript API y otros estándares del Web).



En el campo **Can I use**, introduzca la propiedad CSS3 o el elemento HTML5 que desee utilizar. En este ejemplo, hemos probado con la propiedad hyphens.



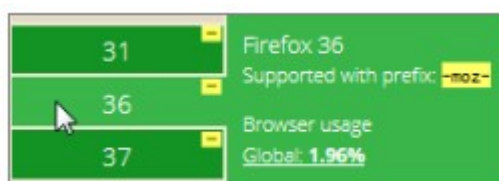
Aquí dispone de la tabla de compatibilidad entre la propiedad introducida y las principales versiones de los navegadores actuales. La tabla emplea cuatro colores:

- **Verde:** Supported.
- **Rojo:** Not supported.
- **Verde flojo:** Partial support.
- **Gris:** Support unknown.

A continuación, dispone de cuatro pestañas que son también muy importantes para obtener toda la información técnica útil: **Notes**, **Known issues**, **Resources** y **Feedback**.

Finalmente, se indica, para cada navegador, si es necesario el uso de prefijos.

En este ejemplo (propiedad hyphens), el uso de prefijos es necesario para los navegadores que reconocen esta propiedad. Si pasa el puntero del ratón por la casilla, se le indicará el prefijo de estos navegadores.



Las nuevas propiedades CSS3 son extremadamente poderosas y deben ser estudiadas una por una, pero para facilitar su aprendizaje vamos a aplicar todas ellas sobre la misma plantilla. Por este motivo comenzaremos por crear un documento

HTML sencillo con algunos estilos básicos:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Nuevos Estilos CSS3</title>
  <link rel="stylesheet" href="nuevocss3.css">
</head>
<body>
  <header id="principal">
    <span id="titulo">Estilos CSS Web 2.0</span>
  </header>
</body>
</html>
```

Listado. Una plantilla simple para probar nuevas propiedades.

Nuestro documento solo tiene una sección con un texto breve en su interior. El elemento `<header>` usado en la plantilla podría ser reemplazado por `<div>`, `<nav>`, `<section>` o cualquier otro elemento estructural de acuerdo a la ubicación en el diseño y a su función. Luego de aplicar los estilos, la caja generada con el código del ejemplo del Listado aparecerá como una cabecera, por consiguiente decidimos usar `<header>` en este caso.

Debido a que el elemento `` se encuentra en desuso en HTML5, los elementos usados para mostrar texto son normalmente `` para líneas cortas y `<p>` para párrafos, entre otros. Por esta razón el texto en nuestra plantilla fue insertado usando etiquetas ``.

Los siguientes son los estilos básicos requeridos por nuestro documento HTML:

```
body {
  text-align: center;
}
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;
}
#titulo {
  font: bold 36px verdana, sans-serif;
}
```

Listado. Reglas básicas CSS con las que comenzar.

No hay nada nuevo en las reglas del Listado, solo los estilos necesarios para dar forma a la plantilla y crear una caja ancha, posicionada en el centro de la

ventana, con un fondo gris, un borde y un texto grande en su interior que dice “EstilosCSS Web 2.0”.

5. HERRAMIENTAS Y TEST DE VERIFICACIÓN

CSS es un estándar de W3C. Para que los desarrolladores puedan comprobar que los estilos que definen cumplen ese estándar, el consorcio de estándares web W3C (World Wide Web Consortium) proporciona herramientas para validar tanto el código HTML como las hojas de estilo CSS, comprobando si éstas son correctas según las gramáticas publicadas.

A este servicio de validación se puede acceder a través de la página web: <http://jigsaw.w3.org/css-validator/>. Su uso es muy sencillo, el usuario solo tiene que introducir la URL del sitio web que pretenda evaluar y seleccionar las opciones que desea considerar.

Si el proceso de validación no encuentra errores, sus autores pueden incluir en la página web un icono como el siguiente. Con la inclusión de esta imagen los visitantes verán que los desarrolladores se han preocupado por crear un sitio web interoperable y acorde al estándar.



Además de la W3C las principales herramientas de desarrollo de sitios web también ofrecen facilidades para el desarrollo y validación de hojas de estilo.

Otras herramientas de validación que pueden utilizarse para comprobar nuestras hojas de estilo podemos encontrarlas ligadas a determinados navegadores; por

ejemplo Firefox ofrece un complemento que puede instalarse y permite validar un sitio web. Dicho complemento puede descargarse en <https://addons.mozilla.org/es-es/firefox/addon/css-validator/>.

XHTML-CSS (<http://xhtml-css.com/>) es otra herramienta de validación que puede utilizarse indistintamente para comprobar la bondad (lo adecuado) de nuestro código XHTML y el CSS. Los informes ofrecidos por estas herramientas, en muchas ocasiones, están relacionados con distintas situaciones que deben ser comprobadas por el desarrollador o diseñador del sitio web, pero que no necesariamente son errores que haya que subsanar obligatoriamente. Dichas situaciones son identificadas como warnings, es decir, se trata de código cuyo bondad o no debe ser comprobada en última instancia por un humano, ya que automáticamente no hay evidencias totalmente objetivas y automáticas que permitan afirmar que hay una vulneración en el uso y construcción de la hoja de estilo.

Además de herramientas de validación, también hay otras ligadas a los navegadores (plugins) que permiten examinar CSS y detectar errores y editar código al instante. Algunos ejemplos son:

- Firebug (<http://getfirebug.com/>): es unplugin para Firefox y Chrome (Firebug Lite) entre otros navegadores, que permite, por ejemplo, analizar CSS en cascada, editar "en vivo" reglas y atributos (propiedades) y autocompletar valores de atributos con sugerencias de contexto. Esta herramienta no es solo para ayudar en el desarrollo de CSS, sino que es un asistente para todo lo que conlleva el desarrollo web. Por ejemplo, permite inspeccionar códigos HTML y Javascript.
- Pendule (<https://chrome.google.com/webstore/detail/gbkffbkamcejhkcaocmkdeiiccpmjfdi>): es un plugin de Chrome que está más especializado en CSS, permitiendo validar CSS pero también visualizar reglas, deshabilitarlas, mostrar los colores usados, etc. Es una herramienta simple para comprobar problemas en un sitio web (depurar).

Por otro lado, en Internet hay una gran cantidad de herramientas de uso gratuito que permiten generar (y ayudar a generar) código CSS automáticamente. Algunos ejemplos son:

- List-o-matic (<http://www.accessify.com/tools-and-wizards/developer-tools/list-o-matic/>): permite crear código CSS para menús. Se indica el texto que aparecerá en los enlaces, se selecciona el diseño preferido y la herramienta crea el CSS que lo representa, para que el desarrollador lo copie e incorpore en su web.
- CSS Layout Generator (<http://www.pagecolumn.com/>): permite crear CSS para páginas web con diferentes capas, distribuidas de la manera que el usuario desee (varias columnas, horizontales, verticales, etc.). Una vez seleccionado el diseño preferido, la herramienta crea el código para copiar y pegar. Se puede elegir solo el CSS o también el HTML que lo usa.

- CSS Text Wrapper (<http://www.csstextwrap.com/>): es una herramienta que permite colocar mediante CSS un texto dentro de la forma que se desee. Lo normal es colocar los textos en rectángulos. Pero esta herramienta genera código para colocarlo según la forma (circular, trapezoidal, triangular, etc.) que el usuario elija visualmente. Es una utilidad muy adecuada para hacer textos aparentes en un web.

La gran cantidad de sitios web que ofrecen de forma gratuita o de pago, plantillas HTML o CSS ya creados, son de gran ayuda para la creación de sitios web. Los desarrolladores, ante la idea de empezar un nuevo sitio web desde cero, pueden optar por usar una plantilla ya creada y modificarla para que obtenga toda la funcionalidad que un cliente quiera.

FreeCSSTemplates (<http://www.freecsstemplates.org/>) es uno de los sitios que ofrecen plantillas CSS3-HTML5 gratuitas.

EJEMPLO

http://designlovr.com/examples/dynamic_stack_of_index_cards/