CSS

Introducción

Contenido

- Qué es CSS
- Cómo crear estilos inline y en documentos
- Representar colores, fuentes y fondos enHTML
- Seleccionar elementos con element, class, id

Objetivos de Aprendizaje

- Qué es CSS y como crear estilos inline y en documentos
- Representar colores, fuentes y fondos enHTML
- Seleccionar elementos con element, class, id
- Entender las propiedades: margin, padding y border properties
- Aplicar borders, paddings y margins
- Usar propieddes de border radius para esquinas redondeadas
- Eliminar espacio entre borde y padding
- Entender como estas propiedades afectan a otras propiedades
- Posicionar elementos en la pantalla
- Explicar la propiedad float y como usarla
- Eliminar floats y comprender por qué es importante
- Select elements using basic selectors
- Select elements combining different selectors
- Select elements based on their relationship to other elements
- Select elements based on their attributes
- Fully understand how Flexbox works.
- Implement flexbox in your projects
- Learn the different properties you can apply to flexbox containers and flexbox items
- Entender y trabajar con CSS Grid
- Create transitions using CSS3 Transitions module
- Explain the differences between transitions and animations
- Understand how to create animations using animation blocks
- Create keyframes
- Apply animation properties

¿Qué es CSS?

- Reciben el nombre de hojas de estilo en cascada o CSS (siglas en inglés de cascading style sheets) las indicaciones de estilo escritas es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML.

- Una indicación de estilo recibe el nombre de **norma** o **regla de estilo:**

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

- El "**destino**" (a quién se aplica una regla) se establece mediante un **selector**.

- El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores

    - http://www.w3.org/Style/CSS/current-work

Aplicar CSS

- Inline

```
<p style="color: red">text</p>
```

- Interno

```
<!DOCTYPE html>
<html>
<head>
<title>CSS Example</title>
<style>

    p {color: red;}

    a {color: blue;}

</style>
...
```

- **Se recomienda no usar estas opciones, ya que acoplan el HTML con el estilo gráfico**

Aplicar CSS Externo (preferido)

- Se lleva el css a un archivo externo.
  - Este se referencia desde el HTML

```
<!DOCTYPE html>
<html>
<head>
   <title>CSS Example</title>
   <link rel="stylesheet" href="style.css">
...
```

  - Fichero style.css

```
p {
   color: red;
}

a {
   color: blue;
}
```

Selectores, propiedades y valores

- Los selectores son nombres asignados a los estilos en las hojas de estilo. De momento etiquetas HTML,
  - por ejemplo *p, div, header....*
- Para cada selector hay "propiedades" entre llaves, que toman la forma de palabras como
  - *color, font-weight or background-color*.
- Se da un valor a cada propiedad después de dos puntos (NO un signo "igual"). Los punto y coma “;” se utilizan para separar las propiedades.

```
body {
    font-size: 14px;
    color: navy;
}
```

Tamaños y porcentajes

Hay unidades generales para los valores usados en CSS para expresar dimensiones como tamaño, ancho, alto.

- **px** (como en font-size: 12px) es la unidad de píxeles.
- **em** (como en font-size: 2em) es la unidad para el tamaño calculado en función del tamaño base de fuente. Así por ejemplo "2em“, indica dos veces el tamaño de fuente actual (por defecto 16 px).
- **pt** (como font-size: 12pt) es la unidad para puntos, para medidas típicamente en medios impresos.
- **%** (Como en width: 80%) es la unidad de porcentajes relativos al ancho o alto de la página o del elemento padre.

Colores

- CSS tiene 16.777.216 colores a tu disposición.
- Pueden tomar la forma de un nombre, un valor RGB (rojo / verde / azul) o un código hexadecimal.
- Por ejemplo, para indicar rojo, se puede usar:
    - red
    - rgb(255,0,0)
    - rgb(100%,0%,0%)
    - rgba(255,0,0,0)
    - #ff0000
    - #f00
- Los colores se pueden aplicar usando color y background-color.

```
h1 {
    color: #ffc;
    background-color: #009;
}
```

Texto

Puedes cambiar el tamaño y la forma del texto en una página web con un rango de propiedades:

- **font-family**: familia de fuentes como Times New Roman, Arial, or Verdana. → font-family: "Times New Roman"
- **font-size**: Tamaño de fuente
- **font-weight**: El grado de negrita de una fuente → font-weight: bold
    - Valores: bold,normal, bolder, lighter, 100, 200, 300, 400 (same as normal), 500, 600, 700 (same as bold), 800 or 900.
- **font-style**: itálica o no → font-style: italic; font-style: normal.
- **text-decoration**: subrayado debajo, encima o atravesado → text-decoration: underline
- **text-transform**: Cambia el tamaño (mayúsculas o minúsculas) del texto → text-transform: capitalize

Texto Ejemplo

- Todo junto se puede ver así:

font-weight: bold

font-style: italic

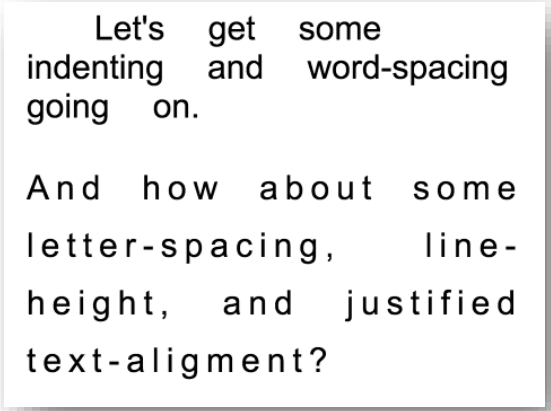
FONT-VARIANT: SMALL-CAPS

TEXT-TRANSFORM: UPPERCASE

```
body {
    font-family: arial, helvetica, sans-serif;
    font-size: 14px;
}

h1 {font-size: 2em;}

h2 {font-size: 1.5em;}

a {text-decoration: none;}

strong {
    font-style: italic;
    text-transform: uppercase;
}
```

Espaciado de texto

- letter-spacing: espaciado entre letras
- word-spacing : espaciado entre palabras
- line-height: Alto de línea
- text-align: Alineado a izquierda, derecha, centro
- text-indent: Indentado de la primera línea de un párrafo




```
p {
    letter-spacing: 0.5em;
    word-spacing: 2em;
    line-height: 1.5;
    text-align: center;
}
```

El modelo de caja

- El modelo de caja funciona así:
- en el medio tienes el área de contenido (digamos una imagen), rodeándolo tienes el padding, rodeándolo tienes el borde y el entorno tienes el margen.
- Se puede representar visualmente así:

- No necesitas usar todos los atributos, pero este modelo funciona para todos los elementos de la página.

Margen y padding

- Margen y padding son las propiedades más comúnmente usadas para espaciar los elementos de una página.
- Un margen es el espacio fuera de un elemento, mientras que el padding es el espacio dentro de algo.

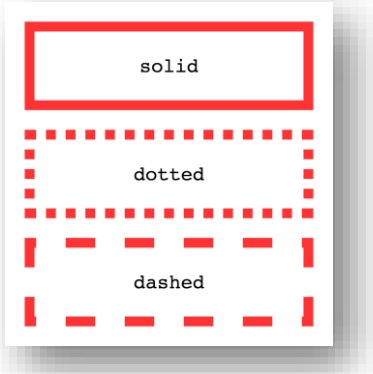
```
h2 {
    font-size: 1.5em;
    background-color: #ccc;
    margin: 20px;
    padding: 40px;
}
```

Bordes

Genera un borde alrededor de un elemento

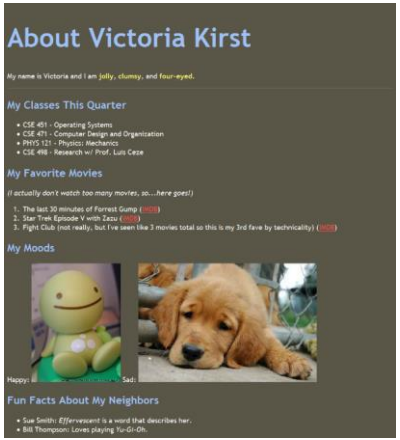
- border-style: Estilo de borde
  - solid, dotted, dashed, double, groove, ridge, inset and outset
- border-width: ancho de borde. Existen propiedades específicas para cada lado:
  - border-top-width, border-right-width, border-bottom-width and border-left-width
- border-color: el color de borde
- La version resumida sería
  - border: ancho estilo color → border: 2px dashed #ccc;

```
h2 {
    border-style: dashed;
    border-width: 3px;
    border-left-width: 10px;
    border-right-width: 10px;
    border-color: red;
}
```



Pongámoslo en práctica

Genera el HTML y CSS para la siguiente página
http://www.webstepbook.com/supplements/labsection/lab1-aboutme/images/aboutme.png



Selectores de clase y por identificador

- Se pueden definir selectores propios en forma de selectores de clase e ID.
- Tiene el beneficio de poder presentar el mismo elemento HTML de manera diferente dependiendo de su clase o ID.
- Usaremos un ID para identificar un elemento concreto, y una clase para identificar elementos con las mismas características.

```
<div id="top">

<h1>Chocolate curry</h1>

<p class="intro">This is my recipe for making curry purely with chocolate</p>

<p class="intro">Mmm mm mmmmm</p>

</div>
```

```
#top {
    background-color: #ccc;
    padding: 20px
}

.intro {
    color: red;
    font-weight: bold;
}
```

- Se puede definir selectores que apliquen a clases de un tipo de elemento concreto
- Por ejemplo: **p.intro o div.intro**

Agrupamiento

- Puede dar las mismas propiedades a una serie de selectores sin tener que repetirlos

```
h2, .thisOtherClass, .yetAnotherClass {
    color: red;
}
```

Anidamiento

- Si el CSS está bien estructurado, no debería ser necesario usar muchos selectores de clase o ID. Esto se debe a que puede especificar propiedades para selectores dentro de otros selectores.

```
#top {
    background-color: #ccc;
    padding: 1em
}

#top h1 {
    color: #ff0;
}

#top p {
    color: red;
    font-weight: bold;
}
```

Imágenes de fondo

- Son una manera potente de agregar una presentación detallada a una página.

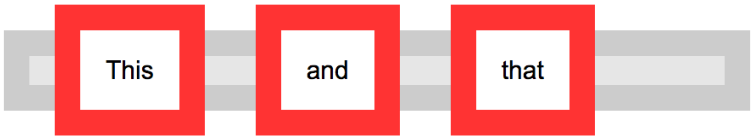
```
body {
    background: white url(http://www.htmldog.com/images/bg.gif) no-repeat top right;
}

div{
    url("bg.jpg") 20% 0% / contain repeat-y fixed content-box padding-box #fff
}
```

- El código anterior fusiona estas propiedades:
  - background-color
  - background-image, ubicación de la propia imagen.
  - background-repeat, cómo se repite la imagen:
    - repeat, efecto “tile” en todo el background,
    - repeat-y, repetir verticalmente,
    - repeat-x repetir horizontalmente
    - no-repeat no se repite.
  - background-position, que puede ser top, center, bottom, left, right, un tamaño, un porcentage, o combinaciones como top right.
  - background-size, el tamaño de la imagen: auto, tamaño, %, contain, cover, ancho alto

Display

Indica cómo se mostrará un bloque

- Inline: caja en linea → li { display: inline }



- block: caja independiente → #navigation a {display: block;}
- inline-block: Mantendrá un cuadro en línea pero prestará mayor flexibilidad de formato a los bloques; por ejemplo, permitiendo margen a la derecha e izquierda de la caja
- none: no muestra el bloque

Layout de página

Posicionamiento (position)

La propiedad de posición se utiliza para definir si un bloque es absoluto, relativo, estático o fijo:

- **static** es el valor predeterminado y representa un cuadro en el orden normal de las cosas, tal y como aparecen en el HTML.
- **relative** es muy similar a la estática, pero la caja puede ser desplazada desde su posición original con las propiedades superior, derecha, inferior e izquierda.
- **absolute** saca una caja del flujo normal del HTML y lo posiciona de manera absoluta. El bloque se puede colocar en cualquier parte de la **página** usando la parte superior, derecha, inferior e izquierda.
- **fixed** se comporta como absolute, pero posicionará absolutamente una caja en referencia a la ventana del navegador en lugar de la página web, por lo que los bloques fijos deben permanecer exactamente donde están en la pantalla, incluso cuando la página se desplaza.

Ejemplo Posicionamiento

```
<div id="navigation">
    <ul>
        <li><a href="this.html">This</a></li>
        <li><a href="that.html">That</a></li>
        <li><a href="theOther.html">The Other</a></li>
    </ul>
</div>

<div id="content">
    <h1>Ra ra banjo banjo</h1>
    <p>Welcome to the Ra ra banjo banjo page. Ra ra banjo banjo.
Ra ra banjo banjo. Ra ra banjo banjo.</p>
    <p>(Ra ra banjo banjo)</p>
</div>
```

```
#navigation {
    position: absolute;
    top: 0;
    left: 0;
    width: 200px;
}

#content {
    margin-left: 200px;
}
```

Flotamiento (float)

- Al flotar una caja, la desplazará a la derecha o a la izquierda de una línea, con el contenido circundante fluyendo alrededor de ella.

- Normalmente, se utiliza el flotante para desplazar partes más pequeñas dentro de una página, como empujar un enlace de navegación a la derecha de un contenedor, pero también puede utilizarse con partes más grandes, como columnas de navegación.

- **static** es el valor predeterminado y representa un cuadro en el orden normal de las cosas, tal y como aparecen en el HTML.

```
#navigation {
    float: left;
    width: 200px;
}
```

Fuentes personalizadas

- Para embeber una fuente haremos uso de la directiva @font-face

```
@font-face {
    font-family: “Nombre de fuente";
    src: url(nombre_fuente.woff);
}
```

- Luego usaremos la fuente con font-family

```
p {
                font-family: “Nombre de fuente", arial, sans-serif;
}
```

Bordes redondeados y sombras

- Bordes redondeados

```
.caja_rond {
    border-radius: 20px;
}
.caja_rond2{
    border-radius: 6px 12px 18px 24px;
}
```

- Sombras:
  - box-shadow: horizontal-offset vertical-offset blur-radius spread-distance color

```
div {
    box-shadow: 5px 5px 3px 1px #999
}
```

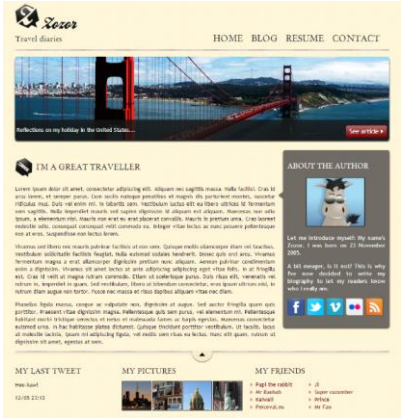
Herramientas de productividad

Revisa de estas herramientas de productividad

colorpicker.com
https://fonts.google.com/
http://css3buttongenerator.com/
https://css-tricks.com

Pongámoslo en práctica

Genera el HTML y CSS para la siguiente página
https://sdz-upload.s3.amazonaws.com/prod/upload/final_page.png



Pseudo elementos

Pseudo-Clases

- Son selectores que pueden actuar sobre información de elementos (podríamos decir que sobre su "estado")
- Esa información, se encuentra fuera del árbol HTML
  - Por ejemplo, los estados predefinidos de un elemento en función de su manejo por parte del usuario (cursor encima, enlace visitado, etc.)
- Ejemplo el elemento <a>
  - Estados normal (link), visitado (visited), cursor encima (hover) y pulsado (active)
  - Cada uno de esos estados podemos definirlo independientemente
  - Su sintaxis consiste en anexar el pseudo-elemento al nombre de la etiqueta mediante el símbolo de (:), como por ejemplo en:

```
a:visited{
    color:green;
}
```

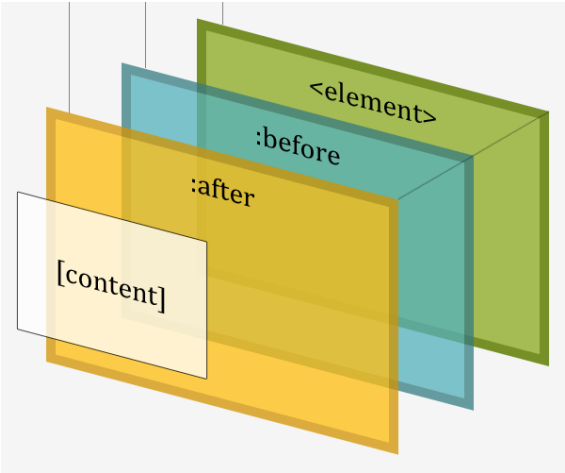
Pseudo-Clases

- Son selectores que pueden actuar sobre información de elementos (podríamos decir que sobre su "estado")
- Esa información, se encuentra fuera del árbol HTML
  - Por ejemplo, los estados predefinidos de un elemento en función de su manejo por parte del usuario (cursor encima, enlace visitado, etc.)
- Ejemplo el elemento <a>
  - Estados normal (link), visitado (visited), cursor encima (hover) y pulsado (active)
  - Cada uno de esos estados podemos definirlo independientemente
  - Su sintaxis consiste en anexar el pseudo-elemento al nombre de la etiqueta mediante el símbolo de (:), como por ejemplo en:

```
a:visited{
    color:green;
}
```

Modelo de caja con pseudo-elementos

- Los pseudo-elementos :before y :after se construyen como capas adicionales que pueden añadirse a un elemento cualquiera.
- Cada capa puede contener gráficos y texto, así como algunos símbolos especiales: circle, disc, square, etc.
- Múltiples posibilidades para completar la información de cualquier elemento.




```
.contenedor div:after {
   content:url(Graficos/HTML5_sintexto.png);
}
```

Pseudo-Clases y Pseudo-Elementos

Selector	Ejemplo	Descripción
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over

Pseudo-Clases y Pseudo-Elementos

Selector	Ejemplo	Descripción
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(language)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child

Pseudo-Clases y Pseudo-Elementos

Selector	Ejemplo	Descripción
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

Ejemplos

```
p:not(.MiClase) {
   color: red;
   font-style: italic;
}
```

```
<div>
   <p>Párrafo 1</p>
   <p>Párrafo 2</p>
   <p>Párrafo 3</p>
   <p>Párrafo 4</p>
</div>
```

```
p:nth-of-type(3){
    font-size:18px;
    color: Red;
}
```

Más sobre CSS

Barras de scroll

- La aparición o no de barras de scroll se puede controlar con la propiedad **overflow**
    - overflow: auto → deja al navegador decidir si muestra o no el scroll
    - overflow: scroll → fuerza mostrar el scroll
    - overflow: hidden → fuerza ocultar el scroll
- Se puede especificar para el scroll vertical u horizontal independientemente
    - Overflow-x: auto → deja al navegador decidir si muestra o no el scroll
    - Overflow-y: scroll → fuerza muestrar el scroll
    - Overflow-x: hidden → fuerza ocultar el scroll

CSS Tables

- Permiten establecer una disposición visual similar al de las tablas HTML, pero sin las etiquetas HTML correspondientes
- Están soportados por todos los navegadores.
- Su uso se basa en los valores table y table-cell de la propiedad display

```
.parent {
    display: table;
    /*width: 200px;*/
}

.parent div{
    display: table-cell;
    text-align: center;
    vertical-align:middle;
}
```

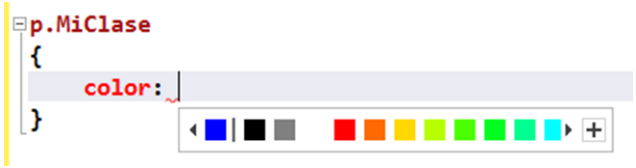
CSS Tables

- Equivalencias

table { display: table }
tr { display: table-row }
thead { display: table-header-group }
tbody { display: table-row-group }
tfoot { display: table-footer-group }
col { display: table-column }
colgroup { display: table-column-group }
td, th { display: table-cell }
caption { display: table-caption }

Más sobre los selectores (revisión)

- **Selectores dependientes**
  - Viene definido en función de otro selector ya existente
- Por ejemplo:



- Podemos usar varias clases simultáneamente en un elemento HTML indicándolo en el atributo class y separando los nombres de las clases con espacios
- El selector universal

  Se define por el signo *

```
* {
   margin: 3px;
   padding: 5px;
 }
```

Más sobre los selectores

- Se puede utilizar una sintaxis específica para indicar el tipo de relación jerárquica que se desea establecer, de acuerdo con lo que vemos en la tabla siguiente:

Formato	Selector	Condición
a b c	Descendiente	c descendiente de b descendiente de a
a * b	Universal	b dentro de a para cualquier ancestro de b
a > b	Hijo directo	b es hijo directo de a
a + b	Adyacente del mismo nivel	b es adyacente a a y de su mismo nivel
a ~ b	Mismo nivel	b es del mismo nivel que a

Más sobre los selectores

- Por ejemplo, con la siguiente sintaxis:

```
.MiClase > div > p { color:green }
```

- …estamos indicando que todos los elementos <p> que sean hijos directos de un elemento cuyo ancestro tenga el atributo class con un valor "MiClase", irán en color verde.

- Mediante estas combinaciones de operadores y selectores, siempre podemos encontrar una opción de selección por especial que sea.
  - Por posición (relativa o absoluta) en el DOM
  - Por identificación (los identificadores deben ser únicos por definición.)
  - Por clase
  - Por una combinación cualquier de esos factores

Más sobre los selectores (revisión)

- **Valores enumerados**
  - La agrupación de valores, indicando una lista enumerada, separada por comas, tiene sentido en ciertas propiedades para indicar alternativas

```
font-family: Arial, Tahoma, 'Segoe UI', 'Times New Roman';
```

- **Selectores por valor de atributos**
  - En CSS 2 se introdujo la noción de selectores por valor de atributos
  - Son formas de seleccionar elementos por los valores que poseen alguno de sus atributos
  - Lista de definiciones sintácticas

```
Elemento [attr] {}          /* Selector de atributo simple */
Elemento [attr='value']{}  /* Selector de comparación atributo/valor*/
Elemento [attr~='value']{} /* Selector de comparación parcial */
Elemento [attr|='value']{} /* Selector de atributo por lenguaje */
```

Más sobre los selectores (revisión)

- Imaginemos que tenemos las siguientes 4 etiquetas como parte de un menú:

```
<a href="Page1.html" rel="Value11"   lang="en-GB">Link 1</a>
<a href="Page2.html" rel="Value11 2" lang="en-US">Link 2</a>
<a href="Page3.html" rel="Value13"   lang="en-AU">Link 3</a>
<a href="Page4.html" rel="Value14"   lang="es-ES">Link 4</a>
```

- El siguiente código CSS selecciona los 4 elementos link y los pone todos en rojo:

```
a[rel] { color: red; }
```

- Los que, teniendo el atributo, coincidan con el valor:

```
a[rel='Value1'] { color: green; }
```

Más sobre los selectores (revisión)

- Elementos cuyo valor coincida parcialmente con la cadena indicada:

```
a[rel~='Value11'] { color: #b200ff; }
```

- Aquellos elementos que indiquen su atributo lang (lenguaje):

```
a[lang|='es'] { color: #cdb281; }
```

- Como vemos, el lenguaje se ha seleccionado de forma parcial, por lo que afectaría a todos los elementos cuyo lenguaje fuera una variante del español (es-*).

Media Queries

Mejoras para consultas relacionadas con multimedia

- Consisten en una serie de mejoras relacionadas con la regla (directiva) **@media**
- Mediante la regla @media y las palabras reservadas disponibles, podemos establecer indicaciones especiales para muchos dispositivos de salida
  - Podemos indicar un cierto estilo aplicable a las salidas por impresora, o la disposición visual en un teléfono o TV
  - En la versión anterior, la lista completa de media types incluía '**aural**', '**braille**', '**handheld**', '**print**', '**projection**', '**screen**', '**tty**' y '**tv**'
  - En la nueva "aural", se considera obsoleta, debido al nuevo soporte de atributos WAI-ARIA
  - Se añaden los tipos "**embossed**", y "**speech**"

Mejoras para consultas relacionadas con multimedia

- Por ejemplo, para indicar una regla para los dispositivos móviles, podemos utilizar los operadores **only** y **and**
- Una indicación de orientación para definir cómo queremos que se muestren los elementos de una lista

```
ul { overflow: hidden; }
li { float: left; }
@media only screen and (orientation: portrait#landscape))
{
    li { float: none; }
}
```

Mejoras para consultas relacionadas con multimedia

- También podemos usar otros tipos de definición condicional en el código:
- Usando la etiqueta <link> para enlazar los archivos CSS externos, se puede utilizar el atributo media para indicar los medios donde hay que aplicar los estilos de cada archivo:

```
<link rel="stylesheet" type="text/css" media="screen" href="basico.css" />
<link rel="stylesheet" type="text/css" media="print" href="especial.css" />
```

- Si la etiqueta "link" no indica el media usado, se usará contando que los estilos se deben aplicar a todos los medios, por lo que es equivalente a indicar media="all".

Mejoras para consultas relacionadas con multimedia

- También podemos
  - Indicar condiciones para el tamaño visual del dispositivo de salida
  - Exigir unos tamaños mínimos o máximos antes de aplicar una regla
- Los prototipos de código adoptarían una sintaxis genérica similar a los siguientes:

```
@media [media] and (device-height:value) { ... }
@media [media] and (max-device-height:value) { ... }
@media [media] and (min-device-height:value) { ... }
```

Mejoras para consultas relacionadas con multimedia

- Es posible establecer condiciones más específicas, como la relación de aspecto (aspect ratio) pero que se dan en conjuntos de dispositivos similares (móviles)

```
@media [medio] and (aspect-ratio: horizontal/vertical) {…}
@media [medio] and (device-aspect-ratio: horizontal/vertical) {…}
```

- Lo mismo cabe decir de otros aspectos visuales como el "pixel ratio" que podríamos manejar comprobándolo mediante indicaciones que tendrán un código genérico del tipo:

```
@media [medio] and (-moz-device-pixel-ratio: [número]) {…}
```

Breakpoints

- Los puntos o medidas de anchura donde se pueden crear saltos en el diseño Responsive, llamados comúnmente breakpoints, a partir de donde aplicar las media queries para Responsive Web Desing.
- Existen un conjunto de breakpoints para distintos dispositivos

```
@media (min-width:320px) { /* smartphones, iPhone, portrait 480x320 phones */ }
@media (min-width:481px) { /* portrait e-readers (Nook/Kindle), smaller tablets @ 600 or @ 640 wide. */ }
@media (min-width:641px) { /* portrait tablets, portrait iPad, landscape e-readers, landscape 800x480 or 854x480 phones */ }
@media (min-width:961px) { /* tablet, landscape iPad, lo-res laptops ands desktops */ }
@media (min-width:1025px) { /* big landscape tablets, laptops, and desktops */ }
@media (min-width:1281px) { /* hi-res laptops and desktops */ }
```

- https://css-tricks.com/snippets/css/media-queries-for-standard-devices/

Ejemplos

```
@media screen and (max-width: 1000px) {
    #content { width: 100% }
}

@media screen and (max-width: 800px) {
    #nav { float: none }
}

@media screen and (max-width: 600px) {
    #content aside {
        float: none;
        display: block;
    }
}
```

```
@media screen and (max-width:
1000px) {
    #content { width: 100% }
}

@media screen and (max-width: 800px)
{
    #nav { float: none }
}

@media screen and (max-width: 600px)
{
    #content aside {
        float: none;
        display: block;
    }

}
```