

UF1303: ELABORACIÓN DE HOJAS DE ESTILO.

Manual teórico.

INDICE

1.	Hojas de Estilo CSS3 en la creación de páginas web.	4
1.1.	Funciones y características.	4
1.1.1.	Las tres capas del diseño web.	4
1.1.2.	Una breve historia del CSS.	5
1.1.3.	CSS, HTML Y XHTML.	6
1.2.	Funciones y características.	6
1.3.	El modelo de forma visual y el modelo de caja.	7
1.4.	Tipos de estilos.	8
1.5.	Selectores y reglas de estilos.	10
1.6.	Atributos de estilo.	14
1.6.1.	Fondo.	15
1.6.2.	Propiedades de los textos.	16
1.6.3.	Los textos en CSS.	18
1.6.4.	Bordes en CSS.	23
1.6.5.	Márgenes con CSS.	30
1.6.6.	CSS Relleno.	33
1.6.7.	CSS Fonts – Fuentes.	35
1.6.8.	Los colores en CSS.	43
1.6.9.	CSS3. Transparencias en colores RGBA.	49
1.6.10.	Contornos con CSS.	51
1.6.11.	Listas con CSS.	56
1.6.12.	Tablas con CSS.	60
1.7.	Prefijos CSS de Navegadores (webkits).	67
1.7.1.	Columnas en css.	69
1.7.2.	Esquinas redondeadas en css.	71
1.7.3.	Efecto sombra en css.	73
1.7.4.	Transformaciones con CSS.	76
1.7.5.	CSS Media Queries.	78
2.	Diseño, ubicación y optimización de los contenidos de una página web.	82
2.1.	Creación de un documento funcional.	82
2.2.	Diseño de los contenidos.	84
2.3.	Identificación de la información que se tiene que ubicar en la página web.	89
2.4.	Selección de contenidos para cada elemento de la página.	89



2.5.	Uso del documento funcional para las especificaciones del diseño.	90
2.6.	Tipos de página para la ubicación de contenidos.	92
2.7.	Definición de los tipos de página según los contenidos y sus funcionalidades.....	93
2.8.	Selección del Tipo de página para la página web.	95
2.9.	Uso del documento funcional para las especificaciones del diseño.	97
2.10.	Especificaciones de navegación.....	98
2.11.	Creación de un mapa de navegación de páginas.	99
2.12.	Uso del documento funcional para integrar el mapa de navegación.....	100
2.13.	Elementos utilizados para navegar.	101
2.14.	Elaboración de una guía de usuario.....	101

1. Hojas de Estilo CSS3 en la creación de páginas web.

1.1. Funciones y características.

El estándar HTML se encuentra en continua evolución. Los navegadores son cada vez más capaces de plasmar la información de forma atractiva.

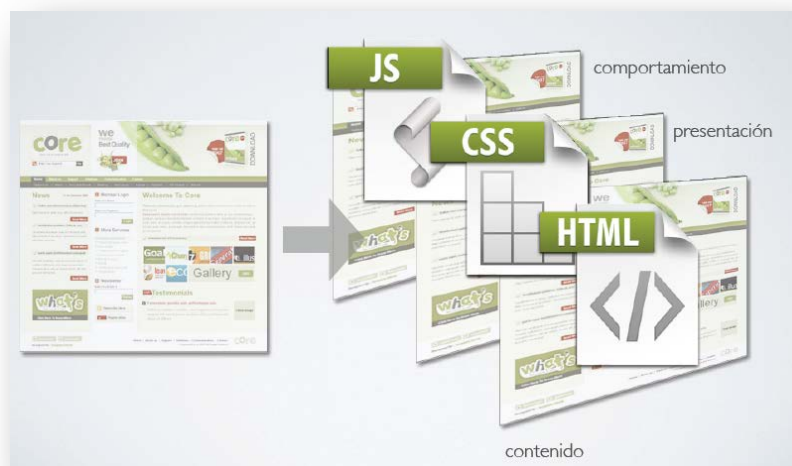
Es por ello que aparecen las hojas de estilo. Son una respuesta a la necesidad de organizar la presentación del contenido de un documento HTML.

La especificación HTML fue concebida inicialmente como un lenguaje de marcas, y la introducción de varios niveles de formateo de elementos y atributos de texto sólo sirvió para crear más confusión. Microsoft decidió entonces implementar un nuevo concepto, las hojas de estilo, soportadas a partir de I.E.3.0, se mejoró la especificación en I.E.4.0. Netscape también implementó las hojas de estilo para sus navegadores a partir de la versión 4.0.

1.1.1. Las tres capas del diseño web.

Los estándares Web fomentan la separación de un contenido Web en 3 capas:

- Contenido: Define estructura semántica y contenido de texto plano, se representa con el documento HTML.
- Presentación: Añade el aspecto del contenido, lo representa la hoja de estilo CSS
- Comportamiento: Añade mayor grado de interacción y funciones complejas, es posible representarlo con un documento JS o de otras tecnologías semejantes



Un documento html es un visualizador de la información.

Para hacer más atractivo este documento html , podemos utilizar diversas posibilidades como: combinación de colores, distintos tipos de letras, fondo del documento, ubicación por coordenadas en píxeles de un determinado texto, etc. Todas esas posibilidades las tenemos en las hojas de estilo o Style Sheets. Las hojas de estilo son una extensión del documento HTML, en las que se diseña la apariencia general del mismo.

Las principales ventajas de utilizar las hojas de estilo son:

- **Accesibilidad.** Los datos se pueden presentar en función del usuario y adaptarlos a sus necesidades. No es lo mismo presentar una web en un Pc, que presentarla en un teléfono móvil, o presentarla a personas con alguna discapacidad.
- **Mantenimiento.** Separa y centraliza la presentación, bien en diferentes archivos o en partes bien conocidas del archivo, con lo cual tenemos localizado el código que queremos mantener y/o actualizar.
- **Simplicidad.** Se obtienen documentos más fáciles de entender y se reduce el tamaño de los mismos.

1.1.2. Una breve historia del CSS.

Su verdadero desarrollo se produjo con el boom de internet y el crecimiento del lenguaje HTML, especialmente en los años de la guerra entre navegadores, ya que la falta de una especificación permitía que una página escrita para un navegador fuera completamente distinta para otro.

En la tabla siguiente podemos ver a evolución de las hojas de estilo por orden cronológico.

Año	Nombre	Descripción
1970	CHSS	Se comienza a hablar de las hojas de estilo. Comienzan a surgir lenguajes de especificación. CHSS (Cascading HTML Style Sheets)
1990	SSP	SSP (Stream-based Style Sheet Proposal)
1994	CSS	Guerra de los navegadores. Se unen los creadores de los lenguajes CHSS y SPP para definir una nueva especificación CSS.
1996	CSS Nivel 1	Primera especificación oficial del lenguaje CSS.
1998	CSS Nivel 2	Aparece la segunda versión del lenguaje. Sigue siendo utilizada en la actualidad con alguna modificación, CSS 2.1.
2007	CSS Nivel 3	Última especificación del lenguaje aunque hoy en día no se ha terminado de implementar en la mayoría de los navegadores.

URL de interes:

Página oficial de CSS: <http://www.w3.org/Style/CSS/>.

1.1.3. CSS, HTML Y XHTML.

Con la aparición de las hojas de estilo, los diseñadores están cambiando la forma de crear las páginas web, para adaptarse a los nuevos modelos de diseño.

Poco a poco las etiquetas HTML que permiten aplicar estilo al contenido (como por ejemplo la etiqueta `` que permite aplicar el tipo de letra, color, tamaño, etc) se consideran obsoletas o no recomendadas y van desapareciendo.

Posteriormente, después del HTML 4.1 apareció el XHTML. No es nada más que una redefinición del HTML 4 con una sintaxis XML, la cual conservaba la mayoría de elementos y atributos, exceptuando los elementos y atributos relacionados con el estilo, siendo XHTML mucho más estricto en formato del documento. Por ejemplo, obligaba a que todos los elementos y atributos estuvieran escritos en minúscula, también que todos los elementos tengan marcas de inicio y fin, etc.

En la actualidad y con HTML5, se tiende a separar los datos de su presentación para poder adaptar mejor el diseño a diferentes dispositivos o usuarios. Por lo tanto en una primera fase del diseño, se utiliza el HTML para formatear los contenidos, es decir se definen la función de cada elemento una vez definidos esos contenidos, con CSS, se procede a definir el aspecto visual de cada elemento: color, tamaño, tipografía, posición dentro página, etc, etc.

1.2. Funciones y características.

Un documento CSS se compone de un conjunto de reglas y cada regla consta de las siguientes partes:

- **Selector:** Indica a qué elemento o parte de la página se aplica la regla. Normalmente los selectores se corresponden con las etiquetas del lenguaje HTML. Por ejemplo `table` es un selector.
- **Declaración:** Indica la definición del estilo a aplicar al selector indicado. La declaración a su vez puede dividirse en dos partes:
 - **Propiedad:** Indica el nombre del atributo al que va a ser aplicado.
 - **Valor:** Define el valor de la propiedad.

Dentro de una regla se puede realizar más de una declaración, para ello las definiciones se separan por punto y coma, no siendo necesario introducirla en la última definición, como podemos ver en el ejemplo siguiente:

```
h2{  
  
    color:green;
```

background-color: red;

font-size: large;

font-family: garamond

{

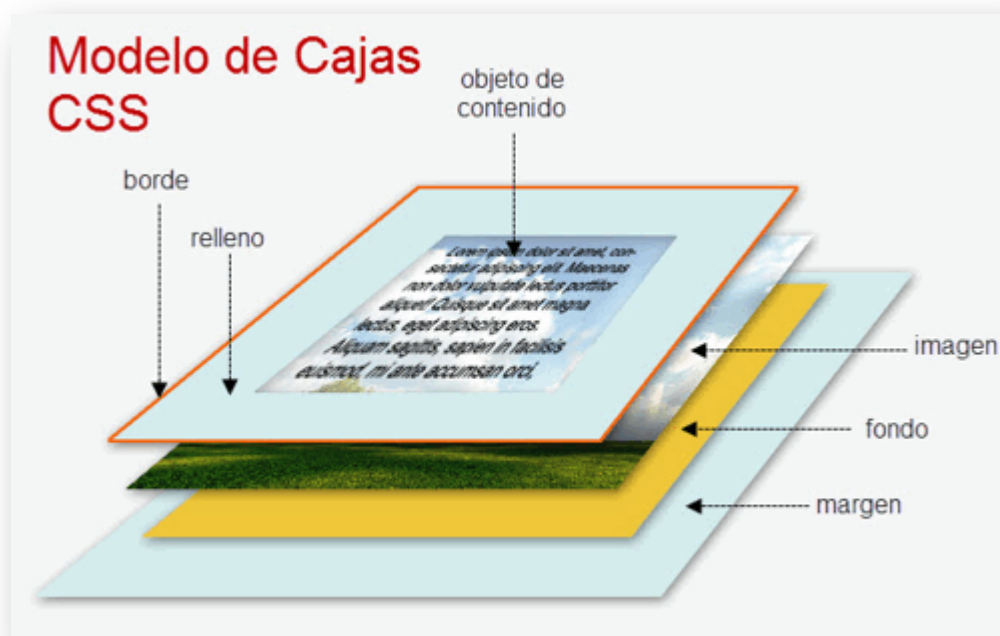
1.3. El modelo de forma visual y el modelo de caja.

El modelo de caja es la base de cualquier diseño CSS, ya que todo elemento CSS se basa en una caja rectangular. Cada elemento está contenido en una caja que contiene diferentes elementos (áreas o subcajas) que aportan diferentes opciones al elemento, en lo referente a márgenes, bordes, relleno y contenido del propio elemento.

Las cajas de una página se crean automáticamente. Cada vez que se inserta una etiqueta HTML, se crea una nueva caja rectangular que encierra los contenidos de ese elemento.

Las cajas de las páginas no son visibles a simple vista porque inicialmente no muestran ningún color de fondo ni ningún borde.

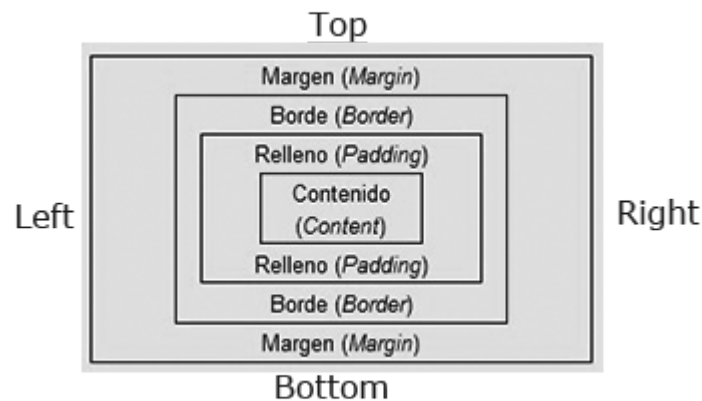
Los navegadores crean y colocan las cajas de forma automática, pero CSS permite modificar todas sus características. Cada una de las cajas está formada por seis elementos:



Las partes que componen cada caja y su orden de visualización desde el punto de vista del usuario son:

- **Objeto de contenido:** se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)

- **Relleno (padding):** espacio libre opcional existente entre el contenido y el borde.
- **Borde (border):** línea que encierra completamente el contenido y su relleno.
- **Imagen de fondo (background image):** imagen que se muestra por detrás del contenido y el espacio de relleno.
- **Color de fondo (background color):** color que se muestra por detrás del contenido y el espacio de relleno.
- **Margen (margin):** separación opcional existente entre la caja y el resto de cajas adyacentes.



El relleno y el margen son transparentes, por lo que en el espacio ocupado por el relleno se muestra el color o imagen de fondo (si están definidos) y en el espacio ocupado por el margen se muestra el color o imagen de fondo de su elemento padre (si están definidos). Si ningún elemento padre tiene definido un color o imagen de fondo, se muestra el color o imagen de fondo de la propia página (si están definidos).

Si una caja define tanto un color como una imagen de fondo, la imagen tiene más prioridad y es la que se visualiza. No obstante, si la imagen de fondo no cubre totalmente la caja del elemento o si la imagen tiene zonas transparentes, también se visualiza el color de fondo. Combinando convenientemente cada una de estas partes se obtienen resultados muy interesantes.

Si el límite del padding tiene un ancho de 0, el límite del relleno sería el mismo que el límite del contenido. Igual ocurrirá para los límites superiores, por ejemplo si el límite de margen tiene un ancho 0, este sería igual al límite del borde.

El ancho de la caja está formado por la suma del ancho de los márgenes, bordes, rellenos izquierdos y derechos, del ancho del contenido. Mientras la altura está dada por la suma de los márgenes, bordes y rellenos superiores e inferiores, y la altura del contenido.

1.4. Tipos de estilos.

Las reglas de estilo pueden ser declaradas: en línea con el elemento, en la cabecera (head) del documento o en un documento externo con extensión CSS.

- **Utilizando estilos** directamente sobre aquellos elementos que lo permiten a través del atributo `<style>` dentro de `<body>`. Pero este tipo de definición del estilo pierde las ventajas que ofrecen las hojas de estilo

al mezclarse el contenido con la presentación. Veamos el siguiente ejemplo:

```
<h2 style="color: red; font-family: Helvetica, Geneva, Arial, sans-serif"> Título
de la cabecera </h2>
```

- **Declaración en el head del CSS.** Utilizando el elemento `<style>`, en el interior del documento al que se le quiere dar estilo, en la sección `<head>`. De esta forma los estilos serán reconocidos antes de que la página se cargue por completo.

```
<!DOCTYPE html>
<html lang="es">
<title>hoja de estilo interna</title>
  <style type="text/css">

    body {
      padding-left: 11em;
      font-family: Georgia, "Times New Roman", serif;
      color: red;
      background-color: #d8da3d;
    }

    h1 {
      font-family: Helvetica, Geneva, Arial, sans-serif;
    }

  </style>
</head>
<body>
  <h1>Aquí se aplicará el estilo de letra para el Título</h1>
</body>
</html>
```

- **Utilizando una hoja de estilo externa que estará vinculada a un documento a través del elemento `<link>`**, el cual debe ir situado en la sección `<head>`.

```
<!DOCTYPE html>
<html lang="es">

<head>
  <title>Título</title>
  <link rel="stylesheet" type="text/css" href="css/misestilos.css">
</head>
<body>
</body>
```

</html>

En **href** indicamos la dirección en la que se encuentra la hoja de estilo. Con **rel="stylesheet"** informamos al navegador que se trata de una hoja de estilo, en **type**, como siempre, le decimos que está escrita en lenguaje CSS.

1.5. Selectores y reglas de estilos.

Como ya sabe, el selector indica el elemento o elementos a los que se le aplica la regla CSS y, muy importante, tenga en cuenta que los selectores distinguen entre mayúsculas y minúsculas.

A continuación se presentan los **tipos de selectores** más usados, se describe su sintaxis, se explica a qué elementos afecta el selector y se presentan ejemplos.

Tipo de Selector	Cómo se indica	Observaciones	Ejemplo
Selector universal	Se indica con un asterisco (*)	Afecta a todos los elementos de la página, por lo que es poco utilizado (difícilmente un estilo se aplica a toda la página).	* { declaración }

Tipo de Selector	Cómo se indica	Observaciones	Ejemplo
Selector de tipo o etiqueta	Se indica con el nombre de una etiqueta html.	Se aplica a todos los elementos de la página cuya etiqueta HTML coincida con el valor del selector.	<p>p { declaración}</p> <p>h1 { declaración}</p> <p>a { declaración}</p>

Tipo de Selector	Cómo se indica	Observaciones	Ejemplo
Selector de ID	Se indica con la almohadilla (#) y el valor del atributo <i>id</i> del elemento que se quiere seleccionar. #texto	Se aplica al elemento específico de la página cuyo atributo <i>id</i> coincida con el texto del selector.	#principal { color: blue; }

Tipo de Selector	Cómo se indica	Observaciones	Ejemplo
Selector de clase	Se indica con un punto (.) y el valor del atributo class del elemento que se quiere seleccionar. .texto	<ul style="list-style-type: none"> • Se aplica a todos los elementos de la página cuyo atributo class coincida con el selector. • Permite seleccionar varios elementos de la páginas, sin importar su tipo, ni el lugar en que estén en la misma. • Se puede restringir el alcance de los selectores, para seleccionar solamente los elementos de un tipo y un atributo class determinado, se indica la etiqueta del elemento y, sin dejar ningún espacio, se indica el selector de clase. 	.texto { color: red; } .pie { color: blue; } em.especial {color: green; }

Tipo de Selector	Cómo se indica	Observaciones	Ejemplo
Selector de clase	Se indica con un punto (.) y el valor del atributo class del elemento que se quiere seleccionar. .texto	<ul style="list-style-type: none"> Se aplica a todos los elementos de la página cuyo atributo class coincida con el selector. Permite seleccionar varios elementos de la páginas, sin importar su tipo, ni el lugar en que estén en la misma. Se puede restringir el alcance de los selectores, para seleccionar solamente los elementos de un tipo y un atributo class determinado, se indica la etiqueta del elemento y, sin dejar ningún espacio, se indica el selector de clase. 	.texto { color: red; } .pie { color: blue; } em.especial {color: green; }

Tipo de Selector	Cómo se indica	Observaciones	Ejemplo
Selector de hijos	Se indica separando los selectores con el signo de mayor (>) sell > sel 2	Permite seleccionar un elemento que es <i>hijo directo</i> de otro elemento (no existen otros elementos entre ellos)	p > span { color: blue; }

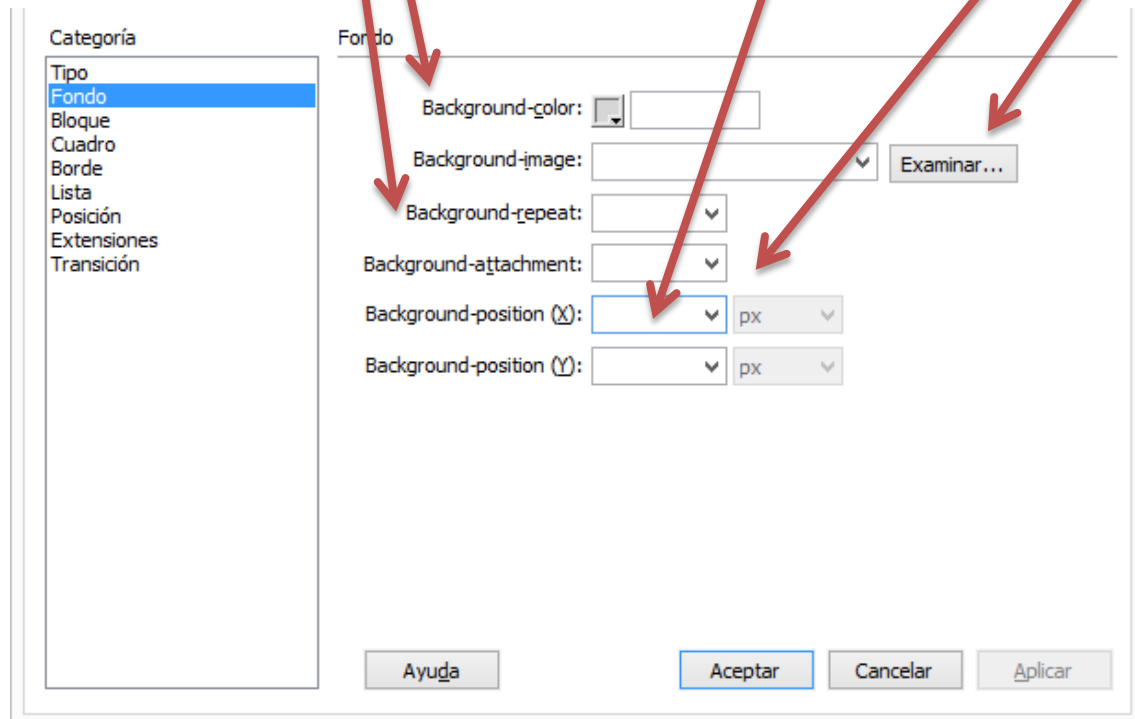
Tipo de Selector	Cómo se indica	Observaciones	Ejemplo
Selector adyacente	Se indica separando los selectores con el signo de más (+) <i>sel1 + sel 2</i>	Permite seleccionar elementos adyacentes (Uno después del otro), afectando la regla al que está de segundo selector.	h1 + h2 { color: blue; }

1.6. Atributos de estilo.

Las propiedades en CSS permiten especificar valores de medida y color a todos los elementos, de muchas formas distintas. A continuación se presentan diferentes formas para aplicar un valor a una propiedad.

1.6.1. Fondo.

Propiedades	Descripción
background-color	Aplicar color de fondo a diferentes elementos.
background-image	Utilizar una imagen como fondo.
background-repeat	La imagen de fondo se repite.
background-attachment	Dejar fija la imagen de fondo.
background-position	Ubicar una imagen en un lugar determinado.



Categoría

- Tipo
- Fondo**
- Bloque
- Cuadro
- Borde
- Lista
- Posición
- Extensiones
- Transición

Fondo

Background-color:

Background-image:

Background-repeat:

Background-attachment:

Background-position (X): px

Background-position (Y): px

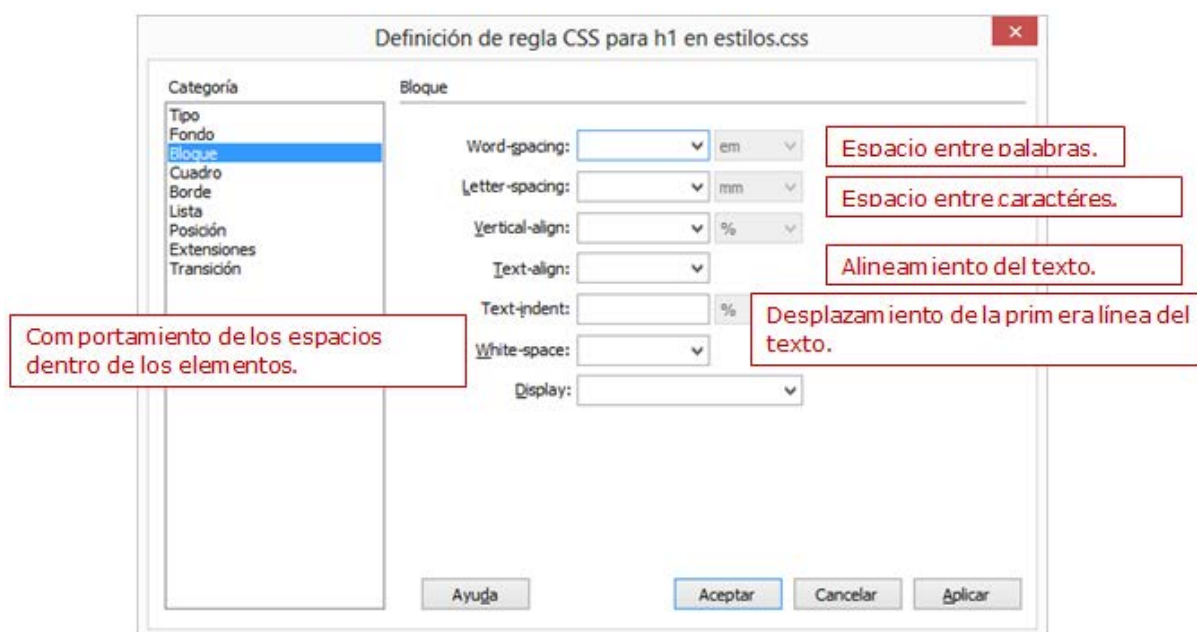
Ayuda Aceptar Cancelar Aplicar

Estilos CSS en Dreamweaver Categoría Fondo

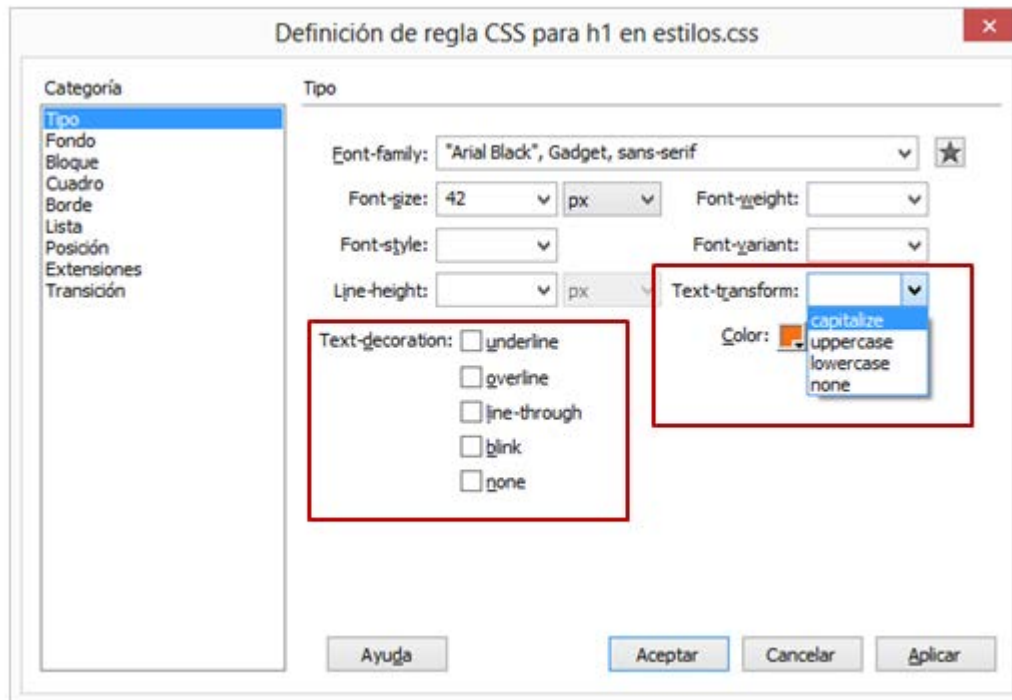
1.6.2. Propiedades de los textos.

Propiedad	Descripción	Valores	Detalles
text-indent	Desplazamiento de la primera línea del texto.	<i>longitud</i>	<i>Longitud</i>
		%	<i>Porcentaje</i>
text-align	Alineamiento del texto.	left	Izquierda
		right	Derecha
		center	Centro
		justify	Justificar
text-decoration	Efectos de subrayado, tachado, parpadeo.	none	Sin efectos
		underline	Subrayado
		overline	Línea por encima
		line-through	Tachado
		blink	Parpadeo
text-transform	Transformaciones del texto a mayúsculas/minúsculas.	capitalize	Convierte en mayúscula el primer carácter de cada palabra
		uppercase	Convierte en mayúscula todas las letras del elemento
		lowercase	Convierte en minúscula todas las letras del elemento
		none	Neutraliza el valor heredado
letter-spacing	Espacio entre caracteres.	normal	Normal
		<i>longitud</i>	<i>Longitud</i>
word-spacing	Espacio entre palabras.	normal	Normal
		<i>longitud</i>	<i>Longitud</i>

Propiedad	Descripción	Valores	Detalles
white-space	Comportamiento de los espacios dentro de los elementos.	normal	Normal
		pre	Preformateado
		nowrap	Los cambios de línea solo ocurren con el elemento <code>br</code>
		pre-wrap	
		pre-line	
direction	Sentido direccional de la escritura.	ltr	Izquierda a derecha
		rtl	Derecha a izquierda
unicode-bidi	Sentido direccional de la escritura.	normal	Normal
		embed	Abre un nivel adicional de incrustación con respecto al algoritmo bidireccional
		bidi-override	Si el elemento es a nivel de línea o es un elemento a nivel de bloque, crea una sustitución



Estilos CSS en Dreamweaver Categoría Bloque



Atributos Text-transform y Text-decoration en Dreamweaver.

1.6.3. Los textos en CSS.

Las propiedades de los textos nos permiten controlar la apariencia de los mismos.

Entre los ajustes que podemos aplicar a los textos, tenemos:

1. La sangría
2. El alineado
3. La decoración
4. Espacio entre letras
5. Espacio entre palabras
6. Mayúsculas y minúsculas
7. Espacios en blanco

- **Sangría de los textos - text-indent**

La propiedad **text-indent** se utiliza para generar sangría en la primera línea de un texto.

Ejemplo

Vamos a ver cómo se comporta un texto con sangría.

Código

```
<head>
<style type="text/css">
p{text-indent:2cm}
</style>
</head>

<body>
<p>En la primera línea de este párrafo
observamos una sangría de 2
centímetros de distancia del borde.</p>
</body>
```

En la primera línea de este párrafo observamos una sangría de 2 centímetros de distancia del borde

• Alineado de los textos - text-align

La propiedad **text-align** se utiliza para alinear un texto a la derecha, izquierda o centro del bloque que lo contiene.

Sintaxis

```
<head>
<style="type: text/css">
selector { text-align: valor}
</style>
</head>
```

Los posibles valores para alinear los textos

left | right | center | justify

Ejemplo

Alineamos un texto a la derecha y otro en el centro.

Código

```
<head>
<style type="text/css">
p.der{text-align:right}
p.cen{text-align:center}
</style>
</head>

<body>
<p class="der">Texto a la derecha</p>
<p class="cen">Texto en el centro</p>
</body>
```

Texto a la derecha
Texto en el centro

- **La decoración de los textos - text-decoration**

La propiedad **text-decoration** se utiliza para subrayar, tachar, remarcar con una línea superior o parpadear un texto.

Sintaxis

```
<head>
<style="type: text/css">
selector {text-decoration: valor}
</style>
</head>
```

Los posibles valores para decorar los textos

none | underline | overline | line-through | blink

Ejemplo

Definimos un texto parpadeante y un enlace sin subrayar.

Código	
<pre><head> <style type="text/css"> p.parpadeo{text-decoration:blink} a.sin_linea{text-decoration:none} </style> </head> <body> <p class="parpadeo">Texto parpadeando</p> Ir a la home de Virtualnauta </body></pre>	<p>Texto parpadeando</p> <p><i>Ir a la home de elprofedemicurso</i></p>

- **Separación entre letras - letter-spacing**

La propiedad **letter-spacing** se utiliza para definir la distancia que queremos dejar entre letra y letra de un mismo texto.

Sintaxis

```
<head>
<style="type: text/css">
selector {letter-spacing: valor}
</style>
</head>
```

Los posibles valores para esta propiedad

normal | distancia

Ejemplo

Vamos definir una distancia entre letras de 5 píxeles.

Código	
<pre><head> <style type="text/css"> p{letter-spacing:5px} </style> </head> <body> <p>La distancia entre letras es de 5 píxeles</p> </body></pre>	La distancia entre letras es de 5 píxeles

- Separación entre palabras - word-spacing

La propiedad **word-spacing** se utiliza para definir la distancia que queremos dejar entre palabras.

Sintaxis

```
<head>
<style="type: text/css">
selector { word-spacing: valor }
</style>
</head>
```

Los posibles valores para esta propiedad

normal | distancia

Ejemplo

Vamos definir una distancia entre palabras de 2 centímetros.

Código	
<pre><head> <style type="text/css"> p{word-spacing:2cm} </style> </head> <body> <p>La distancia entre palabras es de 2 centímetros</p> </body></pre>	La distancia entre palabras es de 2 centímetros

- **Transformar los textos a mayúsculas o minúsculas - text-transform**

La propiedad **text-transform** se utiliza para convertir un texto a mayúsculas o minúsculas.

Sintaxis

```
<head>
<style="type:text/css">
selector {text-transform: valor}
</style>
</head>
```

Los posibles valores para convertir los textos

capitalize | uppercase | lowercase | none

Ejemplo

Vamos definir la primera letra de cada palabra en mayúsculas.

Código	
<pre><head> <style type="text/css"> p{text-transform:capitalize} </style> </head> <body> <p>La primera letra de cada palabra se ha convertido a mayúsculas</p> </body></pre>	La Primera Letra De Cada Palabra Se Ha Convertido A Mayúsculas

- **Espacios en blanco - white-space**

La propiedad **white-space** se utiliza para manipular el comportamiento de los espacios en blanco dentro de cada elemento.

Sintaxis

```
<head>
<style="type:text/css">
selector {white-space: valor}
</style>
</head>
```

Los posibles valores para esta propiedad

normal | pre | nowrap | pre-wrap | pre-line

Ejemplo

Vamos ver el comportamiento de los espacios en blanco dentro de un texto.

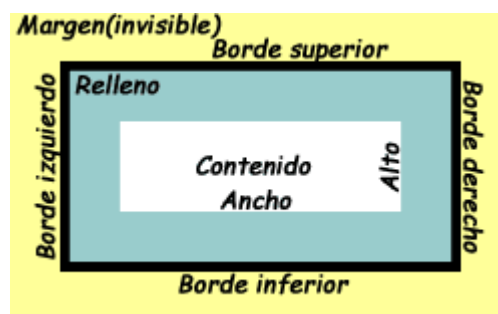
Código	
<pre><head> <style type="text/css"> p.pre{white-space:pre-wrap} p.normal{white-space:normal} </style> </head> <body> <p class="pre">Dejo varios espacios en blanco entre palabra y palabra.</p> <p class="normal">Aquí también dejo varios espacios en blanco entre palabra y palabra pero los comprime a uno solo.</p> </body></pre>	<p>Dejo varios espacios en blanco entre palabra y palabra.</p> <p>Aca también dejo varios espacios en blanco entre palabra y palabra pero los comprime a uno solo.</p>

1.6.4. Bordes en CSS.

Usa color, estilo y espesor en los bordes de todos los elementos.

Los bordes nos sirven para decorar todos los elementos con líneas de diferentes espesores, colores y formas.

Esta propiedad se aplica al área de bordes de la caja.



- **La propiedad de los bordes**

La propiedad **border** especifica el espesor, color y estilo de los bordes.

Es la forma abreviada para definir los bordes y puede comprender en ella todas las propiedades de los bordes juntas.

Sintaxis

selector {**border**: **valor-1 valor-2 valor-n**}

Nota: debemos dejar un espacio en blanco entre los valores

Los posibles valores para definir los espesores de los bordes

thin | medium | thick | tamaño (px, pc, pt, mm, cm, in) | nombre del color(inglés) | #xxxxxx | transparent | none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset

Ejemplo

Vamos a decorar los bordes de una tabla.

Código

```
<head>
<style type="text/css">
table{border: blue double medium}
td{border: green dotted medium}
</style>
</head>

<body>
<table>
<tr>
<td>Esta tabla</td>
<td>tiene los bordes</td>
</tr>
<tr>
<td>decorados con</td>
<td>diferentes propiedades</td>
</tr>
</table>
</body>
```

Esta tabla	tiene los bordes
decorados con	diferentes propiedades

Nota:

Para poder visualizar el borde debemos usar valores de la propiedad **border-style**

- **Espesor de los bordes**

La propiedad **border-width** especifica el espesor del borde

Sintaxis

Establecer el mismo espesor para todos los bordes del selector.

```
selector {border-width: valor}
```

Establecer el espesor de cada borde del selector por separado.

```
selector {  
border-top-width: valor;  
border-right-width: valor;  
border-bottom-width: valor;  
border-left-width: valor;  
}
```

Nota:

no olvidemos poner (;) al final de cada línea

Los posibles valores para definir los espesores de los bordes

thin fino | medium medio | thick grueso | tamaño (px, pc, pt, mm, cm, in)

Ejemplo

Vamos a aplicar un espesor a un borde sobre un texto.

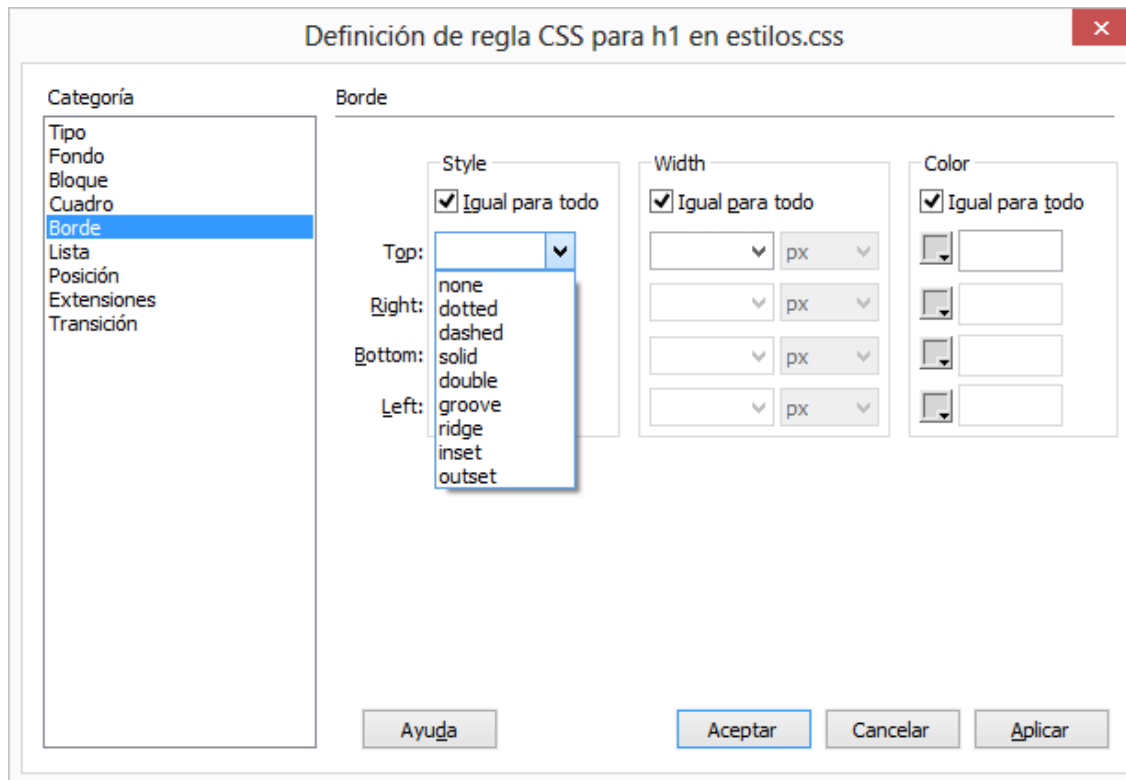
Código

```
<head>  
<style type="text/css">  
p { border-width: 5px;  
border-style: solid; }  
</style>  
</head>  
<body>  
<p>Texto con un borde de 5 pixels</p>  
</body>
```

Texto con un borde de 5 pixels

Nota:

Para poder visualizar el borde debemos usar valores de la propiedad **border-style**



Atributos de borde en Dreamweaver.

- **Color de los bordes**

La propiedad **border-color** especifica el color del borde

Sintaxis

Establecer el color de todos los bordes a la vez.

```
selector {border-color: valor}
```

Establecer el color de cada borde por separado.

```
selector {
border-top-color: valor;
border-right-color: valor;
border-bottom-color: valor;
border-left-color: valor;
}
```

Los posibles valores para colorear los bordes

nombre del color(inglés) | #xxxxxx | transparent

Ejemplo

Vamos a aplicar un color a un borde sobre un título.

Código

```
<head>
<style type="text/css">
h2{
border-color: red;
border-style: solid;
}
</style>
</head>

<body>
<h2>Título con un borde rojo</h2>
</body>
```

Título con un borde rojo

Nota:

Para poder visualizar el borde debemos usar valores de la propiedad **border-style**

- **Estilo de los bordes**

La propiedad border-style especifica el estilo del borde

Sintaxis

Establecer el estilo de todos los bordes a la vez.

```
selector {border-style: valor}
```

Establecer el estilo de cada borde por separado.

```
selector {
border-top-style: valor;
border-right-style: valor;
border-bottom-style: valor;
border-left-style: valor;
}
```

Los posibles valores para dar estilo a los bordes

none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset

Ejemplo

Vamos a aplicar un estilo a cada uno de los bordes de una lista.

Código

```
<head>
<style type="text/css">
h3{
border-top-style: dotted;
border-right-style: double;
border-bottom-style: dashed;
border-left-style: groove;
}
</style>
</head>

<body>
<h3>Diferentes estilos de bordes</h3>
</body>
```

Diferentes estilos de bordes

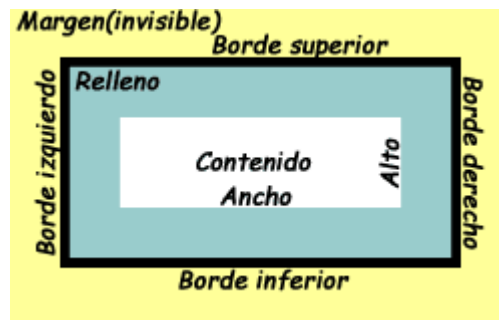
- Tabla de las propiedades de los bordes

Propiedad	Descripción	Valores	Detalles
border	Define ancho, color y estilo general para los 4 bordes. border-top: define ancho, color y estilo para el borde superior border-right: define ancho, color y estilo para el borde derecho border-bottom: define ancho, color y estilo para el borde inferior border-left: define ancho, color y estilo para el borde izquierdo	thin	Fino
		medium	Mediano
		thick	Grueso
		<i>longitud</i>	<i>Longitud</i>
		<i>color</i>	<i>Color</i>
		transparent	Transparente
		none	Nada
		hidden	Oculto
		dotted	Punteado
		dashed	Línea de rayas
		solid	Solido
		double	Doble
		groove	Acanalado
		ridge	En relieve
border-width	Ancho general de los 4 bordes. border-top-width: ancho del borde superior. border-right-width: ancho del borde derecho. border-bottom-width: ancho del borde inferior. border-left-width: ancho del borde izquierdo.	thin	Fino
		medium	Mediano
		thick	Grueso
		<i>longitud</i>	<i>Longitud</i>

Propiedad	Descripción	Valores	Detalles
border-color	Color general de los 4 bordes.	<i>color</i>	<i>Color</i>
	border-top-color: color del borde superior	transparent	Transparente
	border-right-color: color del borde derecho		
	border-bottom-color: color del borde inferior		
	border-left-color: color del borde izquierdo		
border-style	Estilo general de los 4 bordes.	none	Nada
	border-top-style: estilo del borde superior	hidden	Oculto
	border-right-style: estilo del borde derecho	dotted	Punteado
		dashed	Línea de rayas
	border-bottom-style: estilo del borde inferior	solid	Solido
		double	Doble
	border-left-style: estilo del borde izquierdo	groove	Acanalado
		ridge	En relieve
		inset	Recuadro
		outset	Resalte

1.6.5. Márgenes con CSS.

El margen es un espacio invisible alrededor del elemento, que le permite al mismo mantener distancia de otros elementos.



Las propiedades de los márgenes `margin` nos permiten definir el ancho de los mismos.

La propiedad **margin** se utiliza para definir el ancho del espacio que se encuentra entre el borde de un elemento y el elemento cercano a él.

Esta propiedad define el ancho del margen para los cuatro lados de la caja.

Sintaxis

```
<head>
<style="type: text/css">
selector {margin: valor}
</style>
</head>
```

Los posibles valores para definir el ancho de los márgenes

longitud | % | auto

Ejemplo

Vamos a ver cómo se comporta un texto con un margen de 30px alrededor.

Código

```
<head>
<style type="text/css">
p.margin{margin:30px}
</style>
</head>

<body>
<p>En este ejemplo podemos observar que el margen
aleja al elemento de los borde 30 pixels en base al
tamaño de la caja que lo contiene.</p>
</body>
```

En este ejemplo podemos observar que el margen aleja al elemento de los bordes 30 pixels en base al tamaño de la caja que lo contiene.

- Los márgenes de un elemento por separado
`margin-top`, `margin-right`, `margin-bottom`, `margin-left`

Las propiedades **margin-top**, **margin-right**, **margin-bottom**, **margin-left** se utilizan para definir el ancho de los márgenes de cada uno de los lados del elemento por separado.

Puedes definir los 4 lados o solo aquellos que necesites.

Sintaxis

```
<head>
<style="type:text/css">
selector {
margin-top: valor
margin-right: valor
margin-bottom: valor
margin-left: valor
}
</style>
</head>
```

Los posibles valores para definir los anchos de margin

longitud | % | auto

Ejemplo

Vamos a definir el ancho de cada lado por separado.

Código

```
<head>
<style type="text/css">
p{
margin-top:5%
margin-right:30px
margin-bottom:20px
margin-left:50%
}
</style>
</head>

<body>
<p>Este texto tiene definido un ancho de margen
distinto para cada
lado del elemento.</p>
</body>
```

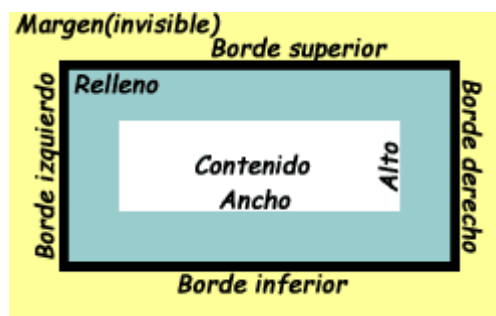
Este texto tiene definido un ancho de margen distinto para cada lado del elemento.

- Tabla de las propiedades de los márgenes

Propiedad	Descripción	Valores	Detalles
margin	Ancho para varios márgenes individuales.	<i>longitud</i>	<i>Longitud</i>
	margin-top: Define el ancho del margen superior.	%	<i>Porcentaje</i>
	margin-right: Define el ancho del margen derecho. margin-bottom: Define el ancho del margen inferior. margin-left: Define el ancho del margen izquierdo.	auto	Automático

1.6.6. CSS Relleno.

Deja un espacio entre el elemento y los bordes del mismo.



Las propiedades del relleno **padding** nos permiten definir el ancho de los mismos.

Esta propiedad define el ancho para los cuatro lados de la caja.

Sintaxis

```
<head>
<style="type: text/css">
selector {padding: valor}
</style>
</head>
```

Los posibles valores para definir los anchos de padding

longitud | %

Ejemplo

Vamos a ver como se comporta un texto con un relleno de 10% alrededor.

Código

```
<head>
<style type="text/css">
p.relleno{padding:10%}
</style>
</head>

<body>
<p>En este ejemplo podemos observar que el relleno
se aleja de los borde un 10% en base al tamaño de la
caja que lo contiene. Al no haber definido los bordes o
los márgenes, el ancho del relleno es la distancia
definida.</p>
</body>
```

En este ejemplo podemos observar que el relleno se aleja de los borde un 10% en base al tamaño de la caja que lo contiene. Al no haber definido los bordes o los márgenes, el ancho del relleno es la distancia definida.

- El relleno de cada lado - padding-top, padding-right, padding-bottom, padding-left

Las propiedades **padding-top**, **padding-right**, **padding-bottom**, **padding-left** se utilizan para definir los anchos de los rellenos de cada uno de los bordes por separado.

Puedes definir los 4 lados o solo aquellos que necesites.

Sintaxis

```
<head>
<style="type: text/css">
selector {
padding-top: valor
padding-right: valor
padding-bottom: valor
padding-left: valor
}
</style>
</head>
```

Los posibles valores para definir los anchos de padding

longitud | %

Ejemplo

Vamos a definir el ancho de cada lado por separado.

Código

```
<head>
<style type="text/css">
p{
padding-top:30px
padding-right:10px
padding-bottom:20px
padding-left:50%
}
</style>
</head>

<body>
<p>Este texto tiene definido un ancho distinto para
cada
lado del elemento.</p>
</body>
```

Este texto tiene definido un ancho distinto para cada lado del elemento.

- **Tabla de las propiedades de padding (relleno)**

Propiedad	Descripción	Valores	Detalles
padding	Tamaños para varios padding individuales.	<i>longitud</i>	<i>Longitud</i>
	<p>padding-top: ancho del padding superior.</p> <p>padding-right: ancho del padding derecho.</p> <p>padding-bottom: ancho del padding inferior.</p> <p>padding-left: ancho del padding derecho.</p>	%	<i>Porcentaje</i>

1.6.7. CSS Fonts – Fuentes.

- **Las fuentes**

Las propiedades de las fuentes nos permiten controlar la apariencia de las mismas.

Posiblemente la definición de las fuentes sea el uso más común de CSS.

Entre los ajustes que podemos aplicar a las fuentes, tenemos:

1. La familia
2. La intensidad
3. El estilo
4. El tamaño
5. La variante
6. La definición general

- **Familia de fuentes - font-family**

Para definir el tipo de fuente usamos la propiedad **font-family**.

Es recomendable usar un tipo de fuente común, que todos los navegadores reconozcan (ej.: Arial, Verdana, Helvetica, sans serif, etc.).

Sintaxis

```
<head>
<style="type: text/css">
selector { font-family: familia de fuente, familia de fuente genérico }
</style>
</head>
```

Nota: debemos utilizar comas entre los valores.

Los posibles valores para definir las familias de fuentes

Las más comunes | arial | Verdana | Helvetica | "Times New Roman" | Courier | Univers

Los posibles valores para definir las familias de fuentes genérico

serif | sans-serif | cursive | fantasy | monospace

Ejemplo

Vamos a definir una familia para una fuente.

Código

```
<head>
<style type="text/css">
p{font-family:"Times New Roman", serif}
</style>
</head>

<body>
<p>La familia de esta fuente es Times New Roman.</p>
</body>
```

La familia de esta fuente es Times New Roman.

- **Intensidad de las fuentes - font-weight**

Una característica muy útil es el control de la intensidad de las fuentes. Para ello utilizamos la propiedad **font-weight**.

Sintaxis

```
<head>
<style="type:text/css">
selector { font-weight: valor}
</style>
</head>
```

Los posibles valores para definir los fondos

normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900

Ejemplo

Vamos a aplicar intensidad a las fuentes del elemento **h2** de esta página.

Código

```
<head>
<style type="text/css">
h2{font-weight:lighter}
</style>
</head>

<body>
<b>La intensidad de este texto es lighter.</b>
</body>
```

La intensidad de este
texto es lighter.

- **Estilo de las fuentes - font-style**

La propiedad **font-style** nos sirve para definir un estilo normal, oblicuo o italic.

Sintaxis

```
<head>
<style="type:text/css">
selector { font-style: valor1}
</style>
</head>
```

Los posibles valores para definir los estilos

normal | italic | oblique

Ejemplo

Vamos a definir el estilo del elemento **h3** de este ejemplo.

Código

```
<head>
<style type="text/css">
h3{font-style:oblique}
</style>
</head>

<body>
<h3>El estilo de este título es oblique.</h3>
</body>
```

El estilo de este título es oblique.

- **Tamaño de las fuentes - font-size**

Controlar el tamaño de las fuentes suele ser de mucha utilidad. La propiedad encargada de eso es **font-size**.

Sintaxis

```
<head>
<style="type: text/css">
selector { font-size: valor}
</style>
</head>
```

Los posibles valores para definir el tamaño de las fuentes

xx-small | x-small | small | medium | large | x-large | xx-large | larger | smaller
| tamaño | %

Ejemplo

Vamos a ver como se comporta cada uno de los valores de la propiedad font-size.

Código

```
<body>
<p style="font-size:xx-small">Tamaño:xx-small</p>
<p style="font-size:x-small">Tamaño:x-small</p>
<p style="font-size:small">Tamaño:small</p>
<p style="font-size:medium">Tamaño:medium</p>
<p style="font-size:large">Tamaño:large</p>
<p style="font-size:x-large">Tamaño:x-large</p>
<p style="font-size:xx-large">Tamaño:xx-large</p>
<p style="font-size:larger">Tamaño:larger</p>
<p style="font-size:smaller">Tamaño:smaller</p>
</body>
```

Tamaño: xx-small

Tamaño: x-small

Tamaño: small

Tamaño: medium

Tamaño: large

Tamaño: x-large

Tamaño: xx-
large

Tamaño: larger

Tamaño: smaller

- Variante de las fuentes - font-variant

Una variación que se le puede dar a las fuentes es el de usar pequeñas mayúsculas, para eso usamos de la propiedad **font-variant**.

Sintaxis

```
<head>
<style="type: text/css">
selector { font-variant: valor }
</style>
</head>
```

Los posibles valores para definir las variantes de las fuentes

normal | small-caps

Ejemplo

Vamos a variar las fuentes de este título **h2**.

Código

```
<head>
<style type="text/css">
h2{font-variant:small-caps}
</style>
</head>

<body>
<b2>El texto esta escrito en minúscula.</b2>
</body>
```

**EL TEXTO ESTA ESCRITO EN
MINÚSCULA.**

- La propiedad de las fuentes - font

La propiedad **font** es la forma comprimida en la cual podemos definir todos los valores de las fuentes de una sola vez. Esta propiedad se aplica a todos los elementos.

Sintaxis

```
<head>
<style="type: text/css">
selector { font: valor1 valor2 valor n }
</style>
</head>
```

Nota: debemos dejar un espacio en blanco entre los valores.

Los posibles valores para definir las fuentes

Todos los valores de *font-family* | *font-style* | *font-variant* | *font-weight* | *font-size* | *caption* | *icon* | *menu* | *message-box* | *small-caption* | *status-bar*

Ejemplo

Vamos a definir un tipo de fuente para **p**.

Código

```
<head>
<style type="text/css">
p{font:large Palatino bold italic}
</style>
</head>

<body>
<p>El tipo de fuente es Palatino large bold en
italic.</p>
</body>
```

*El tipo de fuente es Palatino
large bold en italic.*

- **Propiedades de las fuentes**

Propiedad	Descripción	Valores	Detalles
font	Atajo para establecer el resto de propiedades sobre las fuentes a la vez.	font-style	Estilo de fuente
		font-variant	Variante de fuente
		font-weight	Peso de la fuente
		font-size	Tamaño de la fuente
		font-family	Familia de fuentes
		caption	Fuente a utilizar en los botones, menús desplegables, etc.
		icon	Ícono
		menu	Fuente de los menús desplegables
		message-box	Fuente de las caja de mensajes
		small-caption	Pequeña leyenda
		status-bar	Fuentes de la barra de estado
font-family	Familias de fuentes.	<i>nombre-familia</i>	<i>Nombre de la familia de fuentes</i>
		<i>familia-genérica</i>	<i>Familia genérica</i>
font-style	Estilo de la fuente.	normal	Estilo normal
		italic	Itálica
		oblique	Oblicua
font-variant	Convierte las minúsculas a mayúsculas pero mantienen un tamaño inferior a las mayúsculas.	normal	Normal
		small-caps	Mayúsculas

			pequeñas
font-weight	Intensidad de la fuente.	normal	Normal
		bold	Negrita
		bolder	Negrita mas fuerte
		lighter	Suave
		100	Valor 100
		200	Valor 200
		300	Valor 300
		400	Valor 400
		500	Valor 500
		600	Valor 600
		700	Valor 700
		800	Valor 800
		900	Valor 900
font-size	Tamaño de la fuente.	xx-small	XX-Pequeña
		x-small	X-Pequeña
		small	Pequeña
		medium	Mediana
		large	Grande
		x-large	X-Grande
		xx-large	XX-Grande
		larger	Máxima
		smaller	Mínima
		<i>tamaño</i>	<i>Tamaño</i>
		%	<i>Porcentaje</i>

1.6.8. Los colores en CSS.

Los colores en CSS se pueden indicar de cinco formas diferentes: **palabras clave**, **colores del sistema**, **RGB hexadecimal**, **RGB numérico** y **RGB porcentual**. Aunque el método más habitual es el del RGB hexadecimal, a continuación se muestran todas las alternativas que ofrece CSS.

- **Palabras clave**

CSS define 17 palabras clave para referirse a los colores básicos. Las palabras se corresponden con el nombre en inglés de cada color:

aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow

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		

Además de la lista básica, los navegadores modernos soportan muchos otros nombres de colores.



Nombre	Hexa- gesimal	rgb	Nombre	Hexa- gesimal	rgb	Nombre	Hexa- gesimal	rgb
Pink colors			Green colors			Purple/Violet/Magenta colors		
Pink	FF C0 CB	255 192 203	DarkOliveGreen	55 6B 2F	85 107 47	Lavender	E6 E6 FA	230 230 250
LightPink	FF B6 C1	255 182 193	Olive	80 80 00	128 128 0	Thistle	D8 BF D8	216 191 216
HotPink	FF 69 B4	255 105 180	OliveDrab	6B 8E 23	107 142 35	Plum	DD A0 DD	221 160 221
DeepPink	FF 14 93	255 20 147	YellowGreen	9A CD 32	154 205 50	Violet	EE 82 EE	238 130 238
PaleVioletRed	DB 70 93	219 112 147	LimeGreen	32 CD 32	50 205 50	Orchid	DA 70 D6	218 112 214
MediumVioletRed	C7 15 85	199 21 133	Lime	00 FF 00	0 255 0	Fuchsia	FF 00 FF	255 0 255
Red colors			LawnGreen	7C FC 00	124 252 0	Magenta	FF 00 FF	255 0 255
LightSalmon	FF A0 7A	255 160 122	Chartreuse	7F FF 00	127 255 0	MediumOrchid	BA 55 D3	186 85 211
Salmon	FA 80 72	250 128 114	GreenYellow	AD FF 2F	173 255 47	MediumPurple	93 70 DB	147 112 219
DarkSalmon	E9 96 7A	233 150 122	SpringGreen	00 FF 7F	0 255 127	BlueViolet	8A 2B E2	138 43 226
LightCoral	F0 80 80	240 128 128	MediumSpringGreen	00 FA 9A	0 250 154	DarkViolet	94 00 D3	148 0 211
IndianRed	CD 5C 5C	205 92 92	LightGreen	90 EE 90	144 238 144	DarkOrchid	99 32 CC	153 50 204
Crimson	DC 14 3C	220 20 60	PaleGreen	98 FB 98	152 251 152	DarkMagenta	8B 00 8B	139 0 139
FireBrick	B2 22 22	178 34 34	DarkSeaGreen	8F BC 8F	143 188 143	Purple	80 00 80	128 0 128
DarkRed	8B 00 00	139 0 0	MediumSeaGreen	3C B3 71	60 179 113	Indigo	48 00 82	75 0 130
Red	FF 00 00	255 0 0	SeaGreen	2E 8B 57	46 139 87	DarkSlateBlue	48 3D 8B	72 61 139
Orange colors			ForestGreen	22 8B 22	34 139 34	RebeccaPurple	66 33 99	102 51 153
OrangeRed	FF 45 00	255 69 0	Green	00 80 00	0 128 0	SlateBlue	6A 5A CD	106 90 205
Tomato	FF 63 47	255 99 71	DarkGreen	00 64 00	0 100 0	MediumSlateBlue	7B 68 EE	123 104 238
Coral	FF 7F 50	255 127 80	Cyan colors			White colors		
DarkOrange	FF 8C 00	255 140 0	MediumAquamarine	66 CD AA	102 205 170	White	FF FF FF	255 255 255
Orange	FF A5 00	255 165 0	Aqua	00 FF FF	0 255 255	Snow	FF FA FA	255 250 250
Yellow colors			Cyan	00 FF FF	0 255 255	Honeydew	F0 FF F0	240 255 240
Yellow	FF FF 00	255 255 0	LightCyan	E0 FF FF	224 255 255	MintCream	F5 FF FA	245 255 250
LightYellow	FF FF E0	255 255 224	PaleTurquoise	AF EE EE	175 238 238	Azure	F0 FF FF	240 255 255
LemonChiffon	FF FA CD	255 250 205	Aquamarine	7F FF D4	127 255 212	AliceBlue	F0 F8 FF	240 248 255
LightGoldenrodYellow	FA FA D2	250 250 210	Turquoise	40 E0 D0	64 224 208	GhostWhite	F8 F8 FF	248 248 255
PapayaWhip	FF EF D5	255 239 213	MediumTurquoise	48 D1 CC	72 209 204	WhiteSmoke	F5 F5 F5	245 245 245
Moccasin	FF E4 B5	255 228 181	DarkTurquoise	00 CE D1	0 206 209	Seashell	FF F5 EE	255 245 238
PeachPuff	FF DA B9	255 218 185	LightSeaGreen	20 B2 AA	32 178 170	Beige	F5 F5 DC	245 245 220
PaleGoldenrod	EE E8 AA	238 232 170	CadetBlue	5F 9E A0	95 158 160	OldLace	FD F5 E6	253 245 230
Khaki	F0 E6 8C	240 230 140	DarkCyan	00 8B 8B	0 139 139	FloralWhite	FF FA F0	255 250 240
DarkKhaki	8D 87 6B	189 183 107	Teal	00 80 80	0 128 128	Ivory	FF FF F0	255 255 240
Gold	FF D7 00	255 215 0	Blue colors			AntiqueWhite	FA EB D7	250 235 215
Brown colors			LightSteelBlue	B0 C4 DE	176 196 222	Linen	FA F0 E6	250 240 230
Cornsilk	FF F8 DC	255 248 220	PowderBlue	B0 E0 E6	176 224 230	LavenderBlush	FF F0 F5	255 240 245
BlanchedAlmond	FF EB CD	255 235 205	LightBlue	AD D8 E6	173 216 230	MistyRose	FF E4 E1	255 228 225
Bisque	FF E4 C4	255 228 196	SkyBlue	87 CE EB	135 206 235	Gray/Black colors		
NavajoWhite	FF DE AD	255 222 173	LightSkyBlue	87 CE FA	135 206 250	Gainsboro	DC DC DC	220 220 220
Wheat	F5 DE B3	245 222 179	DeepSkyBlue	00 BF FF	0 191 255	LightGray	D3 D3 D3	211 211 211
BurlyWood	DE B8 87	222 184 135	DodgerBlue	1E 90 FF	30 144 255	Silver	C0 C0 C0	192 192 192
Tan	D2 B4 8C	210 180 140	CornflowerBlue	64 95 ED	100 149 237	DarkGray	A9 A9 A9	169 169 169
RosyBrown	BC 8F 8F	188 143 143	SteelBlue	46 82 B4	70 130 180	Gray	80 80 80	128 128 128
SandyBrown	F4 A4 60	244 164 96	RoyalBlue	41 69 E1	65 105 225	DimGray	69 69 69	105 105 105
Goldenrod	DA A5 20	218 165 32	Blue	00 00 FF	0 0 255	LightSlateGray	77 88 99	119 136 153
DarkGoldenrod	B8 86 0B	184 134 11	MediumBlue	00 00 CD	0 0 205	SlateGray	70 80 90	112 128 144
Peru	CD 85 3F	205 133 63	DarkBlue	00 00 8B	0 0 139	DarkSlateGray	2F 4F 4F	47 79 79
Chocolate	D2 69 1E	210 105 30	Navy	00 00 80	0 0 128	Black	00 00 00	0 0 0
SaddleBrown	8B 45 13	139 69 19	MidnightBlue	19 19 70	25 25 112			
Sienna	A0 52 2D	160 82 45						
Brown	A5 2A 2A	165 42 42						
Maroon	80 00 00	128 0 0						

- **Colores del sistema**

Los colores del sistema son similares a los 17 colores básicos, pero en este caso hacen referencia al color que muestran algunos elementos del sistema operativo del usuario.

Existen varios colores definidos, como por ejemplo **ActiveBorder**, que hace referencia al color del borde de las ventanas activas. La lista completa de colores definidos se puede ver en:

<http://www.w3.org/TR/CSS21/ui.html#system-colors>

Aunque es posible definir los colores en CSS utilizando estos nombres, se trata de un método que casi nunca se utiliza, por lo que se puede considerar prácticamente como una rareza de CSS.

- **RGB hexadecimal**

Aunque se trata del método más complicado de indicar los colores, es el que más se utiliza con mucha diferencia. De hecho, prácticamente todos los sitios web reales utilizan exclusivamente este método. A continuación se muestran los pasos necesarios para definir un color en RGB hexadecimal:

1. Se toman las componentes RGB del color original, por ejemplo R = 71, G = 98, B = 176
2. El valor numérico de cada componente se transforma del sistema numérico decimal al sistema numérico hexadecimal. Esta operación es exclusivamente matemática. En el sistema decimal, se utilizan 10 símbolos para representar los números: del 0 al 9. En el sistema hexadecimal se utilizan 16 símbolos (de ahí su nombre): del 0 al 9 y de la A a la F. Así, en el sistema hexadecimal, después del 9 no va el 10, sino la A. La letra B sería 11, la C sería 12, etc.
3. Si se realiza la conversión hexadecimal de las componentes numéricas anteriores, se obtienen unos nuevos valores: R = 47, G = 62, B = B0.
4. Una vez obtenidas sus componentes hexadecimales, el color se indica concatenando el valor de las componentes y añadiendo el prefijo #. Así, el color anterior en la notación RGB hexadecimal de CSS sería **#4762B0**.

Con esta nueva notación, el color del mismo ejemplo anterior se indica de la siguiente forma:

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

Recuerda que aunque es el método más complicado para definir un color, se trata del método que utilizan la inmensa mayoría de sitios web, por lo que es imprescindible dominarlo. Afortunadamente, todos los programas de diseño gráfico convierten de forma automática los valores RGB decimales a sus valores RGB hexadecimales, por lo que no es necesario realizar a mano estas operaciones matemáticas.

Una de las ventajas del formato RGB hexadecimal es que se pueden comprimir sus valores cuando todas sus componentes son iguales dos a dos:

#AAA = #AAAAAA

#FFF = #FFFFFF

#A0F = #AA00FF

#369 = #336699

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

```
body { background-color: #FFF; color: #000; }
```

```
h1, h2, h3, h4, h5, h6 { color: #C00; }
```

- **RGB decimal o numérico**

En el campo del diseño gráfico, se han definido varios modelos diferentes para referirse a los colores. Los dos modelos más conocidos son RGB y CMYK. Simplificando su explicación, el modelo RGB consiste en definir un color indicando la cantidad de color rojo, verde y azul que se debe *mezclar* para obtener el color. Técnicamente, el modelo RGB es un modelo de tipo "aditivo", ya que se suman colores para obtener el color deseado.

Por lo tanto, en el modelo RGB un color se define indicando sus tres componentes **R (rojo)**, **G (verde)** y **B (azul)**. Cada una de las componentes puede tomar un valor entre cero y un valor máximo. De esta forma, el color rojo puro en RGB se crea mediante el máximo valor de la componente **R** y un valor de **0** para las componentes G y B.

Si todas las componentes valen 0, el color creado es el negro; si todas las componentes toman su valor máximo, el color obtenido es el blanco. En CSS, las componentes de los colores definidos mediante RGB decimal pueden tomar valores entre 0 y 255. El siguiente ejemplo establece el color del texto de un párrafo:

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

La sintaxis que se utiliza para indicar los colores es **rgb()** y entre paréntesis se indican las tres componentes RGB, en ese mismo orden y separadas por comas. El color del ejemplo anterior se obtendría mezclando las componentes **R=71**, **G=98**, **B=176**, que se corresponde con un color azul claro.

Si se indica un valor menor que **0** para una componente, automáticamente se transforma su valor en **0**. Igualmente, si se indica un valor mayor que **255**, se transforma automáticamente su valor a **255**.

- **RGB porcentual**

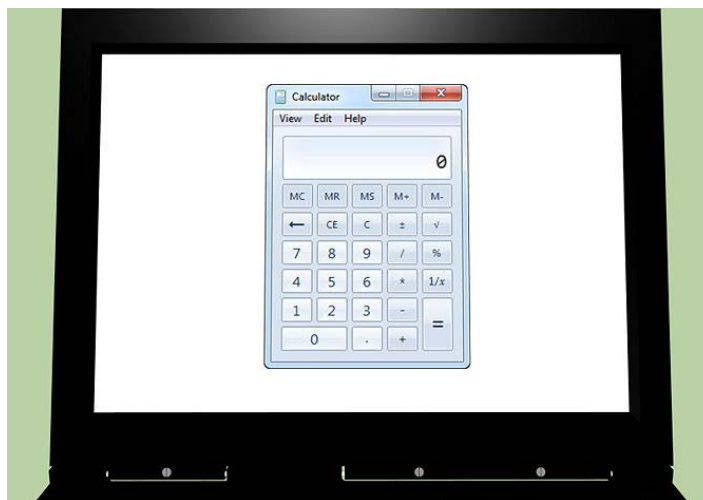
Otra forma de indicar las componentes RGB de un color es mediante un porcentaje. El funcionamiento y la sintaxis de este método es el mismo que el del RGB decimal. La única diferencia en este caso es que el valor de las componentes RGB puede tomar valores entre **0%** y **100%**. El mismo color del ejemplo anterior se puede representar de forma porcentual:

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

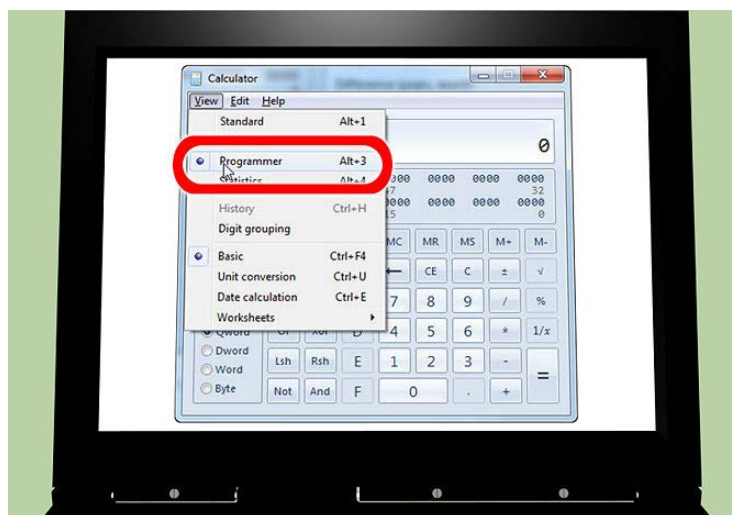
Al igual que sucede con el RGB decimal, si se indica un valor inferior a **0%**, se transforma automáticamente en **0%** y si se indica un valor superior a **100%**, se trunca su valor a **100%**.

- Convertir de decimal a hexagesimal con la calculadora de Windows

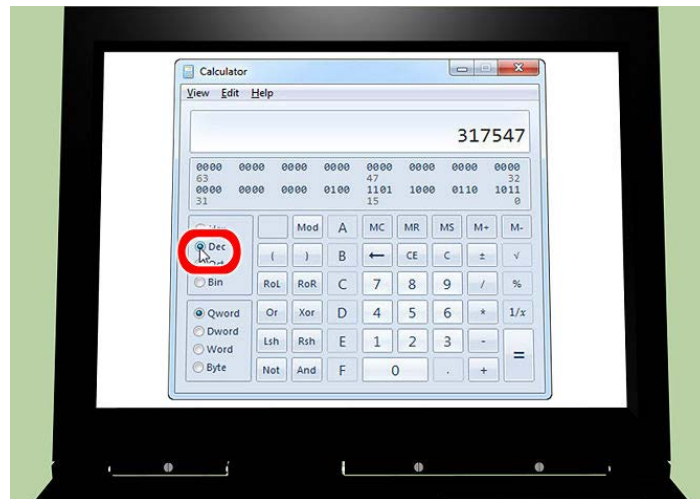
1. En tu computadora con Windows, presiona Inicio, elige 'Accesorios' y luego 'Calculadora'. Una calculadora debe aparecer en tu pantalla.



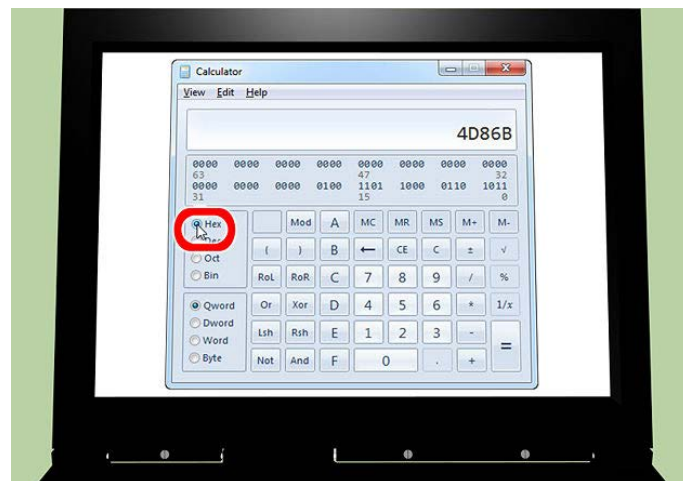
2. En esta ventana de la calculadora selecciona 'Ver' y elige 'Programador'.



3. Después de asegurarte que la opción 'Dec' es elegida.



4. Ahora selecciona la opción 'Hex'. El resultado aparecerá automáticamente.



1.6.9. CSS3. Transparencias en colores RGBA

Ahora, por medio de los colores en RGBA en CSS 3, podremos aplicar nuevas transparencias a los colores que especificamos con CSS, abriendo nuevas posibilidades a los diseñadores sin necesidad de complicarse con pequeños trucos como el uso de imágenes de fondo semitransparentes en PNG, etc. Además, como los colores RGBA se pueden aplicar a cualquier elemento que soporte asignación de color, las aplicaciones aumentan todavía más. **Notación de color RGBA**

Para definir un color RGBA, se deben especificar cuatro valores, de la siguiente manera:

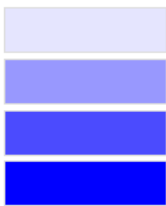
```
rgba(255, 125, 0, 0.5);
```

Los tres primeros valores son números en sistema decimal, que corresponden con los valores de rojo, verde y azul. Siempre tienen que ser números entre 0 y 255.

El cuarto valor es un número entre 0 y 1. Por ejemplo 0 sería totalmente transparente, 1 sería totalmente opaco y 0.5 sería una transparencia al 50%, es decir, mitad opaco mitad transparente.

Colores RGBA con CSS 3

Ejemplo de capas con fondo azul y varias transparencias



Ejemplo de capas con fondo verde y varias transparencias, sobre una capa con fondo amarillo



Ejemplo de capas con fondo naranja y varias transparencias, sobre una capa con una imagen de fondo



1.6.10. Contornos con CSS

Los contornos u **outlines** son líneas que se encuentran alrededor de objetos visuales tales como botones, formularios activos o mapeado de imágenes.

Los contornos se diferencian de los bordes en:

Los contornos no ocupan espacio

Los contornos no necesariamente son rectangulares

Un contorno definido con la propiedad **outline** es dibujado "fuera" de la caja y siempre se encuentra por encima del elemento y no afecta a las cajas anexas a este.

- **El espesor de los contornos**

La propiedad **outline-width** especifica el espesor del contorno. Esta propiedad se puede aplicar a todos los elementos.

Sintaxis

```
<head>
<style="type:text/css">
selector {outline-width: valor}
</style>
</head>
```

Los posibles valores para definir el espesor de los contornos

thin | medium | thick | tamaño (px, pc, pt, mm, cm, in)

Ejemplo

Vamos a aplicar un espesor determinado para el contorno de un botón.

Código

```
<head>
<style type="text/css">
button
{
outline-width: 3px;
outline-style: solid;
}
```

Un botón rectangular con el texto "Enviar" que tiene un contorno sólido de 3 píxeles de espesor.

```
</style>
</head>

<body>
<button>Enviar</button>
</body>
```

Nota:

Para poder visualizar el contorno debemos usar valores de la propiedad **outline-style**

- **El estilo de los contornos**

La propiedad **outline-style** especifica el estilo del contorno. Esta propiedad se puede aplicar a todos los elementos.

Sintaxis

```
<head>
<style="type: text/css">
selector {outline-style: valor}
</style>
</head>
```

Los posibles valores para definir el estilo de los contornos

none | dotted | dashed | solid | double | groove | ridge | inset | outset

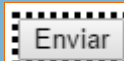
Ejemplo

Vamos a aplicar un estilo determinado para el contorno de un botón.

Código

```
<head>
<style type="text/css">
button
{
outline-style: dotted;
}
</style>
</head>

<body>
```



```
<button>Enviar</button>
</body>
```

- **El color de los contornos**

La propiedad **outline-color** especifica el color del contorno. Esta propiedad se puede aplicar a todos los elementos.

Sintaxis

```
<head>
<style="type:text/css">
selector {outline-color: valor}
</style>
</head>
```

Los posibles valores para definir el estilo de los contornos

color | invert

Ejemplo

Vamos a definir un color determinado para el contorno de un texto.

Código

```
<head>
<style type="text/css">
p
{
outline-color: blue;
outline-style: solid;
}
</style>
</head>

<body>
<p>Texto rodeado de un contorno de
color azul</p>
</body>
```

Texto rodeado de un contorno de color azul

Nota:

Para poder visualizar el contorno debemos usar valores de la propiedad **outline-style**

- **La propiedad general de los contornos**

La propiedad **outline** especifica todas las propiedades de los contornos de una sola vez.

Sintaxis

```
<head>
<style="type: text/css">
selector {outline: valor-1 valor-2 valor-n}
</style>
</head>
```

Los posibles valores para la propiedad outline

thin | medium | thick | tamaño (px, pc, pt, mm, cm, in) | none | dotted | dashed | solid | double | groove | ridge | inset | outset | color | invert

Ejemplo

Vamos a aplicar todos los valores al contorno de un botón.

Código

```
<head>
<style type="text/css">
button{outline: groove red thick}
</style>
</head>

<body>
<button> Enviar </button>
</body>
```



Enviar

Propiedad	Descripción	Valores	Detalles
outline	Propiedades individuales de los contornos. <i>No funciona en IE 5, 6 y 7</i>	outline-color	Color de la línea
		outline-style	Estilo de la línea
		outline-width	Ancho de la línea
outline-width	Ancho del contorno. <i>No funciona en IE 5, 6 y 7</i>	thin	Fino
		medium	Medio
		thick	Grueso
		<i>longitud</i>	<i>Longitud</i>
outline-style	Estilo del contorno. <i>No funciona en IE 5, 6 y 7</i>	none	Nada
		hidden	Oculto
		dotted	Punteada
		dashed	Líneas de rayas
		solid	Solida
		double	Doble
		groove	Acanalado
		ridge	En relieve
		inset	Recuadro
		outset	Resalte
outline-color	Color del contorno. <i>No funciona en IE 5, 6 y 7</i>	<i>color</i>	<i>Color</i>
		invert	Color inverso al color de fondo

1.6.11. Listas con CSS

Las propiedades de las listas nos permiten establecer el estilo de las mismas, la imagen, número o letra de los diferentes Items y la posición de la misma.

- **El tipo de estilo de las listas**

La propiedad **list-style-type** especifica el formato visual de la lista. Esta propiedad se aplica a todo elemento con "display: list-item".

Sintaxis

selector {list-style-type: valor}

Los posibles valores para definir el estilo de las listas

disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | none

Ejemplo

Vamos a aplicar un estilo a una lista.

Código

```
<head>
<style type="text/css">
ol{list-style-type: lower-latin}
</style>
</head>

<body>
<ol>
<li>HTML</li>
<li>CSS</li>
<li>JavaScript</li>
</ol>
</body>
```

a. HTML
b. CSS
c. JavaScript

- **Listas con imágenes**

La propiedad **list-style-image** define la imagen que va a ser usada como marca de cada ítem.

Esta propiedad se aplica a todo elemento con "display: list-item".

Sintaxis

selector {list-style-image: valor}

Los posibles valores para definir las imágenes de las listas

URL | none

Ejemplo

Vamos a aplicar una imagen a una lista.

Código

```
<head>
<style type="text/css">
ul{list-style-image: url("punto.gif")}
</style>
</head>

<body>
<b>ul</b>
<li>XML</li>
<li>VBScript</li>
<li>AJAX</li>
</ul>
</body>
```

- XML
- VBScript
- AJAX

- **La posición de la marca en la lista**

La propiedad **list-style-position** especifica la posición del marcador de los ítems con respecto a la caja de la lista.

Esta propiedad se aplica a todo elemento con "display: list-item".

Sintaxis

```
selector {list-style-position: valor}
```

Los posibles valores para definir la posición de los marcadores

inside | outside

Ejemplo

Vamos a ubicar los marcadores en diferentes posiciones.

Código

```
<head>
<style type="text/css">
ul.dentro{ list-style-position: inside}
ul.fuera{ list-style-position: outside}
</style>
</head>

<body>
<ul class="dentro">
<li>Primer elemento de la lista</li>
<li>Segundo elemento de la lista</li>
<li>Tercer elemento de la lista</li>
</ul>

<ul class="fuera">
<li>Primer elemento de la lista</li>
<li>Segundo elemento de la lista</li>
<li>Tercer elemento de la lista</li>
</ul>
</body>
```

- Primer elemento de la lista
- Segundo elemento de la lista
- Tercer elemento de la lista

- Primer elemento de la lista
- Segundo elemento de la lista
- Tercer elemento de la lista

- **Todas las propiedades de las listas**

La propiedad **list-style** se usa para definir todos los valores de las listas a la vez.

Sintaxis

```
selector {list-style: valor1 valor2 valor-n }
```

Los posibles valores para definir las imágenes de las listas

disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | URL | none | inside | outside

Ejemplo

Vamos a aplicar diferentes propiedades a una lista.

```
<head>
<style type="text/css">
ul{list-style: square inside}
</style>
</head>

<body>
<ul>
<li>XML</li>
<li>VBScript</li>
<li>AJAX</li>
</ul>
</body>
```

- XML
- VBScript
- AJAX

• Tabla de las propiedades de las listas

Propiedad	Descripción	Valores	Detalles
list-style	Permite establecer el estilo de la lista, la imagen y/o la posición.	list-style-type	Tipos de listas
		list-style-position	Posición de la lista
		list-style-image	Imagen de la lista
list-style-type	Estilo aplicable a los marcadores visuales de las listas.	disc	Disco
		circle	Círculo
		square	Cuadrado
		decimal	Nro.decimal
		decimal-leading-zero	Nro.decimal comenzando de 0 No funciona en IE 5, 6 y 7
		lower-roman	Nro.romano minúscula
		upper-roman	Nro.romano mayúscula
		lower-greek	Letra griega

			minúscula No funciona en IE 5, 6 y 7
		lower-latin	Letra latina minúscula No funciona en IE 5, 6 y 7
		upper-latin	Letra latina mayúscula No funciona en IE 5, 6 y 7
		armenian	Letra armenia No funciona en IE 5, 6 y 7
		georgian	Letra gregoriana No funciona en IE 5, 6 y 7
		lower-alpha	Letra alfabeto en minúscula
		upper-alpha	Letra alfabeto en mayúscula
		none	Nada
list-style-image	Imagen aplicable a los elementos de las listas.	URL	URL
		none	Nada
list-style-position	Posición dentro de la lista de los elementos marcadores de las listas.	inside	Dentro
		outside	Fuera

1.6.12. Tablas con CSS

Las propiedades de las tablas nos permiten definir el comportamiento, el diseño y la ubicación de los elementos que componen cada tabla.

Los temas que estudiaremos:

1. La ubicación del título
2. El formato de las tablas
3. El modelo de los bordes
4. El espacio entre celdas

5. El comportamiento de las celdas vacías

- **Ubicación del título**

La propiedad **caption-side** nos permite ubicar el título de la tabla por encima o por debajo de la misma. La alineación horizontal del mismo se puede establecer con la propiedad **text-align**.

Sintaxis

selector {**caption-side**: **valor**}

Los posibles valores para definir la ubicación del título

top | bottom | inherit

Ejemplo

Vamos a ubicar el título de la tabla por debajo de la misma.

Código

```
<head>
<style type="text/css">
caption{caption-side: bottom}
</style>
</head>

<body>
<table border="1">
<caption>Precio de los
lácteos</caption>
<tr>
<th>Producto</th>
<th>Precio</th>
</tr>
<tr>
<td>Manteca</td>
<td>4.00</td>
</tr>
<tr>
<td>Leche</td>
<td>1.50</td>
</tr>
</table>
</body>
```

Precio de los lácteos

Producto	Precio
Manteca	4.00
Leche	1.50

Nota:

Internet Explorer ubica el título siempre por encima de la tabla

- **Formato de las tablas**

La propiedad **table-layout** se usa para diseñar las filas, columnas o celdas de una tabla. Entre las posibilidades de diseño podemos definir si las mismas van a tener el tamaño fijo que estipulemos (**fixed**) o se adecuarán al contenido sin importar la medida que hayamos establecido(**auto**).

Sintaxis

```
selector { table-layout: valor }
```

Los posibles valores para definir el formato de las tablas

auto | fixed | inherit

Ejemplo

Vamos a comparar las 2 propiedades de una tabla.

```
<head>  
<style type="text/css">  
table.auto{table-layout: auto}  
table.fija{table-layout: fixed}  
</style>  
</head>  
  
<body>  
<table class="auto" style="border:  
solid; width: 100%">  
<caption>Tabla con formato automático  
</caption>  
<tr>  
<td width="10%">  
11111111111111111111111111111111</td>  
<td width="40%">222222222222</td>  
<td width="50%">33333</td>  
</tr>  
</table>  
  
<table class="fija" style="border: solid;  
width: 100%">  
<caption>Tabla con formato fijo</caption>  
<tr>  
<td width="10%">  
11111111111111111111111111111111</td>  
<td width="40%">222222222222</td>  
<td width="50%">33333</td>  
</tr>  
</table>  
</body>
```

Tabla con tamaño automático

[illegible]

Tabla con tamaño fijo

[illegible]

- **Modelo de los bordes**

La propiedad **border-collapse** nos permite seleccionar la apariencia de los bordes de cada celda de la tabla. Existen 2 modelos diferentes de bordes: separados y continuos.

Sintaxis

```
selector {border-collapse: valor}
```

Los posibles valores para los diferentes modelos de bordes

Ejemplo

Vamos a comparar los dos modelos de bordes.

Código

```
<head>
<style type="text/css">
table.plegado{ border-collapse: collapse}
table.separado{ border-
collapse: separate}
</style>
</head>

<body>
<table class="plegado" style="border: solid
1px">
<caption>Tabla con los bordes
plegados</caption>
<tr>
<th>Nombre</th>
<th>Edad</th>
</tr>
<tr>
<td>Luis</td>
<td>23</td>
</tr>
</table>
<table class="separado" style="border: solid
1px">
<caption>Tabla con los bordes
separados</caption>
<tr>
<th>Nombre</th>
<th>Edad</th>
</tr>
<tr>
<td>Roberto</td>
<td>19</td>
</tr>
</table>
</body>
```

Tabla con los bordes plegados

Nombre	Edad
Luis	23

Tabla con los bordes separados

Nombre	Edad
Roberto	19

- Espacio entre celdas

La propiedad **border-spacing** especifica la separación entre celdas adyacentes. Si especificamos un solo valor, este actúa sobre toda la tabla. Si especificamos 2 valores el primero define la separación horizontal y el segundo la vertical.

Sintaxis

selector { **border-spacing**: valor }

Los posibles valores para definir la separación entre celdas

distancia(horizontal) distancia(vertical) | inherit

Ejemplo

Vamos a definir la separación de las celdas.

Código

```
<head>
<style type="text/css">
table{border-spacing: 10px 20px; border-collapse:
separate}
</style>
</head>

<body>
<btable style="border: solid 1px"&>
<tr>
<th>Marca del automovil</th>
<th>Modelo</th>
</tr>
<tr>
<td>Ford</td>
<td>Mustang</td>
</tr>
<tr>
<td>Toyota</td>
<td>Corolla</td>
</tr>
<b</table>
</body>
```

Marca del automovil	Modelo
Ford	Mustang
Toyota	Corolla

- Comportamiento de las celdas vacías

La propiedad **empty-cells** nos permite controlar la visualización de los bordes y fondos de una celda vacía.

Sintaxis

selector { **empty-cells**: valor }

Los posibles valores para controlar las celdas vacías

show | hide | inherit

Ejemplo

Vamos a comparar el comportamiento de los dos valores.

Código

```
<head>
<style type="text/css">
td.muestra{empty-cells: show}
td.oculta{empty-cells: hide}
</style>
</head>

<body>
<table style="border-collapse: separate; border solid
1px">
<tr>
<th>Nombre</th>
<th>Edad</th>
<th>Estado civil</th>
</tr>
<tr>
<td>Carlos</td>
<td class="oculta"></td>
<td>casado</td>
</tr>
<tr>
<td>Julieta</td>
<td>27</td>
<td class="muestra"></td>
</tr>
</table>
</body>
```

Nombre	Edad	Estado civil
Carlos		casado
Julieta	27	





- **Propiedades de las tablas**

Propiedad	Descripción	Valores	Detalles
caption-side	Posición del título respecto de la tabla.	top	Superior No funciona en IE 5, 6 y 7
		bottom	Inferior No funciona en IE 5, 6 y 7
table-layout	Control del algoritmo usado para el formato de las celdas, filas y columnas.	auto	Automático
		fixed	Fijo
border-collapse	Selección del modelo de los bordes.	collapse	Plegado
		separate	Separado
border-spacing	Espaciado entre los bordes de celdas adyacentes.	<i>longitud</i>	<i>Longitud</i> No funciona en IE 5, 6 y 7
empty-cells	Visibilidad de los bordes de celdas sin contenido.	show	Muestra No funciona en IE 5, 6 y 7
		hide	Oculto No funciona en IE 5, 6 y 7

1.7. Prefijos CSS de Navegadores (webkits).

Se llaman prefijos de navegador o prefijos comerciales (vendor prefixes) a un prefijo que se antepone a una regla CSS destinado a que dicha regla sea leída y aplicada exclusivamente por un navegador concreto (por ejemplo Chrome) pero no por el resto de navegadores. El uso de prefijos suele aplicarse a propiedades que se encuentran en fase experimental o que aún no se han convertido en un estándar.

Al igual que los comentarios condicionales que se han venido usando específicamente para Microsoft Internet Explorer, los prefijos son un tipo de filtro que permite que una instrucción CSS se aplique específicamente a un navegador o familia de navegadores pero no a los demás. Sin embargo, a diferencia de los comentarios condicionales, existen prefijos específicos para todos los tipos de navegador.

Prefijo	Familia de navegadores a los que aplica	
-webkit-	Chrome, Safari, Android, iOS	
-moz-	Firefox	
-o-	Opera	
-ms-	Microsoft Internet Explorer	

- **Buen uso y mal uso de prefijos**

A la hora de usar propiedades experimentales y aplicar prefijos conviene plantearse cuáles son las ventajas y los inconvenientes existentes para cada caso particular. Supongamos que simplemente trataba de aplicar una imagen con gradiente de color al fondo de una caja en una página web. ¿Qué ventajas e inconvenientes presentaría el uso de la propiedad linear-gradient?

Ventajas:

- Posiblemente el tiempo de carga de la página sea más rápido usando esta propiedad que usando una imagen. Con esta propiedad el navegador simplemente tiene que renderizar (dibujar) a partir de una instrucción, mientras que con una imagen es necesario descargar el archivo y cada descarga de archivo implica un pequeño consumo de tiempo.

Inconvenientes:

- Al ser una propiedad con poca trayectoria histórica las diferentes versiones de navegadores requieren de distintas sintaxis, lo que obliga a la repetición de varias líneas de código con el mismo fin.

- Podemos tener dudas de que la visualización vaya a ser buena en la mayor parte de dispositivos y navegadores, ya que algunos de ellos (en especial los más antiguos) no reconocerán esta propiedad.

1.7.1. Columnas en css

• PROPIEDAD COLUMN-COUNT

PROPIEDAD CSS column-count	
Función de la propiedad	Permite definir el número de columnas con que se debe mostrar el contenido dentro de un elemento.
Valor por defecto	auto
Aplicable a	Contenedores como elementos block, table, table-cell, etc.
Valores posibles para esta propiedad	Un número entero igual o superior a 1
	inherit (se heredan las características del elemento padre).
Ejemplos	.myContainer { column-count: 3; } .myContainerSP { column-count: 5; }

• PROPIEDAD COLUMNS

Esta propiedad es un shorthand que permite especificar el número de columnas bien en forma de número de columnas (lo que sería equivalente a usar column-count), bien indicando una unidad de medida (lo que sería equivalente a usar column-width).

Ejemplo: myBox { columns: 3; }

• PROPIEDAD COLUMN-WIDTH

PROPIEDAD CSS column-width	
Función de la propiedad	Permite sugerir un ancho de columna deseado, aunque su aplicación no será estricta si existe column-count, que induce un ancho basado en el número de columnas especificado.
Valor por defecto	auto
Aplicable a	Contenedores como elementos block, table, table-cell, etc.
Valores posibles para esta	auto (el número de columnas derivará del establecido con column-count)

PROPIEDAD CSS <code>column-width</code>	
propiedad	Una unidad de medida válida en CSS
	inherit (se heredan las características del elemento padre).
Ejemplos	<code>.myContainer { column-width: 150px; } .myContainerSP { column-width: 5em; }</code>

En presencia de `column-count`, puede omitirse esta propiedad ya que `column-count` induce un ancho de columna basado en el número de columnas indicado. Puede entrar en conflicto con `column-count` si los valores indicados son incompatibles entre sí. En ausencia de `column-count`, puede inducir un número de columnas en base al ancho especificado. Si nos fijamos, `column-count` y `column-width` vienen siendo dos formas de expresar lo mismo.

- **PROPIEDAD COLUMN-GAP**

Esta propiedad sirve para definir un espacio de separación entre columnas. Ejemplo: `column-gap: 20px;`

En algunos navegadores es necesario el uso de prefijo específico de navegador.

- **PROPIEDAD COLUMN-RULE-WIDTH, COLUMN-RULE-STYLE Y COLUMN-RULE-COLOR**

Estas propiedades tienen por finalidad establecer el ancho, style y color de la línea de separación entre columnas.

`column-rule-width` funciona de forma análoga a `border-width` (valor por defecto `medium`, y valore posibles `thin`, `medium`, `thick` ó una unidad de medida válida CSS).

`column-rule-style` funciona de forma análoga a `border-style`.

`column-rule-color` funciona de forma análoga a `border-color`.

- **PROPIEDAD SHORTAND COLUMN-RULE**

Esta propiedad permite establecer los valores de `column-rule width`, `column-rule-style` y `column-rule-color` en una sola línea. Ejemplo: `column-rule: 3px solid blue;`

1.7.2. Esquinas redondeadas en css

Las cajas CSS son rectangulares y cuando aplicamos propiedades como un color o imagen de fondo, se aplican sobre la caja rectangular. Sin embargo, para hacer los diseños web más atractivos los diseñadores usan esquinas redondeadas. Este efecto antiguamente no era fácil de conseguir y había que recurrir a utilizar imágenes de fondo con transparencia y redondeadas u otras técnicas.

CSS 3 incorpora nuevas propiedades para el control de bordes de los elementos. Ahora se permiten bordes con las esquinas redondeadas, bordes con imágenes (incluso varias imágenes se pueden utilizar para definir el aspecto del borde), sombras, etc.

Tenemos la propiedad `border-radius`, que permite definir bordes redondeados en las esquinas, especificando las medidas del radio que deben darse a la curva de las esquinas. Su uso sería aproximado al que vemos a continuación:

`border-radius: 5px;`

Definiría un radio de 5 píxeles en el redondeo de las esquinas del elemento. Por el momento Mozilla ha adoptado este atributo con un nombre especial, que es válido para productos como Firefox, mientras que las especificaciones de CSS3 no hayan alcanzado el estado "Candidate Recommendation", que es cuando se supone que los distintos navegadores deben implementarlas. El nombre del atributo por el momento es:

`-moz-border-radius`

Los navegadores basados en WebKit, como Google Chrome o Safari, también soportan las esquinas redondeadas de CSS 3, pero el atributo `border-radius` tampoco funciona directamente, como en el caso de Firefox, sino que hay que utilizar un "alias": **`-webkit-border-radius`**.

`DIV {`

`border: 1px solid #000000;`

`-moz-border-radius: 7px;`

`-webkit-border-radius: 7px;`

`padding: 10px;`

`}`

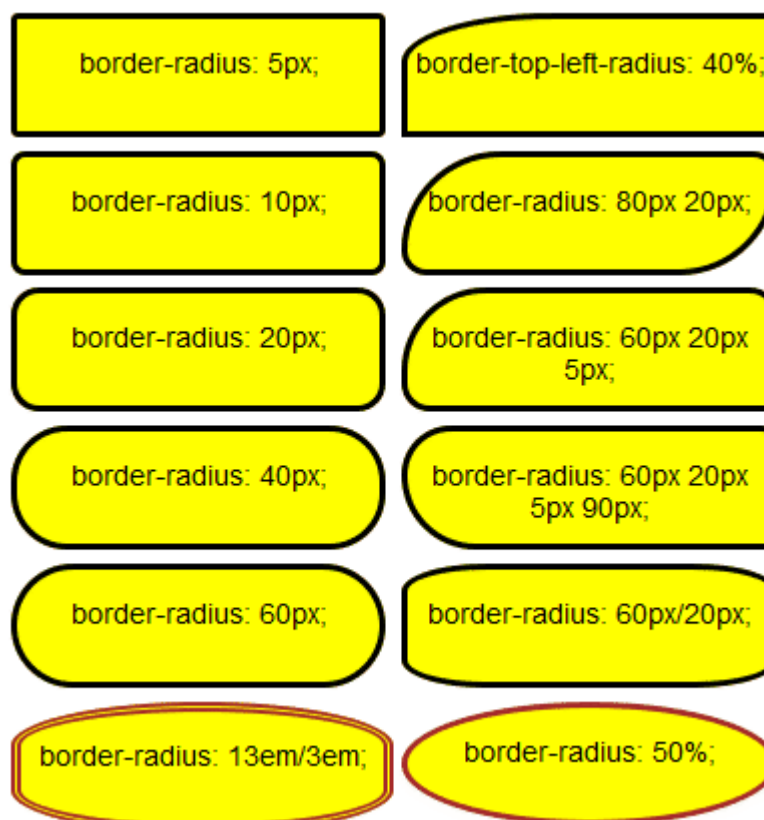
Con esto conseguimos que todos los div tengan un borde redondeado en las esquinas de radio de 7 píxeles. Fijarse en el uso de los atributos `-moz-border-radius` y `-webkit-border-radius`, que sirven para lo mismo, pero en navegadores basados en Mozilla y basados en WebKit.

Pero el atributo `border-radius` tiene otras posibles configuraciones, en la que se pueden definir los valores para el radio de las cuatro esquinas por separado. De esta manera:

`-moz-border-radius: 7px 27px 100px 0px;`

Así estaríamos definiendo un borde redondeado con radio de 7 píxel para la esquina superior izquierda, luego 27px para la esquina superior derecha, de 100px para la inferior derecha y 0px para la inferior izquierda. (Hay que explicar que un `border-radius` de 0px es un borde con esquina en ángulo recto)

Los bordes redondeados con CSS 3, como se podrá imaginar, sólo se ven si se tiene definido algún borde visible al elemento que se los asignamos, ya sea `solid`, `dotted`, etc.



1.7.3. Efecto sombra en css

La propiedad box-shadow facilita la introducción de sombras y efectos 3D. Este efecto antiguamente no era fácil de conseguir y se solía recurrir al uso de imágenes u otras técnicas.

- **Atributo box-shadow**

El atributo box-shadow requiere varios valores para especificar las características de la sombra, como difuminado, separación de la sombra y la propia caja o color. La sintaxis es como esta:

box-shadow: 5px -9px 3px #000;

Por orden de aparición, los valores que se indican en box-shadow son:

- **Desplazamiento horizontal de la sombra:** La sombra de un elemento suele estar un poco desplazada con respecto al elemento que la produce y su posición será en función del ángulo con el que llegue la luz. En el caso de este ejemplo el primero de los valores, 5px, quiere decir que la sombra aparecerá 5 píxeles a la derecha. Si la sombra quisiéramos que apareciera un poco hacia la izquierda del elemento original que la produce, pondríamos un valor negativo a este atributo. Cuanto más desplazamiento tenga una sombra, el elemento que la produce parecerá que está más separado del lienzo de la página.
- **Desplazamiento vertical de la sombra:** El segundo valor que colocamos en el atributo box-shadow es el desplazamiento vertical de la sombra con respecto a la posición del elemento que la produce. Este valor es similar al desplazamiento horizontal. Valores positivos indican que la sombra aparecerá hacia abajo del elemento y valores negativos harán que la sombra aparezca desplazada un poco hacia arriba. En el caso del anterior ejemplo, con -9px estamos indicando que la sombra aparecerá desplazada 9 píxeles hacia arriba del elemento.
- **Difuminado:** El tercer valor indica cuánto queremos que esté difuminado el borde de la sombra. Si el difuminado fuera cero, querría decir que la sombra no tiene ningún difuminado y aparece totalmente definida. Si el valor es mayor que cero, como en nuestro

ejemplo 3px, quiere decir que la sombra tendrá un difuminado de esa anchura, 3 píxeles en el ejemplo.

- **Color de la sombra:** El último atributo que se indica en el atributo box-shadow es el color de la sombra. Generalmente las sombras en el mundo real tienen un color negro o grisáceo, pero con CSS3 podremos indicar cualquier gama de color para hacer la sombra, lo que nos dará bastante más versatilidad a los diseños gracias a la posible utilización de sombras en distintos colores, que puedan combinar mejor con nuestra paleta.

- **Atributo box-shadow para navegadores basados en Mozilla, como Firefox:**

De manera temporal, Firefox es capaz de interpretar el atributo -moz-box-shadow, por ejemplo:

-moz-box-shadow: 1px 1px 0px #090;

Atributo box-shadow para navegadores basados en WebKit, como Safari o Google Chrome: En estos momentos y de manera temporal, navegadores como Chrome o Safari entienden el atributo: -webkit-box-shadow, por ejemplo:

-webkit-box-shadow: 3px 3px 1px #fc8;

Como podremos imaginar, si deseamos ampliar al máximo la compatibilidad con box-shadow, necesitaríamos indicar tanto el propio atributo box-shadow (que funciona en Opera y en el futuro funcionará en todos los navegadores), así como -moz-box-shadow y -webkit-box-shadow para que funcione en las versiones actuales de Firefox, Safari, Chrome, etc.

- **Ejemplos de Sombras CSS3**

Ahora veamos varios ejemplos de sombras creadas directamente con CSS 3 y el atributo box-shadow, con sus variantes para compatibilidad temporal en los navegadores Mozilla o WebKit.

```
#cajasombra{
```

```
background-color: #ddd;
```

```
width: 300px;
```

```
padding: 10px;
```

```
box-shadow: 5px 5px 0 #333;  
-webkit-box-shadow: 5px 5px 0 #333;  
-moz-box-shadow: 5px 5px 0 #333;  
}
```

Esto crearía una capa con un gris claro como color de fondo y una sombra desplazada abajo y a la derecha en 5 píxeles y sin difuminado. Además, hemos definido un color de sombra gris oscuro para el elemento.

```
#sombraclara{  
width: 200px;  
padding: 10px;  
background-color: #999;  
color: #fff;  
  
box-shadow: 2px 2px 2px #ffc;  
-webkit-box-shadow: 2px 2px 2px #ffc;  
-moz-box-shadow: 2px 2px 2px #ffc;  
}
```

Este otro ejemplo es para una sombra un poco menor, también desplazada hacia abajo y a la derecha y con un difuminado de 2 píxeles. Además hemos indicado un color amarillo claro para la sombra, por lo que, para verla bien, tendríamos que colocar este elemento sobre un fondo oscuro.

```
#sombreadondeada{  
background-color: #090;  
color: #fff;  
width: 400px;  
padding: 10px;  
-moz-border-radius: 7px;  
-webkit-border-radius: 7px;  
  
box-shadow: 15px -10px 3px #000;  
-webkit-box-shadow: 15px -10px 3px #000;
```

```
-moz-box-shadow: 15px 10px 3px #000;
```

```
}
```

En este tercer ejemplo tenemos un caso curioso de sombra, pues está aplicada sobre un elemento que tiene las esquinas redondeadas con CSS 3. Así pues, la sombra también debe tener las sombras redondeadas, para ajustarse al elemento que la produce. Ambos navegadores con compatibilidad a sombras y CSS 3 funcionan correctamente con sombras y bordes redondeados.

Esta es una capa con sombra creada con CSS 3.

Esta caja la pongo sobre fondo negro, para hacer un elemento con sombra clara.

Esta capa tiene las esquinas redondeadas y la sombra se adapta a la forma de la capa, por lo que tendrá esquinas redondeadas.

1.7.4. Transformaciones con CSS

Las **transformaciones** de un elemento utilizando sólo CSS son uno de los proyectos de la versión 3 que aún se discute y que algunos navegadores ya implementan con ciertas variantes. En teoría, será tan simple como escribir cualquier otra propiedad:

```
transform: funcion(parámetros);
```

En los navegadores se usan de este modo:

```
-moz-transform: funcion(parámetros); // en Mozilla
```

```
-webkit-transform: funcion(parámetros); // en Safari y Chrome
```

```
-o-transform: funcion(parámetros); // en Opera
```

- No tienen equivalencias en Internet Explorer aunque allí hay un filtro llamado Matrix que permitiría simularlas.

En principio, hay varias funciones:

- **scale(x,y)**

Aumenta o disminuye el tamaño del elemento; su valor normal es 1. Valores superiores aumentan su tamaño e inferiores lo disminuyen así, 1.5 hará que se vea un 50% más grande y 0.5 un 50% más chico; si sólo se coloca un valor, este se aplica a ambas direcciones, caso contrario, el primero indica el ancho y el segundo el alto.

scaleX(x) y **scaleY(y)** son lo mismo pero sólo afectan al ancho (X) o al alto (Y).

por ejemplo:

```
-moz-transform:scale(2); -webkit-transform:scale(2); -o-transform:scale(2);
```

- **rotate(a)**

Gira un elemento. El valor se expresa en grados, si es positivo gira en el sentido de las agujas del reloj, si es negativo, en el sentido contrario.

```
-moz-transform:rotate(10deg); -webkit-transform:rotate(10deg); -o-transform:rotate(10deg);
```

- **skew(a, b)**

Skew significa sesgar o sea, inclinar y eso hace esta propiedad que también utiliza ángulos como valor, el primero afecta al eje X (horizontal) y el segundo al eje Y (vertical).

skewX(a) y **skewY(a)** hacen lo mismo pero sólo sobre uno de los ejes.

```
-moz-transform:skew(5deg); -webkit-transform:skew(5deg); -o-transform:skew(5deg);
```

- **translate(x, y)**

Desplaza el elemento, el primer valor lo hace en el eje horizontal y el segundo en el eje vertical.

translateX(x) y **translateY(y)** hacen lo mismo pero sólo sobre uno de los ejes.

```
-moz-transform:translate(20px); -webkit-transform:translate(20px); -o-transform:translate(20px);
```

1.7.5. CSS Media Queries

Las Media Queries sirven para definir estilos diferentes para distintos tamaños de la pantalla. Son sencillas de entender y aplicar, aunque el estándar es bastante sofisticado, con diversas posibilidades. Existen muchos usos, algunos no tan habituales en el mundo del diseño actual, pero que podrán tener su protagonismo en algún momento.

- **Alternativas para implementar Media Queries**

Aunque la utilidad es novedosa, la sintaxis es parecida a lo que ya conocemos de las CSS, por lo que nos resultará bien sencilla.

Para producir Media Queries debemos tener siempre en mente la expresión condicional, aquella que debe cumplirse para que se apliquen ciertos estilos. Además la expresión condicional puede tener incluso varias condiciones, usando operadores lógicos como "and" para combinarlas. De modo que las circunstancias que se deban cumplir para aplicar unas reglas CSS sean de lo más variadas.

- **Alternativa 1:**

La primera alternativa de las Media Queries es a través del atributo media de la etiqueta LINK. Como sabemos, esa etiqueta es la que se usa para enlazar una hoja de estilo con un documento HTML. En ese enlace podemos especificar condicionales que deben cumplirse para que los estilos enlazados se apliquen. Por ejemplo, que se esté imprimiendo un documento o que la pantalla tenga cierta anchura mínima.

Recordamos, la etiqueta LINK tiene esta forma:

```
<link rel="stylesheet" href="estilos-generales.css">
```

Pues ahora simplemente le podemos agregar el atributo "media" indicando la condición que se debe cumplir para que estos estilos se apliquen:

```
<link rel="stylesheet" href="estilo-imprimir.css" media="print">
```

Este atributo media="print" quiere decir que los estilos deben aplicarse sólo cuando la página se están mostrando para la impresión.

```
<link rel="stylesheet" href="estilo-pantallas-grandes.css" media="(min-width:1200px)">
```

Este uso será seguramente más novedoso para ti. Quiere decir que esos estilos deben aplicarse sólo cuando la pantalla del usuario (En caso de ordenadores de escritorio, la ventana del navegador) tenga una anchura mínima de 1200 píxeles.

El problema de escribir tus Media Queries así es que tienes archivos de CSS separados. Es decir, aquellos estilos para impresión o para pantallas de 1200px están en archivos independientes, lo que es sencillo de administrar para nosotros, pero una mala práctica en términos de optimización de la web, puesto que se tienen que realizarse varias solicitudes al servidor distintas para traerse los CSS. En la práctica ralentiza la carga de la página en relación a hacer una única solicitud de un archivo CSS que contenga todo el código de los estilos.

- **Alternativa 2:**

Este método que vamos a ver ahora es más interesante y es el que se usa habitualmente a nivel profesional. Consiste en incorporar los estilos en una construcción @media donde se apliquen entre llaves todos los estilos que queremos para una consulta de medio dada.

```
@media (min-width: 500px) {  
  h1{  
    margin: 1%;  
  }  
  .estiloresponsive{  
    float: right;  
    padding-left: 15px;  
  }  
}
```

Como puedes ver, tenemos la sentencia @media en la que podemos indicar entre paréntesis las condiciones que deben cumplirse para que se aplique esta media query. En este caso será para pantallas que tengan una anchura mínima de 500 píxeles.

Luego entre llaves colocamos todas las reglas y atributos de estilos CSS que necesitemos aplicar en esta situación. Las reglas de estilos son las mismas que pondrías fuera de la estructura condicional. Cuando la sentencia entre paréntesis se evalúe como verdadera, se aplicarán todas ellas.

- **Operadores lógicos para las Media Queries**

Para combinar diversas condiciones podemos usar los operadores lógicos, de una manera similar a como se usan en lenguajes de programación. Los que tenemos disponibles son:

- **Operador and:** las dos condiciones deben cumplirse para que se evalúe como verdadera.
- **Operador not:** es una negación de una condición. Cuando esa condición no se cumpla se aplicarán las media queries.
- **Operador only:** se aplican las reglas solo en el caso que se cumpla cierta circunstancia.
- **Operador or:** no existe como tal, pero puedes poner varias condiciones separadas por comas y cuando se cumpla cualquiera de ellas, se aplicarán los estilos de las media queries.
-

```
@media (max-width: 600px) and (orientation: landscape) {  
  h1{  
    color: red;  
  }  
}
```

Esta regla se aplicaría para pantallas con una anchura máxima de 600 píxeles y cuando la orientación está en horizontal.

Nota:

Ten en cuenta que la mayoría de smartphones simulan tamaños de pantalla mayores, haciendo una especie de dimensiones virtuales que faciliten la lectura de webs que no están diseñadas para Responsive Web Design. Por ello, lo más seguro es que tengas que poner el "viewport" en el documento HTML a algo como:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

Sin ese viewport tu móvil simularía que tiene unas dimensiones de unos 980 píxeles, cuando quizás la pantalla solo tenga una anchura real de 320 o 500 píxeles. Internet Explorer ubica el título siempre por encima de la tabla

Para ver en funcionamiento la parte del orientation (landscape o portrait), tienes que usar un móvil o tablet y cambiar la posición de la pantalla, para que esté en horizontal o vertical.

Nota:

Si tienes un ordenador que no reconozca el cambio de orientación la posición siempre será considerada será landscape.

```
@media tv and (min-width: 1200px){  
  h1{  
    margin: 10%;  
  }  
}
```

Esta regla aplicaría en dispositivos de tipo televisión y cuya resolución mínima de anchura sea de 1200 píxeles.

```
@media (min-width: 600px), handheld and (orientation: portrait) {  
  h1{  
    color: green;  
  }  
}
```

Este mediaquery servirá para pantallas de minimas dimensiones 600 píxel y también para todos aquellos dispositivos handheld que estén en posición vertical.

Nota:

Handheld es un término inglés que sirve para especificar aquellos dispositivos que son de mano. Pequeños ordenadores que se llevan en la mano, como los palm o agendas electrónicas que aparecieron antes de popularizarse los smartphones o tablets. Por mis pruebas handheld no aplica a los móviles o tablets.

Las Media Queries tienen muchas más posibilidades, pero sin duda con lo que ahora sabes podrás resolver el 98% de las necesidades que te puedas encontrar en el mundo del diseño web.

2. Diseño, ubicación y optimización de los contenidos de una página web.

2.1. Creación de un documento funcional.

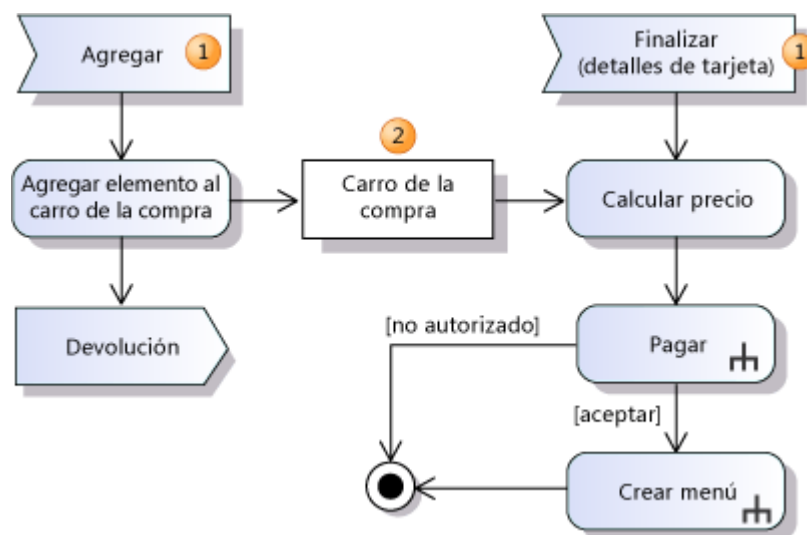
A la hora de enfrentarnos con un proyecto Web debemos de tener en cuenta algunas consideraciones que se alejan mucho de empezar “ya” con el editor de turno Dreamweaver, Frontpage, Golive etc, más bien debemos de empezar por pulsar el botón de nuestro ordenador para apagarlo y sentarnos enfrente de una hoja en blanco para plantear el proyecto. De cómo realicemos esos primeros bocetos en esa primera hoja y de cómo pensemos cada una de los elementos que se destacan a continuación depende el éxito del sitio y con ello la satisfacción de nuestro cliente.

- ¿Qué es el funcional de un sitio web?

El funcional de un site es un documento técnico donde se especifican todos los procesos y comportamientos previstos en el back-end (es decir, los componentes que se ponen en marcha como resultado de la interacción del usuario). En otras palabras, es un análisis del comportamiento de navegación que puede tener un determinado usuario dentro del web y que forma parte de la arquitectura de la información.

El funcional de un sitio web se elabora con la participación del cliente, que es quien conoce con detalle los objetivos del proyecto. Asimismo, hay que tener en cuenta que sólo se aconseja llevarlo a cabo en los casos en que los proyectos web adquieren una complejidad elevada, dado que deben ofrecer una considerable cantidad de funcionalidades. Un buen ejemplo de ello puede ser una tienda online, donde se despliega un amplio abanico de opciones ante el consumidor.

Para llevar a cabo el funcional de una web, se utilizan **técnicas de UML** (*unified modeling language*), lo que en castellano se conoce como LMU (lenguaje unificado de modelado. Mediante las mismas, se puede generar un **diagrama de casos de uso**, con todas las funciones que puede tener el sitio web o *software* en cuestión. De este modo, y gracias al esquema resultante, podemos observar cómo se desarrolla el proceso de comunicación y cuál es el comportamiento de un sistema, todo ello mediante su interacción con los usuarios y a través de las posibilidades que brindan sus funcionalidades.



Ejemplo de diagrama UML para una tienda online. En él se reflejan las funcionalidades del sitio y los diferentes comportamientos que se pueden dar.

Dado que estos diagramas describen cómo se comporta un usuario final (por ejemplo, si hace clic en un determinado botón o apartado, cómo realiza las búsquedas o si responde a una llamada a la acción), y cómo utiliza el sistema, estos recursos son una referencia para conocer qué requisitos debe desarrollar el equipo de programación para el site en cuestión.

Sin embargo, cuando se trata de proyectos web de menor complejidad, no se precisa ni de funcionales ni de diagramas UML de casos de uso. Basta con un documento escrito donde se especifiquen las prestaciones principales y sus finalidades. En él, hay que detallar aspectos como:

- cuál debe ser el comportamiento de un buscador interno.
- cómo debe funcionar un slideshow o carrusel
- dónde colocar un menú desplegable.

En resumen, debemos considerar el funcional como un instrumento para reflexionar sobre las necesidades del proyecto y pulir pequeños detalles. Asimismo, también es útil a la hora de plantear la programación del sitio. Para poder elaborar un análisis funcional eficiente y poderlo tomar como base de trabajo, es indispensable la participación activa del cliente, que es quien conoce los objetivos precisos del proyecto.

A continuación, se detalla el **índice** de una posible **especificación de requisitos de un documento funcional**:

1. Objetivos del sitio web.
2. Estructura y diseño del sitio web:
 - a. La página de inicio.
 - b. Páginas secundarias.
 - c. Menú principal de navegación.
 - d. Menú secundario de navegación.
3. Tipología de los usuarios:
 - a. Usuario no registrado.
 - b. Usuario registrado.
 - c. Usuario colaborador.
 - d. Usuario experto.
 - e. Usuario administrador.
 - f. Usuario superadministrador.
4. Flujo de trabajo.
5. Desarrollo del motor de búsqueda.
6. Usabilidad de la plataforma.
7. Accesibilidad de la plataforma.
8. Copia de seguridad de los contenidos.
9. Posicionamiento en buscadores: gestión de las palabras clave.

2.2. Diseño de los contenidos.

Es hora de empezar a diseñar, al menos **wireframes**.

Para ello, primero hace falta entender y estudiar un poco más a los usuarios, saber qué necesitan y cómo el diseño puede responder a sus necesidades.

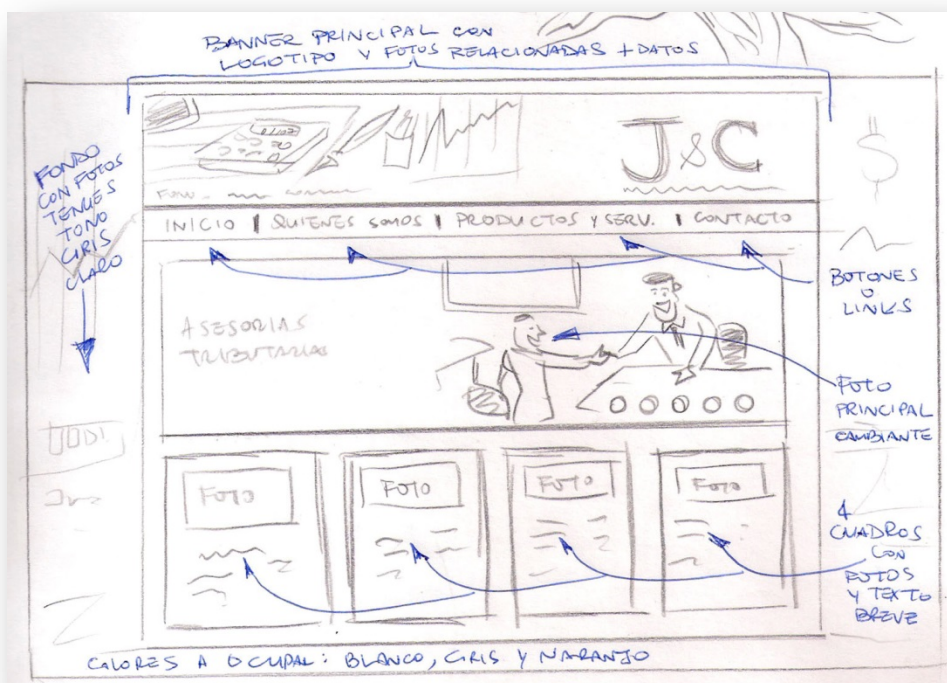
Conocer a los usuarios permite diseñar una aplicación que tenga en cuenta sus motivaciones, necesidades y problemas, como eje a partir del cual construir una propuesta. Este conocimiento no se basa en suposiciones y teorías, sino en estudios que ayuden a determinar el perfil de los usuarios de la aplicación. «Personas» y «Viaje del usuario», entre otras, son algunas de las metodologías utilizadas para conseguirlo.

Un buen estudio previo, nos podría ayudar a definir los siguientes pasos en la creación de nuestra app: **arquitectura de la información, wireframe, testeo en papel, mockups, diseño de la interfaz y testeo del prototipo digital**.

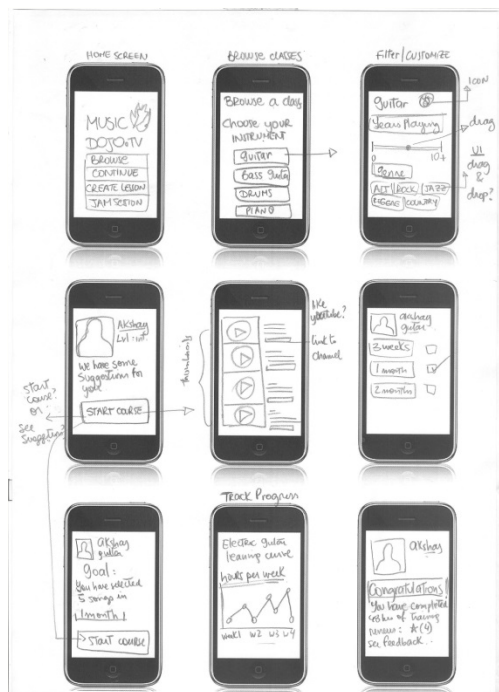
Muchas veces en el desarrollo de productos y/o aplicaciones se cae en el error de unificar o tratar como igual el concepto de **Wireframes, Mockups y prototipo digital** y de usarlos como sinónimos.

- **Diseño de wireframes.**

Plasmar en un papel ya sea a mano o de forma digital el esquema y funcionalidades de una web o una app.



Ejemplo de Boceto web o wireframe



Ejemplo de Wireframe de un diseño de una web móvil o app.

- **Diseño de mockups**

Esta manera de crear un prototipo, consiste en utilizar archivos digitales de plantillas que contengan los elementos básicos de interacción y de interfaz — botones, listas y cabeceras—, para construir en el ordenador, utilizando programas de diseño como photoshop, pantallas básicas a partir de estos componentes.

En dispositivos móviles y dependiendo del sistema operativo, se puede echar mano de diferentes recursos, siendo aquellos de iOS los más difundidos, seguidos de cerca por los de Android.



Ejemplo de Mockups de un diseño web

- **El prototipo digital**

Generalmente, se trata de maquetas con una interacción suficiente para poder navegar entre las diferentes pantallas.

Pueden estar basados en wireframes o en diseños visuales, y su fidelidad puede ser mayor o menor dependiendo de cuanto se corresponda su apariencia y comportamiento con la versión final esperada de la aplicación.

No es necesario hacer un prototipo que contenga todas las pantallas posibles de una aplicación. Los prototipos están destinados a pruebas, por lo tanto, deben desarrollarse sólo aquellas pantallas necesarias para completar de principio a fin la tarea que se quiere probar. Por ejemplo, si se quiere sondear un registro de usuario, el prototipo debería contemplar las pantallas necesarias desde el ingreso de datos hasta el mensaje de éxito final.



Ejemplo de prototipo digital de un diseño web

- **Algunas herramientas gratuitas para la creación de bocetos**
 - **Pencil Project**

Permite crear diagramas y realizar interfaces de usuarios, con la ayuda de una gran paleta de herramientas. Ofrece numerosos elementos gráficos (plantillas) para construir el esquema de las páginas y también permite:

- La creación de páginas múltiples.
- La creación de enlace entre las páginas.
- La edición de texto directamente en el boceto.

- La exportación a los formatos HTML, PNG, OpenOffice.org, Word y PDF.
- Operaciones estándares: alineación, rotación, cambio de tamaño.

[Pencil Project](#)

<http://pencil.evolus.vn/Default.html>

- o **Lumzy**

Esta herramienta colaborativa de creación de bocetos online (prototipos de sitio web) ofrece varios elementos gráficos (botones, listas, ventanas de texto, barras de navegación, figuras geométricas, etc.), para disponerlos libremente en un espacio de trabajo. Soporta por supuesto la creación de páginas múltiples (pestañas separadas).

Una ventana ofrece herramientas de formato asociada a cada elemento gráfico (alineación, colores...) para facilitar la personalización del proyecto.

El creador del proyecto puede invitar a terceros a participar en el proyecto para colaborar en tiempo real.

Lumzy integra por otro lado una herramienta de retoque de imágenes, una mensajería instantánea, y numerosos botones de acciones estándares (duplicación de ítems, superposición, bloqueo de los iconos registrados)

[Lumzy](#)

<http://lumzy.com/>

- o **MockFlow**

En versión gratuita, esta herramienta de wireframing online es abierta sólo para proyectos de 4 páginas, con 2 colaboradores máximo. Una configuración suficiente para la creación de pequeños sitios web (escaparate).

Pone a disposición numerosos elementos que hay que insertar en el boceto para esquematizar numerosos proyectos: aplicación web, formulario de contacto, software, etc.

[MockFlow](#)

<http://www.mockflow.com/>

- o **Cacoo**

Esta herramienta de creación de diagramas online polivalente ofrece numerosos elementos para incluir en el boceto y funcionalidades para crear de manera fácil los mapas del sitio web (sitemaps). También permite a varios usuarios trabajar juntos y simultáneamente en el mismo proyecto.

[Cacoo](#)

<https://cacoo.com>

2.3. Identificación de la información que se tiene que ubicar en la página web.

Uno de los grandes hándicaps a la hora de empezar a generar un proyecto web es el de identificar qué información debe aparecer en el proyecto y cual no.

A la hora de tomar esta decisión, nos puede ayudar el contestar o reflexionar sobre cada una de las cuestiones siguientes y de la forma más precisa posible.

- **De qué trata mi página. (¿El qué?).**

Breve descripción de los contenidos de la página, su título principal...

- **Finalidad que persigo al hacerlo. (¿Por qué).**

Fines informativos, comerciales, entretenimiento, comunicación, reclutar, formación etc.

- **Análisis de Audiencia. (¿Para quién?).**

Describir cual es mi público objetivo, nivel informático idiomas, problemas físicos etc.

- **Imagen corporativa. (Cómo es).**

Que lenguaje, sistemas de iconos, colores, fuentes... voy a utilizar, dependiendo a quien va a ir dirigido.

- **Horizonte. (Cuánto).**

Tiempo de vida de la página, establecer periodos de actualización, etc.

2.4. Selección de contenidos para cada elemento de la página.

Antes de elaborar cualquier sitio web, es preciso seguir un plan ordenado.

- **Delimitación del tema:** de qué va a tratar el sitio web.
- **Delimitación de contenidos.**

- **Recolección de la información:** recopilar y seleccionar la información que se va a incluir.
- **Agregación:** hacer un balance equilibrado entre linealidad y jerarquización. Creación de páginas, nodos, secciones y subsecciones.
- **Estructuración de los contenidos:** unión de los diferentes nodos y páginas teniendo en cuenta la jerarquización y ordenación de los contenidos. Creación de nodos de meta-información sobre otros nodos y enlaces que permitan la estructuración horizontal y vertical. Creación de la página inicial y de las páginas principales. En esta etapa se definen tanto las estructuras jerárquicas y horizontales, como las taxonomías y esquemas de clasificación.

2.5. Uso del documento funcional para las especificaciones del diseño.

Un documento funcional con una especificación de requisitos es un documento que describe todas las características que debe cumplir el sitio web que va a ser desarrollado, con el fin de garantizar su cumplimiento antes de la finalización del mismo.

Cuando el desarrollo del sitio web es contratado a una empresa externa esta especificación debe incluirse como parte del contrato para garantizar su cumplimiento. Por tanto, nunca se firmará un contrato que no tenga asociada la especificación de requisitos del sitio web.

La redacción de una especificación de requisitos requiere una profunda reflexión sobre los objetivos que pretendemos alcanzar con el sitio web. En función de su naturaleza, podemos identificar diferentes tipos de requisitos:

- **Requisitos funcionales del sitio web:** estos requisitos se obtendrán a partir de los intereses manifestados tanto por el responsable del sitio web, como de las personas que tendrán que interactuar directamente con él. Para extraer esta información es conveniente entrevistarse con todas las partes involucradas en la gestión y desarrollo del sitio web. Algunas de las preguntas que habrán de responderse en esta fase son:
 - **¿Cuál es el objetivo del sitio web?**
 - **¿Qué tipo de usuarios tendrá?**
 - **¿Qué tareas llevarán a cabo a los distintos tipos de usuarios?**

- **Requisitos técnicos:** son aquellos requisitos que garantizan la calidad del desarrollo informático del sitio web. Concretamente tendremos que supervisar:
 - **Administración y mantenimiento del sitio web:** atendiendo a los requisitos funcionales habrá que decidir qué gestor de contenidos vamos a utilizar. En estos momentos los dos gestores de contenido (de código no propietario) más utilizados son Joomla (caracterizado por su sencillez) y Drupal (caracterizado por la potencia de sus herramientas de comunicación). Es muy importante que el gestor de contenidos tenga definidos los tipos de usuarios (con sus correspondientes permisos) que se hayan decidido (en la especificación de requisitos funcionales), y que permita el desarrollo del flujo de trabajo que se ha previsto. También es esencial que permita la realización de copias de seguridad de los contenidos.
 - **Codificación y formato de los contenidos del sitio web:** se hará diferenciando contenidos y formato, y respetando escrupulosamente los estándares que existen a tal efecto, en este momento XHTML (versiones Transitional o Strict) para la especificación de los contenidos y CSS para la especificación del formato. La utilización de otros lenguajes se hará sólo cuando su uso o la naturaleza del contenido lo justifique (para mejorar la apariencia de los menús, incluir alguna animación o video, reproducir un fichero de audio, etc.). También es conveniente intentar prever el tipo de contenidos multimedia que vamos a incluir en nuestro sitio, y así implementar la tecnología que permita su reproducción desde nuestra web, o en su defecto facilitar el enlace a los plugins que el usuario necesitará para su visualización (aunque en la medida de lo posible se desaconseja esta última opción). Por último, no debemos olvidar mencionar en la especificación de requisitos la necesidad de que el sitio web sea desarrollado con una codificación que lo haga funcional y usable en los principales navegadores (principalmente en Internet Explorer y Mozilla Firefox).
 - **Arquitectura del sitio web:** la organización de los contenidos en el sitio web deberá ser coherente. Especialmente relevante será contar con buenos menús de navegación. Un sitio web tendrá una buena navegación si no necesitamos acudir a los botones de navegación del navegador para desplazarnos por él. Para la correcta comprensión e implementación de la arquitectura es muy importante la elaboración de prototipos que ilustren la organización de los contenidos.
 - **Usabilidad:** el uso del sitio web debe resultar sencillo y cómodo a todos sus usuarios.
 - **Accesibilidad:** en la medida que sea posible habrá de garantizarse un nivel mínimo de accesibilidad para los usuarios con necesidades especiales. Si el sitio web está siendo desarrollado para una

institución pública (española) se ha de tener presente que, por ley, deberá cumplir un nivel de accesibilidad AA conforme a la especificación del WAI.

- **Posicionamiento:** tanto la arquitectura del sitio, como el gestor implementado, deberán facilitar la gestión de los contenidos (y muy especialmente de los metadatos) de manera que podamos potenciar las palabras clave para las que deseamos posicionarnos.
- **Otras consideraciones:** a estos aspectos generales habría que sumar aquellos propios de nuestro sitio web, como podría ser: desarrollo del buscador interno, creación y mantenimiento de un tesoro, etc.
- **Creación de los sistemas de navegación y búsqueda:** creación de páginas guía, ayudas a la navegación, tablas de contenido, índices, sumarios, mapas de navegación, glosarios, páginas de búsqueda, uso de iconos y barras de navegación, utilización de metáforas, etc. accesibles desde cualquier otra página del sitio web.
- **Diseño y estilo gráfico:** estilos y formatos textuales, coherencia gráfica, diseño de fondos y distribución de los elementos dentro de la página, inclusión de material multimedia, cantidad y tamaño de las imágenes, etc. Dar homogeneidad y coherencia a todo el sitio web. Utilizar metáforas orgánicas, funcionales y visuales.
- **Ensamblaje final:** últimos enlaces, diseño de portadas y estilos gráficos, logotipos, enlaces sobre autoría, contacto, fechas de creación o de actualizaciones, etc.
- **Evaluación y test de uso:** comprobación del funcionamiento, vínculos y páginas rotas, usabilidad, accesibilidad, últimos ajustes, etc.

2.6. Tipos de página para la ubicación de contenidos.

Un sitio web se compone de contenidos que pueden ser desarrollados en diversos formatos: texto, vídeo, audio, fotografías, etc. Si estos elementos forman parte de un listado que no tiene ningún orden, obligamos al usuario a recorrer una lista, en la que todos los elementos aparecen mezclados. Si solo tenemos cuatro contenidos web quizá la cosa funcione, puede que el usuario encuentre por casualidad lo que está buscando, pero esta opción no parece la más operativa, sobre todo si estamos hablando de docenas, cientos o millones de contenidos.

Por tanto, necesitamos organizar esa información de alguna manera. Lo ideal es que diseñemos una forma de agrupación de contenidos con la que el mayor porcentaje de usuarios pueda encontrar lo que necesita lo antes posible y con el menor esfuerzo, dependiendo de sus particulares necesidades de información.

Hay múltiples formas de agrupar la información, un sitio web puede utilizar solo una o varias de esas formas para conseguir que sus contenidos se encuentren:

Algunos ejemplos:

- **Agrupaciones exactas.** Un contenido web solo puede pertenecer a una categoría y no a otra:
 - **Por orden cronológico**
 - **Por ubicación geográfica**
- **Agrupaciones ambiguas.** Un contenido web podría pertenecer a más de una categoría. Que esté en una u otra dependerá de muchos factores, lo que quiera el dueño de la web, lo que aporte un análisis de usuarios, etc.
 - **Por jerarquías**
 - **Por facetas**
 - **Por tareas**
 - **Por audiencias**
 - **Por prominencia:** contenidos destacados, por ejemplo «lo que sale en la página de inicio» frente a los «contenidos de base».
 - **Híbridos:** se mezclan varios tipos que no se deberían usar en un mismo espacio porque pueden desorientar al usuario.

La creación agrupación o clasificación de contenidos se puede hacer mediante, fundamentalmente, dos estrategias:

- **De arriba abajo (top-down):** primero se deciden las categorías principales y, después, cada una se va desglosando en subcategorías.
- **De abajo arriba (bottom-up):** primero se listan las subcategorías o temáticas que se van a tratar o para las cuales se tiene información y luego se van agrupando en función de nuestros objetivos y de cómo creemos que serán los modelos mentales más probables de los usuarios; es decir, cómo creemos que la mayoría de los usuarios va a buscar el contenido web.

2.7. Definición de los tipos de página según los contenidos y sus funcionalidades.

Existen varias clasificaciones de páginas web pero nosotros abordaremos el tema desde el punto de vista de sus funciones, por ser considerado el aspecto más importante:

- **Páginas Web Transaccionales, e-comercio o e-commerce**

Bajo este modelo, las empresas exhiben sus productos en la página para que los clientes los compren a través de internet. El principal objetivo de éstas páginas es el de realizar ventas o transacciones online pero un segundo objetivo muy común es el de alojar información sobre sus productos. Tal vez el mejor ejemplo de éste tipo de sitio es Amazon, una de las tiendas online más grandes y con mayor presencia a nivel mundial.

- **Páginas Web Orientadas al Servicio**

Esta modalidad está orientada a brindar información sobre productos y servicios y no poseen el famoso “carrito de compra”, por lo que los productos no pueden ser comprados directamente a través de la página. Este tipo de página tiene como principal objetivo estimular a los clientes para que terminen el proceso de compra offline. Un segundo objetivo en este caso es el de construir relaciones con clientes (servicio al cliente). También pueden generarle al departamento de ventas de la empresa datos de posibles compradores para que finalicen la operación. Estas páginas son comúnmente utilizadas por proveedores de servicios, profesionales y todo tipo de negocio que exija que el proceso de venta termine en persona.

- **Páginas Web dirigidas a la Construcción de una Marca**

La página web ofrece una experiencia para afianzar la imagen de la marca. Por lo general, los productos no pueden ser comprados en línea pero muchas veces tienen la opción de comprar material POP de la marca. Este tipo de sitios es muy utilizado por marcas de productos de consumo masivo y de precios bajos como bebidas, alimentos, cigarrillos, productos de higiene personal, entre otros, y su objetivo fundamental es el de fidelizar la marca entre sus clientes. Un ejemplo es la página web de CocaCola.

- **Páginas Web de Contenido**

El principal producto de este tipo de páginas es su contenido. Éstas proveen información a sus usuarios para que accedan a través de ella a los 3 primeros tipos de páginas descritos anteriormente vía motores de búsqueda, directorios, noticias, publicidad, patrocinios y programas de afiliados, entre otros. Estas páginas generan ingresos de distintas formas pero la vía más popular actualmente es la publicidad.

- **Páginas Web 2.0**

- **Un blog** , en español también bitácora digital, cuaderno de bitácora, ciber bitácora, ciber diario, o web blog, o weblog es un sitio web en el que uno o varios autores publican cronológicamente textos o artículos, apareciendo primero el más reciente, y donde el autor conserva siempre la libertad de dejar publicado lo que crea pertinente. También suele ser habitual que los propios lectores participen activamente a través de los comentarios. Un blog puede servir para publicar ideas propias y opiniones de terceros sobre diversos temas.

Los términos ingleses blog y web blog provienen de las palabras web y log ('log' en inglés es sinónimo de diario).

- **Un Wiki** (del hawaiano wiki wiki, «rápido») es un sitio web colaborativo que puede ser editado por varios usuarios. Los usuarios de una wiki pueden así crear, editar, borrar o modificar el contenido de una página web, de una forma interactiva, fácil y rápida; dichas facilidades hacen de una wiki una herramienta efectiva para la escritura colaborativa.

- **Comunidades, Foros o Redes Sociales**

Estos portales están dedicados a estimular la interacción y la generación y el intercambio de información entre sus usuarios. Tal vez el caso más notable de ésta categoría es Facebook, pero también existen casos de comunidades, foros o redes sociales, mucho más pequeños, que forman parte de una empresa o página web particular que han demostrado traer consigo muchos beneficios para los negocios.

2.8. Selección del Tipo de página para la página web.

Hagámoslo con un ejemplo:

Imaginemos una página corporativa de una micropyme. Debe preguntarse: ¿qué información requiere de mí un cliente potencial en mi sitio web para tomar la decisión de solicitar mis servicios o comprar mis productos?

Puede necesitar tan solo mostrar a sus clientes potenciales una definición de a qué se dedica la empresa, una localización física de la misma, los servicios o productos que oferta, quizá un portfolio de clientes y la información legal necesaria, así como aportar algún medio para que el cliente pueda comunicarse con nosotros.

Solo con esto podemos listar los contenidos:

- **La empresa**
- **Nuestros servicios/nuestros productos**
- **Información Legal**
- **Localización geográfica**
- **Contáctanos**
- **Nuestros cliente**
- **A qué nos dedicamos**

Ahora resulta que tenemos muchos productos distintos, o tenemos clientes de muchos tipos o de sectores distintos. Tendremos que listarlos también, lo que nos lleva a empezar a sumar muchos contenidos:

- **La empresa**
- **Localización geográfica**
- **Producto 1**
- **Cliente1**
- **Cliente 2**
- **Sector 1**
- **Producto 2**
- **Información Legal**
- **Cliente 3**
- **Nuestros servicios / nuestros productos**
- **Sector 2**
- **Cliente 4**
- **Producto 3**
- **Nuestros clientes**
- **Cliente 5**
- **Producto 4**
- **Contáctanos**
- **Sector 3**
- **Sector 4**
- **A qué nos dedicamos**

Esto empieza a desmadrarse. Ahora lo que tendremos que hacer es organizar esto de alguna manera, pensando en que le sea fácil a nuestro cliente encontrar lo que necesite. Por ejemplo:

- **La empresa**
- **A qué nos dedicamos**
- **Localización geográfica**
- **Contáctanos**
- **Nuestros clientes**
 - **Sector 1**
 - **Cliente 1**
 - **Sector 2**
 - **Cliente 2**
 - **Sector 3**
 - **Cliente 3**
 - **Sector 4**
 - **Cliente 4**
 - **Cliente 5**
- **Nuestros servicios/nuestros productos**
 - **Producto 1**
 - **Producto 2**

- **Producto 3**
- **Producto 4**
- **Información Legal**

Obviamente podemos jugar con estas agrupaciones como necesitemos, solo es un ejemplo. Pero este tipo de estructuración le está dando a nuestro cliente la información de forma mucho más clara que con el listado inicial, que no tenía ningún orden.

Entre muchas otras, podemos usar técnicas muy sencillas, como el card sorting, que nos permitirán ver cómo los usuarios creen que deberían estar organizadas esas secciones. En el caso de grandes portales, podemos incluso recurrir a técnicas de organización por coocurrencias, etc., pero para empezar, usemos el sentido común, y preguntemos a nuestros usuarios.

Ahora, alinear las necesidades de búsqueda del usuario con las del sitio web, ya es otro cantar. Lógicamente, queremos que además de encontrar lo que buscan, lo compren. De eso se ocupan el diseño de interacción y el estudio de la conducta de uso de nuestros clientes potenciales, antes y después de publicar el sitio web.

2.9. Uso del documento funcional para las especificaciones del diseño.

Es importante tener en cuenta que la unidad básica de información de un documento hipertextual no es la página, sino la pantalla.

Así pues, el diseño de la página y la disposición de los elementos dentro de ella para ser vistos en pantalla, son uno de los aspectos principales a la hora de diseñar el hiperdocumento. Las páginas deben tener un esquema ordenado y legible de un vistazo.

He aquí dos ejemplos de disposición de página, la imagen de la izquierda muestra una página ilegible y desordenada, mientras que la imagen de la derecha, que sigue un esquema ordenado, facilita la navegación y comprensión del contenido.



En el documento funcional también deberá quedar constancia que en el encabezamiento de los documentos es imprescindible que aparezca el título destacado y el uso de gráficos sensibles o botones de cabecera que indiquen los recorridos posibles para orientar la navegación.

También es corriente la utilización de un logotipo u otro sello gráfico que identifique la imagen institucional u oficial, comercial, etc. del sitio web.

En el diseño de las páginas hay que tener en cuenta una serie de factores como: enlaces locales y ayudas a la navegación, encabezamiento de documentos, tipografía (contraste visual, esquema y diseño de páginas, tipos de letras, establecimiento de títulos y subtítulos, etc.). Creación de pies de página con información sobre el autor, e-mail de contacto, enlaces a otras páginas relacionadas, fechas de creación y actualización, etc.

2.10. Especificaciones de navegación

Muchos de los sitios Web que encontramos en Internet presentan una serie de enlaces hacia sus diferentes secciones a lo largo de todas sus páginas, como una manera de mostrar al usuario los contenidos del sitio. Estos menús de navegación pueden presentarse en diferentes formatos y organización, ya sean

simples listas de opciones, sistemas gráficos, sistemas de pestañas, menús desplegables, etc

Uno de los principios básicos de la Web estriba en el hecho de que los usuarios pueden moverse libremente por las diferentes secciones que componen un sitio Web o entre los “infinitos” volúmenes de información diseminados por millones de páginas Web distribuidas en servidores de todo el mundo. Esta intrínseca característica de Internet provoca que en numerosas ocasiones el usuario se sienta perdido en el ciberespacio y sienta incapacidad de encontrar el camino hacia la información que realmente le interesa. Tal y como comenta **Jakob Nielsen en el libro “Usabilidad. Diseño de sitios Web”** esta sensación de pérdida se produce cuando el usuario:

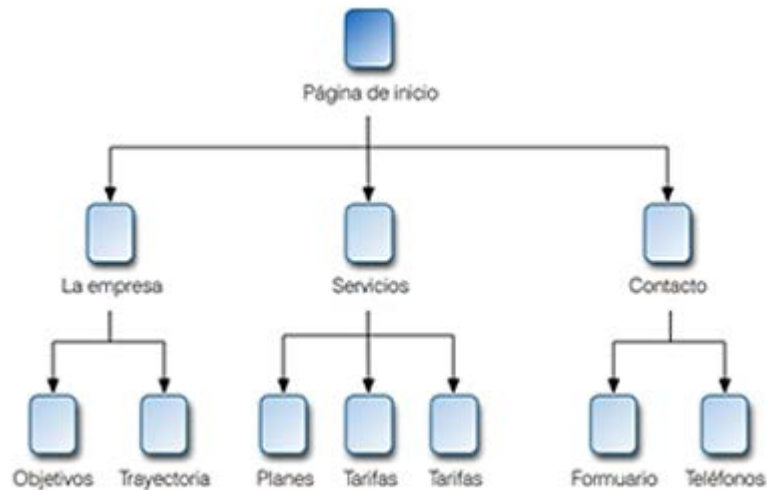
- **No sabe dónde está:** El usuario desconoce su situación actual en relación a la estructura global de la información del sitio Web y no percibe la relación que hay entre la página actual y el resto de páginas.
- **No sabe dónde ha estado:** El usuario desconoce la ruta de navegación que ha seguido hasta la posición actual y por consiguiente no es capaz de identificar las páginas ya visitadas.
- **No sabe a dónde puede ir:** El usuario no puede identificar los enlaces que contienen información relacionada con la página actual. Este problema de orientación se ha intentado resolver mediante la búsqueda de sistemas que permitan al usuario ubicarse y desplazarse a través de las estructuras de información de una manera fácil y eficaz.

Los sistemas de navegación, compuestos por los diferentes enlaces a las diferentes secciones de un sitio Web, permiten ubicarnos y desplazarnos a través de las estructuras de la información, facilitando a los usuarios saber en cada momento dónde están, dónde pueden ir y cómo está organizada la información. Existen diversas formas de organizar estos enlaces dentro de una página Web y según los autores Louis Rosenfeld y Peter Morville en el libro "Arquitectura de la información para la World Wide Web"² podemos diferenciar entre cuatro tipos diferentes de sistemas de navegación: Sistemas de navegación jerárquicos, Sistemas de navegación globales, Sistemas de navegación locales y Sistemas de navegación ad hoc.

2.11. Creación de un mapa de navegación de páginas.

Los mapas de navegación proporcionan una representación esquemática de la estructura del hipertexto, indicando los principales conceptos incluidos en el espacio de la información y las interrelaciones que existen entre ellos. Un mapa es, por ejemplo, una representación completa (o resumida) del sitio web para orientar al lector/usuario durante el recorrido o para facilitarle un acceso directo al lugar que le interese. Reflejará la estructura del web por medio de enlaces a

los nodos principales, y éstos también pueden desarrollarse para mostrar los subnodos. El mapa de navegación puede representarse bien en forma textual, bien en forma gráfica, o una combinación de ambas.



- **Google Sitemaps**

Google Sitemaps es una herramienta que la compañía Google pone a disposición de los webmaster registrados para una mejor búsqueda y posicionamiento en su buscador. Al crear un Sitemap, Google puede rastrear más fácilmente los contenidos, además de proporcionar estadísticas de acceso y posibles errores de rastreo del robot de indexación o araña.

El Sitemap no es más que un archivo con un listado de las páginas e información sobre su contenido, actualizaciones, etc. Es importante saber que el archivo puede tener varios formatos, aunque el más utilizado en la actualidad sea el .XML, ya que nos proporciona información adicional sobre las páginas.

2.12. Uso del documento funcional para integrar el mapa de navegación.

Un ejemplo del uso del mapa de navegación en un documento funcional podría ser el siguiente:

- **Secciones:** se debe intentar que sean las menos posibles, con el fin de concentrar las acciones del usuario en pocas áreas; hay que considerar que cada una de las áreas a integrar en el árbol del mapa de navegación requerirá de mantenimiento posterior en contenidos, gráfica y funcionalidad, lo que encarecerá el costo final de operación del sitio.
- **Niveles:** se debe intentar que el usuario esté siempre a menos de tres clicks del contenido que anda buscando. Por ello no se debería crear más de tres niveles de acceso; esto significa una Portada, una Portadilla de Sección y los Contenidos propiamente tales.
- **Contenidos relacionados:** se debe considerar que habrá funcionalidades que estén presentes en todo el sitio. Entre ellas se incluyen elementos como Buscador, Preguntas Frecuentes y Formularios de Contacto. Se

recomienda que este tipo de elementos quede fuera del «árbol» y «floten» sobre éste, con el fin de indicar que desde todas las páginas habrá enlaces a ellos.

2.13. Elementos utilizados para navegar.

Entre los elementos más relevantes que conforman el sistema de navegación se cuentan los siguientes:

- **Menú General:** siempre presente en todo el sitio, permite el acceso a cada una de las áreas del sitio.
- **Pié de Página:** usualmente ubicado en la parte inferior de cada página, indica el nombre de la institución, teléfonos, dirección física y de correo electrónico.
- **Barra Corporativa:** ofrece diversas opciones de información respecto del sitio y tal como el anterior, se muestra en todas las páginas.
- **Ruta de Acceso:** listado que aparece en la parte superior de cada página y que muestra el trazado de páginas que hay entre la Portada del sitio hasta la página actual que se esté revisando; cada una de ellas debe tener un enlace, para acceder al área de la cual depende la página. Cada uno de los elementos que conforman este «camino» debe tener un enlace que permita el acceso a esas áreas. En la literatura internacional en inglés sobre este tema, se llama a este elemento como «breadcrumbs».
- **Fecha de publicación:** para saber la vigencia de publicación del contenido desplegado.
- **Botón Home:** para ir a la portada.
- **Botón Mapa del sitio:** para ver el mapa del Sitio Web.
- **Botón Contacto:** para enviar un mensaje al encargado o diseñador del sitio.
- **Buscador:** presente en cada página si es que la funcionalidad existe en el sitio.
- **Botón Ayuda:** para recibir ayuda sobre qué hacer en cada pantalla del sitio.
- **Botón Imprimir:** para imprimir el contenido de la página; se espera que el formato de impresión del documento que se muestra en pantalla sea más simple que la página normal del Sitio Web, para dar la impresión al usuario de que hay una preocupación por ayudarlo en la tarea de llevar impreso el contenido.

2.14. Elaboración de una guía de usuario.

La elaboración de una guía de usuario para web pretende transmitir los conceptos y estructura de una nueva Web para que cualquier usuario pueda sacar el máximo partido de la misma. En realidad es un manual que debe estar especialmente dirigido a usuarios de la Web, sin embargo puede ser útil también para usuarios con tareas de mantenimiento o edición de portales institucionales como visión global de la web y sus procedimientos de publicación y promoción de noticias.

Esta guía de usuario debería comenzar explicando los objetivos del proyecto de diseño para que se pueda entender la solución adoptada.

Posteriormente, se pasará a desgranar toda la Web desde el punto de vista del usuario, explicando la estructura de navegación, de la o las páginas principales, y cada una de las partes u otras páginas como la presentación de la empresa, los canales de información, actualidad y eventos, la o las páginas de localización, la de contacto y la estructura del portal por cada "unidad organizativa".

Finalmente, se deberá incluir un glosario con la terminología utilizada en toda la guía o manual así como la indicación de donde dirigirse para consultar las **Preguntas Frecuentes** y para cualquier comentario, ofreciendo al usuario que no dude en ponerse en **contacto** a través de **direcciones de correo o números de teléfono**.

Algunos ejemplos de guía de usuario web:

http://www.unex.es/organizacion/servicios-universitarios/servicios/siue/archivos/ficheros/documentacion/Manual_Web_Usuario.pdf

<http://www.viauniverso.com/Files/GUIA%20DE%20USUARIO.pdf>

Guía de usuario incrustada en la propia web:

<http://www.hipocampo.org/uso.asp>