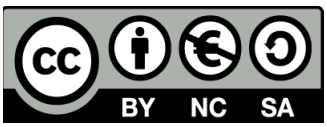
# Tema 4

## Integración editada de las partes en un documento HTML

### TALLER DE TRABAJO COLABORATIVO EN LA NUBE PARA E-LEARNING: HERRAMIENTAS

Universidad Internacional de Andalucía

**Proyecto OpenCourseWare-UNIA (ocw.unia.es)**




**Autor/@s**
**Paco Aguaza**

# ÍNDICE

**Introducción**

**Objetivos**

**Contenidos**

1. Lenguaje hipertextual y conceptos relacionados
   - 1.1. Multimedia y nuevas formas narrativas
2. Fundamentos de HTML5
3. Fundamentos de CSS3
   - 3.1. Dimensiones
   - 3.2. Posición
   - 3.3. Color
   - 3.4. Tipografías
4. Utilización combinada de HTML5 y CSS3
5. Las librerías abiertas para la automatización de los procesos comunicativos: Bootstrap

**Ideas clave**

**Referencias Bibliográficas**

## INTRODUCCIÓN

El e-learning, como forma de transmisión de conocimiento estructurado y secuenciado a través de Internet debería tener en cuenta los condicionantes que le plantea este canal de difusión. Como ya señaló Marshall McLuhan, "El medio es el mensaje"*(1)*. Una cita que plantea que varias cuestiones; entre ellas, que la forma en la que un contenido discursivo llega a su receptor, en cuanto a que su extensión y estructuración y en cuanto a la secuenciación de la información sea adecuada al medio a través del cual se transmite, puede llegar a cobrar más importancia que el propio mensaje que se quiere transmitir. Otro aspecto que plantea la reflexión de McLuhan es que cada medio de difusión tiene sus limitaciones y sus potencialidades y ante ellas la persona que produce el contenido discursivo tiene que adaptarse lo máximo posible.

Los propios medios de difusión y las herramientas de marketing se encuentran cada vez más entrelazados con la tecnología. Según **Gastón Roitberg,** secretario de redacción multimedia de *La Nación*, compilador del libro ***Periodismo Disruptivo, dilemas y estrategias para la innovación***, "Las redacciones (de medios de comunicación) van a ser cada vez más parecidas a una start up tecnológica"*(2)*. Una realidad que ya es patente en la transmisión de contenidos informativos y a la que el campo de la educación no puede permanecer ajena.

Lo digital va demostrando cada día diferentes posibilidades a la hora de producir y difundir cualquier forma de "mensaje", un concepto que en la era de la comunicación digital va mutando hacia el concepto de "contenido" -¿quizá porque mensaje y forma en que se presenta están cada vez más interrelacionados y tienen unas fronteras más difusas?-También lo demuestra en cuanto a capacidad de integrar, organizar y modificar elementos de distintos formatos textuales y audiovisuales para transmitir cualquier tipo de idea, información o conocimiento de forma adecuada a cada receptor. En el ámbito educativo, la forma de transmitir un "contenido" no podrá ser la misma si se realiza a través de una plataforma digital que si se realiza a través de un libro o una pizarra.

Esa alianza comunicación-informática es una consecuencia del desarrollo tecnológico de los dispositivos a través de los cuales se accede a la información y el conocimiento, pero también de los propios lenguajes con los que se brindan los contenidos a través de la web y que cada vez ofrecen más formas de automatizar los procesos y de llevar el concepto de discurso a cotas poco imaginadas años atrás. El receptor de información es, en este ámbito, "usuario", cada vez más acostumbrado a la capacidad de interactuar con ciertos elementos del contenido que se le presenta. La web es, ante todo, visual.

Dos lenguajes, ya universales, han facilitado la integración de información en la web de una forma relativamente sencilla. Nos referimos al lenguaje HTML (HyperText

Markup Language), como básico de la *www* y a otro lenguaje que es imprescindible para la edición visual de contenidos para la web, el CSS (Cascading Style Sheets), ambos interpretados por cualquier navegador. Su uso permite la creación y difusión global de contenidos aplicados a cualquier área del conocimiento dentro de ciertos estándares visuales y de, en un concepto más amplio, lo que en el ámbito del diseño web se denomina *usabilidad*(3) y que tiene que ver con facilitar a la persona usuaria la forma de interactuar con las diferentes opciones que se le presentan . Además, su esencia es totalmente colaborativa, fruto de años de inteligencia colectiva de desarrolladores y programadores y que, como cualquier otro idioma, tiene una institución que estandariza su norma y uso. Se trata de la W3C (World Wide Web Consortium). Detrás de cualquier contenido que hay en la web y que llega a nosotros en forma de diferentes áreas de información e interacción están involucrados, al menos, estos dos lenguajes.

Ya no es necesario ser programador informático para aprovechar en la medida necesaria sus potencialidades a la hora de desarrollar contenidos informativos, educativos o creativos. De hecho, y aunque vamos a utilizar otras herramientas que facilitan y automatizan esa labor, se puede desarrollar un documento multimedia que se puede difundir a través de la *www* y adaptado a cualquier dispositivo simplemente con un editor de texto plano(4) y la utilización de los códigos necesarios.

Gracias a esta potencialidad, la comunicación digital es totalmente personalizable: no solo podemos señalar en que punto de la secuencia de información se presentará un dato o elemento del contenido en concreto, por ejemplo, una imagen, sino que también podemos decidir que porcentaje de la anchura respecto a del total de la página ocupará o si se situará a la izquierda o a la derecha, además de poder utilizar los recursos ya automatizados de numerosos frameworks ( 5 ) o librerías. Por supuesto, la capacidad de elección en cuanto a colores, tipografías, iconos y prácticamente cualquier aspecto que afecte a lo visual, incluso animaciones, de un elemento puede ser infinita y los límites pueden estar más relacionados, en la mayoría de los casos, con la imaginación que con las posibilidades.

La maquetación(6), como necesidad visual heredada de la edición de textos, se encuentra con el reto reciente y cada vez más asumido de tener que adaptar la distribución y el tamaño de los elementos para que sean "consecuentes" con el tamaño de cualquier dispositivo a través del cual se acceda, ya sea PC, portátil, tablet o móvil. Una necesidad que el "Internet de las cosas"(7) ha encontrado como inexcusable. Por ello, a la hora de desarrollar cualquier tipo de contenido visual para la web, hay que tener en cuenta que el tamaño de los dispositivos con los que se accede a ella son cada vez más variopintos y se hace necesario ofrecer una solución a cada uno de ellos. El *responsive design*(8) como concepto y su automatización a través de recursos online como *Bootstrap*(9) será otro de los pilares de este bloque, en cuanto a la mencionada adaptabilidad del contenido a cualquier dispositivo.

El objetivo final de este curso es que el productor de contenidos online pueda optimizar la construcción del discurso multimedia que quiera transmitir, valiéndose de las diferentes herramientas y códigos libres disponibles e integrándolas y

combinándolas de la forma que resulte más adecuada para la idea que quiere hacer llegar a su público.

Para ello, en el desarrollo de esta unidad didáctica vamos a ahondar en cómo utilizar los lenguajes HTML5 y CSS3 como aliados para la integración de contenidos de diferente formato (texto, foto, audio, vídeo, infografías, ...) y para automatizar el proceso de hacer diferentes versiones según el soporte a través del que se accede.

También vamos a entender de forma básica como funciona la difusión de contenidos a través de Internet y diferentes alternativas para difundir los contenidos que hayamos desarrollado.

Si "el medio es el mensaje", hay que intentar adaptar ciertas características del mensaje al medio, no solo que se utiliza para transmitirlo sino también a aquel con el que la persona usuaria lo recibe.

## OBJETIVOS

El objetivo final de este taller es poder desarrollar un documento de tipo multimedia que puede ser después difunfido como un documento html autónomo, o integrado en cualquier CMS (Wordpress) o LCMS (Moodle), entre otros. Para ello, se han de conseguir otros objetivos parciales:

- Conocer ciertos fundamentos del HTML5, en especial los tipos de etiquetas que podemos utilizar dentro de un documento.
- Conocer ciertos fundamentos de CSS3, en especial el aplicable a la edición visual de documentos, en cuanto a dimensiones, distribución, colores y tipografías de los distintos elementos.
- Aprender a utilizar ambos lenguajes de forma complementaria.
- Aprender los fundamentos de la librería abierta Bootstrap para automatizar el proceso de edición visual.

# CONTENIDOS

## 1. Lenguaje hipertextual y algunos conceptos asociados

El lenguaje hipertextual, **¿qué es?**

**Forma de construir un relato (en el sentido más amplio) de una forma interactiva**. No se brinda toda la información de forma continua y lineal, sino que está secuenciada en distintas partes, normalmente jerarquizadas y relacionadas entre sí. El usuario puede elegir cuál de las partes quiere ver y puede acceder fácilmente a ella.

**Usabilidad.** Ser conscientes que dicho contenido se presenta en un espacio delimitado por las dimensiones de una pantalla y que se ha distribuir el acceso a las secuencias de información de forma clara, organizada, jerarquizada y estandarizada. "que el usuario no se pierda"

**Diseño**. Captar al usuario o a la audiencia no se consigue sólo con el contenido, sino recreando un entorno agradable, estéticamente aceptable (en esto las modas también influyen).

### 1.1. Multimedia y nuevas formas narrativas

Como dijo McLuhan, el medio es el mensaje. Por ello, tenemos que tener en cuenta cuál es la teleología de cada canal y sus potencialidades. Con el objetivo de que cada una de las piezas de audio, vídeo, texto o infografía que se hayan producido previamente, se integren en el mismo relato, o discurso narrativo, ya sea periodístico o no.

Al final y al cabo, el objetivo es aprovechar una tecnología que permite integrar cualquier tipo de formato audiovisual y textual y que permite cierta interacción por parte de los usuarios, que deciden qué segmentos de información quieren ver y cuáles no.

*"El término **multimedia** se utiliza para referirse a cualquier objeto o sistema que utiliza múltiples medios de expresión físicos o digitales para presentar o comunicar información. De allí la expresión multimedios. Los medios pueden ser variados, desde texto e imágenes, hasta animación, sonido, video, etc. "* (Wikipedia)

Necesitamos "traducir" al sistema lo que queremos hacer en su propio lenguaje. Hay decenas de lenguajes de programación. Vamos a centrarnos en dos de ellos, que

son los que se utilizan para la maquetación... o para la construcción de una web... ejemplo de la casa.

Por un lado tenemos el lenguaje **HTML5 HyperText Markup Language)**, la última versión del lenguaje HTML que permite desarrollar una **estructura** multimedia dentro de un documento para integrar **contenidos** de distinto formato (texto, imágenes, vídeo, audio, etc).

Por otro lado, tenemos el lenguaje **CSS3 Cascade Style Sheet (Hoja de estilos en cascada)**, la última versión del lenguaje CSS que nos va a permitir dotar de **forma** y **estilo** a dicha estructura.

**Estos dos lenguajes los podemos utilizar tanto para diseñar la plantilla de un sitio web como para la edición de contenidos de cualquier sitio basado en un CMS, así como para la creación de documentos online o multimedia.**

## 2. HTML5

El HTML5*(10)* es un lenguaje que se define como hipertextual. HTML quiere decir *Hypertext Markup Language* (lenguaje de marcas de hipertexto). Sus dos objetivos fundamentales son estructurar e integrar el contenido de un sitio web.

Su unidad fundamental es la etiqueta, que está conformada por una apertura y un cierre (excepto en el contenido incrustado, como imágenes o vídeos, que solo tienen etiqueta de apertura, ya que su contenido se define de otra forma). Según la etiqueta que se introduce se puede señalar si se trata de una parte de la estructura, una agrupación de elementos o un elemento individual de contenido, tal como texto, imagen, vídeo, etc. Una etiqueta puede contener un número indefinido de etiquetas, que a su vez pueden contener otras y así sucesivamente. El contenido de una etiqueta es todo lo que se encuentra entre su apertura y su cierre.

Una etiqueta puede tener relacionados múltiples atributos. Uno fundamental es el atributo src (fuente), que define la ruta o *url* donde el navegador puede "encontrar" una imagen, vídeo, audio y cualquier tipo de contenido incrustado que se incluya.

### 2.1. Sintaxis HTML5

HTML5 se puede definir como un lenguaje hombre-máquina. Un navegador (como Chrome, Firefox, IE), al no poseer imaginación ni capacidad de interpretación, necesitan que el mensaje enviado por el usuario que esté escrito al "pie de la letra" para poder comprenderlo. Es decir, que respete de forma absoluta todas las "normas lingüísticas".

Como cualquier lenguaje, el lenguaje de etiquetas HTML5 tiene una sintaxis propia:

***<Etiqueta atributo="valor">Contenido</etiqueta>***


**La mayoría de los fallos o resultados visuales inesperados tras escribir una línea de código están relacionados con errores en la sintaxis.**

## 2.2. Etiquetas HTML5

A nivel de la semántica HTML, una etiqueta representa la introducción de un elemento dentro de la estructura del documento. En este sentido, según el tipo de elemento que se vaya a introducir, se requerirá una u otra etiqueta.

**<style>** → Etiqueta que sirve para contener una "hoja de estilos" dentro del propio documento html. Todo lo que incluye se declara en lenguaje CSS. En este caso, vamos a utilizar para incluir el código CSS necesario para la edición visual del documento.

** ** → Representa un contenedor genérico sin ningún significado especial.

**<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>**, **<h6>** → Los elemento de cabecera implementan seis niveles de cabeceras de documentos; <h1> es la de mayor y <h6> es la de menor impotancia. Un elemento de cabecera describe brevemente el tema de la sección que introduce.

**<p>** → Define una parte que debe mostrarse como un párrafo.

**<ol>** → Define una lista ordenada de artículos.

**<ul>** → Define una lista de artículos sin orden.

**<li>** → Define un artículo de una lista ennumerada.

**<a>** → Representa un hiperenlace , enlazando a otro recurso.

**** → Representa un texto enfatizado , como un acento de intensidad.

**** → Representa un texto especialmente importante .

**** → Representa texto sin un significado específico. También representa un "elemento de línea" y, en el caso de bootstrao, su combinación con un atributo de clase puede representar iconos

**<br>** → Representa un salto de línea.

**<img>** → Representa una imagen.

**<iframe>** → Representa un contexto anidado de navegación, es decir, un documento html embebido o incrustado cuyo contenido estamos obteniendo de otro recurso web, como *Youtube* o *Genial.ly*

**<video>** → Representa un vídeo, con la interfaz necesaria para reproducirlo.

**<audio>** → Representa un sonido o stream de audio, con la interfaz necesaria para reproducirlo.

## TABLAS

**<table>** → Representa datos con más de una dimensión.

**<caption>** → Representa el título de una tabla.

**<tr>** → Representa una fila de celdas en una tabla.

**<td>** → Representa una celda de datos en una tabla.

**<th>** → Representa una celda encabezado en una tabla.

## FORMULARIOS

**<form>** → Representa un formulario, consistendo de controles que puede ser enviado a un servidor para procesamiento.

**<fieldset>** → Representa un conjunto de controles.

**<legend>** → Representa el título de un <fieldset>.

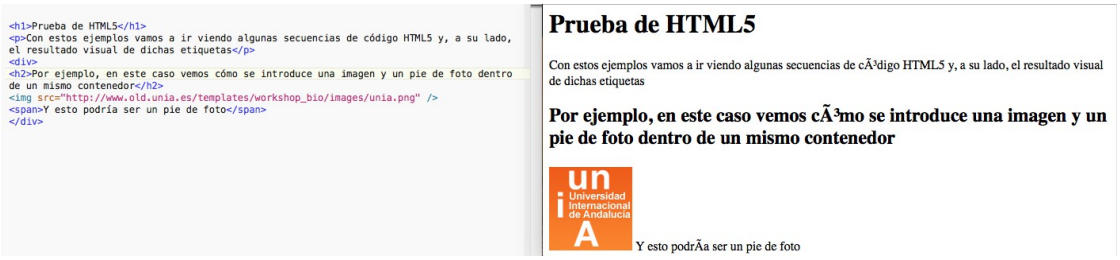
**<label>** → Representa el título de un control de formulario.

**<input>** → Representa un campo de datos escrito que permite al usuario o usuaria editar los datos.

**<button>** → Representa un botón.

**Existen más etiquetas, cuyo listado completo se puede encontrar en la web de MDN(10)**

ej ejemplo




**Ejemplo, a modo de doble ventana, sobre cómo interpreta un navegador diferentes secuencias de código. En este caso sólo son etiquetas, sin atributos ni propiedades de estilo. Se puede observar, por ejemplo, que los acentos no son bien reconocidos por el navegador; esto es porque no hemos indicado con etiquetas y atributos correspondientes, que es lenguaje español, ni el tipo de codificación que utilizamos (veremos más adelante cómo resolverlo).**

---

## 2.3. Atributos HTML5

Un atributo añade determinadas cualidades a un etiqueta, es decir, a un elemento que introducimos. Cada cualidad dependerá del tipo de atributo que le otorguemos y del valor que tiene dicho atributo. A continuación, algunos atributos HTML5 (los puntos suspensivos representan a la etiqueta a la que van añadidos, respetando la sintaxis html.

**<... lang="es-es">** → añadiendo este valor al atributo lang, expresamos que se trata lenguaje español de España. Añadiendo este atributo a la etiqueta html, se verá afectado todo el documento, tanto en su parte visual (<body>), como en sus metadatos (<head>). Puede incluirse en cualquier etiqueta

**<... charset="utf-8">** → El atributo *charset* expresa con qué juego de caracteres queremos que utilice el navegador. En este caso, el unicode 8, estándar en teclados "occidentales". En este caso concreto, si se adjunta a una etiqueta <meta> (de metadatos), dentro del <head>, afecta a todo el documento, aunque también puede aplicarse a etiquetas o elementos concretos.

**<... href="***url o archivo***">** → Acompaña a las etiquetas <a> o <link>, para indicar hacia dónde se dirige el enlace <a> o el archivo que debe cargar la etiqueta <link>

**<... src="***url o archivo***">** → Proviene del inglés 'source' (fuente) e indica el archivo que queremos cargar como contenido, por ejemplo, de una imagen o vídeo

**<... style="..." >** → Sirve para indicar al sistema que queremos dar estilo a un determinado elemento, aunque hay otras formas más recomendables de hacerlo, ésta puede utilizarse también y es la que el sistema prioriza si entra en contradicción con otras. Se denomina también "estilo en línea". Todo lo que va dentro de las comillas se declara en lenguaje css

**<... class="..." >** → Indica a un sistema que un elemento es de una clase determinada, cuyas características definimos en la hoja de estilos. Puede haber muchos elementos o etiquetas con la misma clase, al igual que un elemento puede tener varias clases. En lenguaje css se representa con un punto (.)

**<... id="...">** → Indica una clave que identifica a algún elemento. Similar a la 'class', la diferencia es que las id son únicas para cada elemento. Es decir, no puede haber dos o más elementos con la misma id. En lenguaje css se representa con una almohadilla (#)

**<... title="...">** → Señala el título del elemento. Por ejemplo, para señalar dónde nos dirige un enlace.

## 3. CSS3

CSS*(10)* es un acrónimo inglés que significa *Cascade Style Sheet* (**hoja de estilos en cascada**). Se trata de un lenguaje con ciertas características:

- Su **vocabulario** es muy similar al inglés.
- Su **estructura**: *hoja de estilos en cascada* quiere decir que dentro de una hoja de estilos se van señalando elementos más o menos específicos, y dicha especificidad tiene más importancia a la hora de ser traducida por un navegador si encuentra contradicciones con otra secuencia de código dentro de la misma hoja de estilo. Si le decimos al sistema que a todas las imágenes le queremos dar un tratamiento, pero señalamos que una imagen específica tiene otro tratamiento, esta última imagen se quedará con las cualidades que son más específicas para ella.
- Su **sintaxis**, como lenguaje hombre-máquina, es rígida.

**SINTAXIS CSS:**

***selector*** *(procede de una etiqueta o un atributo html)* ***{ propiedad:valor; propiedad:valor; }***

**Ejemplos de sintaxis CSS:**

```
h2{ font-family:arial; font-size:24px; }
p{ font-family:verdana; font-size:14px; font-style:italic; color:red;}
img{ width:40%; }
.autor{ color:red; }
#barra-derecha{background:rgb(0,60,40);}

@media screen and (max-width:768px) {
        img{ width:100%; }
}
```

**¿PARA QUÉ SIRVE EL LENGUAJE CSS3?**

Dar formato visual (estilo) al contenido. Señala las cualidades gráficas de cada elemento, desde la altura a la anchura, hasta el color, la tipografía, el fondo. También sirve para realizar transiciones y animaciones.

## UTILIZACIÓN DEL CSS3

En primer lugar, lo que hay que tener en cuenta es que **es un lenguaje íntimamente relacionado con el HTML5, ya que sirve para otorgar ciertas cualidades gráficas a los elementos que introducimos previamente en código html**.

Para elaborar una hoja de estilos, se puede hacer de **dos formas**:

- **En un documento independiente**, de formato .css, que mediante una etiqueta <link> se puede enlazar con el documento html que queramos. Resulta útil cuando se quiere desarrollar una hoja de estilos para el conjunto de un sitio web, ya que hay que almacenar cientos de líneas de código.
- **En el mismo documento html**, útil cuando se trata de la edición visual de documentos multimedia, ya que no es necesario escribir muchas líneas de código y el proceso se realiza de forma más ágil. Por tanto, **ésta es la opción que vamos a desarrollar para el proyecto**.

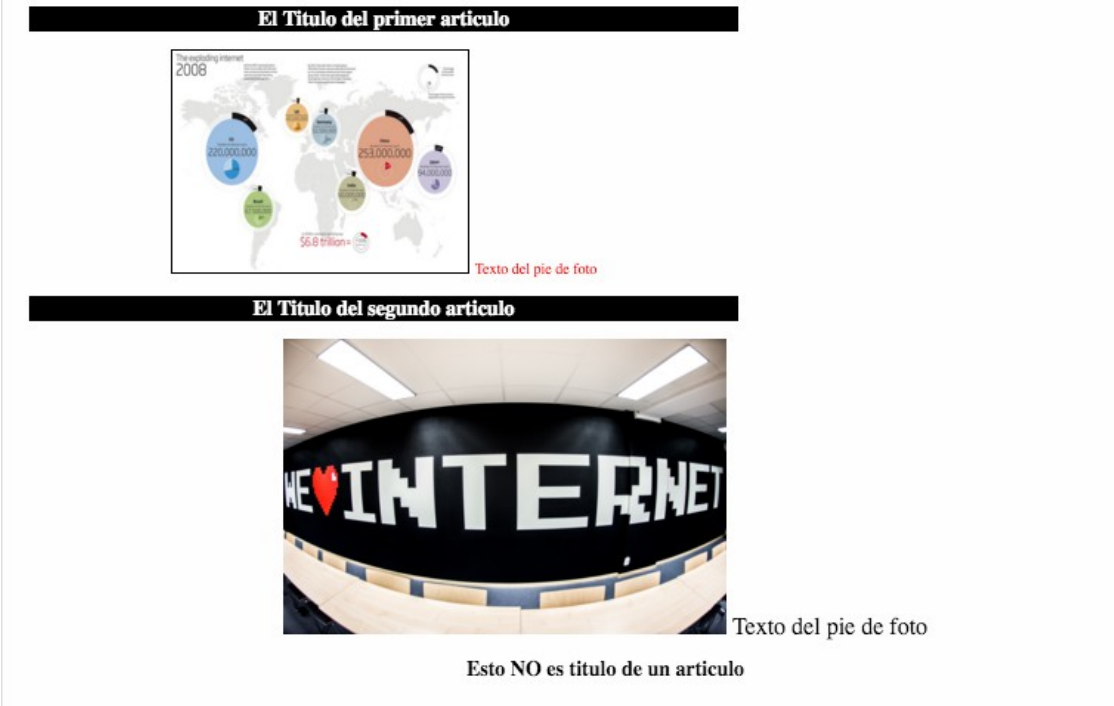
El lenguaje HTML5 incluye una etiqueta <style> para este último caso. Todo lo que se declara dentro de una etiqueta <style> (de apertura) y </style> (de cierre), se considera que está dentro de la hoja de estilos del documento y se escribe en lenguaje CSS.

## ej ejemplo

**El siguiente ejemplo, aunque breve, puede aportar bastantes explicaciones sobre lo referido anteriormente acerca de la aplicación de estilos a través de lenguaje CSS integrado en un documento html y respecto a otros conceptos que se señalarán a continuación. Para ello, se presenta una secuencia de código HTML son su hoja de estilo integrada en lenguaje CSS3 dentro de <style>...</style>. Posteriormente, se presenta el resultado de esta combinación de ambos lenguajes: el HTML para integrar contenido y el CSS3 para su edición visual:**

```
<div class="encabezado">
        <h2 class="titular">El Titulo del primer articulo</h2>
</div>
<div id="foto-primera">
        <img src="https://farm3.staticflickr.com/2739/4090557332_b804ce0c10_o_d.jpg" >
        <span>Texto del pie de foto</span>
</div>
<div class="encabezado">
        <h2 class="titular">El Titulo del segundo articulo</h2>
</div>
<div id="foto-segunda">
        <img src="https://farm4.staticflickr.com/3734/12301114235_3b9ce9bca4_o_d.jpg">
        <span>Texto del pie de foto</span>
</div>
<div>
        <h2>Esto NO es titulo de un articulo</h2>
</div>
<style>
        div{ width:800px; text-align:center; }
        img{ width:300px; }
        h2{ font-size:14px; }
        #foto-primera{ width:500px; }
        #foto-primera img{ width:200px; height:150px; border: 1px solid black;}
        #foto-primera span{ color:red; font-size:10px; }
        .encabezado{width:480px; margin-left:10px;}
        .titular{ background:black; color:white; }
</style>
```

**Resultado visual:**




Sin ser necesario tener en cuenta ahora mismo las propiedades (cualidades de los elementos) que veremos más adelante, y **centrando la atención en los selectores** (los elementos en sí, relacionados con sus propias etiquetas o atributos en html), se puede observar que lo que se escribe entre las etiquetas <*style*>*...*</*style*> tiene una sintaxis diferente a lo que está escrito antes. Y es que, en primer lugar se encuentran los elementos que incluimos en el documento, el contenido y sus fuentes, que se declaran en HTML. Lo que está posteriormente declarado entre las mencionadas etiquetas <*style*>*...*</*style*> está en lenguaje CSS y sirve para definir las cualidades de los elementos (etiquetas html) que hemos introducido previamente.

Dichos elementos HTML pueden ser representados en CSS de diferentes maneras:

- **Como etiqueta:** una etiqueta se representa en CSS de forma idéntica. Es decir, si queremos otorgar una cualidad a una etiqueta genérica (que puede ser única pero también puede repetirse un número indefinido de veces), simplemente hay que escribir su nombre en el lugar de selector.

En el ejemplo anterior, hay dos elementos que tienen etiqueta <*div*>. En la hoja de estilo se indican dos cualidades determinadas por sus respectivos valores , que son una anchura (*width*) del 80% y 200px de altura (*height*) para todos los <*div*>.

- **Como *id*:** Se representan en CSS con una almohadilla (**#**). La **id es un atributo de HTML** que puede estar incluido dentro de cualquier etiqueta y que le da carácter de **unicidad** a la misma. Es decir, que puede haber muchas fotos, pero con este atributo podemos "ponerle nombre a cada una de ellas" para darles cualidades específicas, teniendo en cuenta que ninguna id puede estar repetida, como una huella dactilar.

En el ejemplo anterior hay dos elementos con etiqueta <*div*>, y las cualidades que van incluidas dentro del selector div{..} les afectan a los dos. Sin embargo, uno de ellos, que va representado con su id, #foto-primera{...} presenta una contradicción. Como *div* debería tener una anchura (width) de 800px; sin embargo, como *#foto-primera*, debería tener una anchura de 500px. Esta contradicción la resuelve el navegador en favor del selector más específico en detrimento del más genérico; es decir, prevalece la *width* declarada dentro de *#foto-primera{...}*

- **Como *class*:** Se representan en CSS con un punto (.). La *class* es otro atributo de HTML pero, a diferencia del anterior, no es sinónimo de unicidad, sino que agrupa determinados elementos para otorgarles ciertas cualidades idénticas.

En el ejemplo anterior, hay dos títulos <h2> que tienen como atributo la misma class (*.titular*) con lo cual ambas adoptarán las cualidades de su etiqueta y las de su class y si entran en contradicción, primará lo que indique la class. El tercer título <h2>, por tanto, sólo tendrá las cualidades de su etiqueta, pero ninguna otra. El resultado será que todos los títulos tendrán un tamaño de letra (*font-size*) de 14px, pero solo los de la class *.encabezado* tendrán un color de letra (*color*) blanco y un fondo (*background*) negro

## 3. 1. CSS3: Dimensiones

La dimensión de un elemento significa el espacio que ocupa, expresado tanto en anchura como en altura, y puede resultar el aspecto más importante de la maquetación web, sobre todo en lo relativo al *responsive design* o diseño adaptativo. En definitiva, que se abandonan los preceptos de la maquetación basada en una altura y una anchura fijas y se han de concebir "diferentes tipos de maquetas", dependiendo de la anchura del dispositivo con el que se reciba dicho contenido. "Concebir" no quiere decir "elaborar" ya que diferentes herramientas, como secuencias de código y librerías, nos permiten automatizar este proceso. Sin embargo, hay que tener en consideración ciertas claves:

- El ***responsive design*** o diseño adaptativo parte de la "condicionalidad". Es decir, que **a partir de reglas condicionales, denominadas *media query***, se indica hasta que anchura máxima determinadas reglas tienen vigencia; si esa anchura máxima es superada, tienen vigencia otras reglas, hasta el próximo punto límite. Dichas reglas se refieren a un elemento y a una serie de cualidades que tendría, dependiendo del dispositivo en el que se reciba. Gracias al desarrollo de *frameworks* o librerías, como *Bootstrap*, este proceso se puede hacer de forma automática y será el objeto del próximo tema, así que no es necesario utilizar *media queries* en el proyecto, aunque puede resultar útil para personalizar determinados elementos.

- Por el tamaño de una serie de dispositivos y por la extensión de librerías como *Bootstrap* (que veremos y utilizaremos más adelante), **se han estandarizado una serie de "tamaños delimitantes",** donde se pasa de un *media query* a otro. Estas anchuras son 768 píxels, 992px y 1200px; es decir, lo que correspondería a dispositivos móviles (menos de 768px), tablets (768px-991px), PC portátil (992px-1199px), y PC de sobremesa (1200px y más). Esto no se trata de algo exacto; por ejemplo, hay portátiles con gran resolución que se encontrarían en el rango más ancho, pero puede servir para conceptualizarlo.

- En cuanto a **los elementos** que componen el documento (identificados con las etiquetas HTML y también con determinados atributos, como class e id), en *responsive design* hay que expresarlos, en su inmensa mayoría, con **dimensiones relativas a las dimensiones del elemento que lo contiene o del tamaño del dispositivo utilizado por el usuario**, tales como %, vh y vw, en lugar de con proporciones fijas (px).

Relacionado con lo anterior, y a modo de ejemplo, podríamos indicar dentro de los media queries que determinado elemento ocupe toda la anchura en un móvil y una tablet, pero que un portátil ocupe la mitad y en uno se sobremesa ocupe la cuarta parte. Entonces, en este caso, cuando el navegador detecte que la anchura del dispositivo es de 800px (tablet), dicho elemento ocupará 800px de anchura; si su anchura es de 850px, pues ocupará 850px; pero si la anchura del dispositivo es de 1000px (portátil), dicho elemento ocuparía 500px.

**FORMAS DE INDICAR LA ANCHURA:**

Se puede indicar de tres formas principales:

- **píxels (px)** → es la forma de declarar la anchura o altura de un elemento de forma fija.

Ej: img{ width:160px; height:90px; }

- **%** → es la forma de declarar la anchura o altura de un elemento en relación a la anchura del elemento que lo contiene. Si no lo contiene ningún elemento, tendrá la anchura del total del documento.

Ej:
```
<div class="contenedor-foto">
        <img src="..." >
</div>
<style>
        .contenedor-foto{ width: 50%; }
        .contenedor-foto img{ width:100%; }
</style>
```

En este ejemplo, las imágenes ocuparán el 100% de la anchura del contenedor (div) que los contiene y este div ocupará la mitad de la anchura del documento.

- **vw, vh** → representan el porcentaje de la anchura (vw) y la altura (vh) del dispositivo. Pueden resultar de utilidad para mantener las proporciones de elementos incrustados, como fotos y vídeos

**idea**

**En los inicios del responsive design se hizo un problema automatizar que cierto contenido incrustado, como vídeos e imágenes, mantuvieran su ratio en cuanto proporciones de anchura y altura. Por ejemplo, que si un vídeo era originalmente de formato 16:9, mantuviese esta ratio en cualquier dispositivo.**

**Sin embargo, se llegó a una solución ingeniosa dándole tanto a la anchura como a la altura un porcentaje de la anchura del dispositivo a ambos. Es decir, que la anchura del vídeo o de la foto fuese un determinado porcentaje de la anchura del dispositivo y que la altura de dicho vídeo o foto fuese un porcentaje también de la anchura.**

**Si, por ejemplo, una imagen originalmente tenía una proporción de 2000px de ancho por 1500px de alto (4:3), podríamos señalar que la anchura de la imagen fuese del total del dispositivo (100vw) y que la altura fuese de las tres cuartas partes de la anchura del dispositivo (75vw), así mantendrá siempre su ratio de 4:3, independientemente de la anchura del dispositivo.**

**Si, por ejemplo, queremos incrustar un vídeo de YouTube, primero debemos eliminar la anchura (width=" ") y altura (height=" ") que vienen por defecto como atributos -y que representa un formato de 16:9- y después hacer que ese ratio no venga determinado de forma fija, sino relativa. Imaginemos que queremos que el vídeo ocupe el 80 por ciento de la pantalla del dispositivo, entonces debemos indicar que su anchura es de 80vw. ¿Cuál debería ser la altura para mantener el ratio? Si tenemos en cuenta que 16/9=1.7778 sólo tenemos que hacer la operación: 80vw/1.7778=45vw será la altura del vídeo de Youtube para mantener el ratio de 16:9 respecto a una anchura de 80vw. Para cualquier vídeo o foto que venga en formato 16:9, basta con dividir su anchura por 1.7778 para averiguar cuál debería ser su equivalencia en altura para mantener ese ratio.**

**Veamos cuál sería el código necesario para desarrollar estos dos casos, y haciéndolos que sólo sean operativos para dispositivos móviles (pantallas con 768px de anchura o menos)**

```
<img src="https://farm4.staticflickr.com/3373/3485479724_95a17f39ab_o_d.png" >
<iframe src="https://www.youtube.com/embed/29gzSnwWsdE" frameborder="0"
allowfullscreen></iframe>

<style>
@media all and (max-width:768px){
        img{ width:100vw; height:75vw; }
        iframe{ width: 80vw; height: 45vw; }
}
</style>
```

## 3. 2. CSS3: Posición

**La situación de los elementos dentro del espacio que comprende el documento completo o dentro de otros elementos en los que puedan estar incluidos depende de una serie de propiedades como se observa a continuación:**

**width, height** → anchura y altura. Se pueden declarar con valores absolutos (px) o relativos (%). En el caso de ser relativos, lo será respecto a su contenedor inmediato.

**top, bottom, left, right** → indica las coordenadas en las que se posicionará un elemento. También se puede expresar en valores absolutos y relativos. Del resultado, como veremos más adelante, dependerá mucho si la *position* del elemento es *absolute*, *relative* o *fixed*.

**margin (o margin-top, margin-bottom, margin-left, margin-right)** → indica los márgenes exteriores que tendrá un elemento respecto a otro contiguo. Determina una una zona, que NO es del contenido del elemento donde no podrá "colocarse" ningún otro elemento.

**padding (o padding-top, padding-bottom, padding-left, padding-right)** → indica los márgenes interiores de un elemento, cuyo tamaño hay que sumarlo al del propio elemento que los contiene para determinar su dimensión total tendrá un elemento respecto a otro contiguo.

**display** → En este caso, considerando que cada elemento en una web es un "rectángulo", señala cuál será su comportamiento respecto hacia los otros "rectángulos".

- ***display:inline*** → mantendrá el flujo de la linea y no provocará un salto hacia otro punto más bajo, además, aunque aumenten sus dimensiones (por ejemplo con padding), el resto de elementos no se verán afectados
- ***display:block*** → no se mantendrá al lado de otro elemento, si no que se situara inmediatamente debajo del anterior, lo más a la izquierda posible.
- ***display:inline-block*** → similar al 'inline', pero un cambio en sus dimensiones sí afecta al resto
- ***display:none*** → hace que el elemento no se muestre, simplemente desaparece.

**float** → Se trata si del elemento "flota" o no... quiere decir que un elemento le "permite" a otro situarse a su lado, si hay espacio suficiente para los dos. Resulta útil, ya que por defecto, los elementos con *display:block;* se sitúan uno debajo de otro. Para ello, debe de flotar tanto el propio elemento, como su contiguo. Si flota, puede hacerlo a la izquierda (*float:left;*) o a la derecha (*float:right;*), situándose lo más a la izquierda o derecha que su margen lo permita.

**position** → Éste es un atributo fundametal para la posición de los elementos dentro del documento y está muy relacionado con lo declarado en width, (coordenadas), margin el float.

- Si un elemento tiene ***position:relative;*** se situará en relación a su elemento inmediatamente superior y con el mismo nivel jerárquico; es decir, acorde al "flujo". Si, además, flota, y tiene espacio, se colocará justo al lado.
- Si un elemento tiene ***position:absolute;*** se situará en una posición marcada dentro de su elemento contenedor y determinada por sus coordenadas. Es parcialmente independiente al "flujo" de la página.
- Si un elemento tiene ***position:fixed;*** permanecerá siempre visible en un punto de la pantalla del navegador determinado por sus coordenadas. Es decir, que aunque hagamos scroll top y scroll bottom, el elemento permanecerá visible en pantalla y en la misma posición, con un "flujo" totalmente independiente de la página

## 3. 3. CSS3: Color

Antes de conocer las diferentes propiedades de CSS3 en las que puede intervenir el color, lo primero que hay que tener en cuenta es que se trata de un lenguaje orientado al diseño web, por lo tanto, a pantallas. Esto quiere decir que se trata de combinaciones de colores bajo el paradigma de la **luz**, no del pigmento. Dicho de otra forma, **la combinación de sus tres colores primarios nos dan como resultado el blanco**. Los 3 colores primarios para la luz son rojo, verde y azul, en inglés *Red Green Blue*, por lo que se conoce como color **RGB**.

**Dentro del color RGB tenemos diferentes formas de utilizarlo en CSS: Textual, RGB decimal y RGB hexadecimal.**

### TEXTUAL

Como su nombre indica, simplemente declararemos el color con un palabra correspondiente al color en idioma inglés, como "red" o como "light-blue". Hay casi 150 formas textuales reconocidas por el consorcio W3C para el lenguaje CSS3 y se pueden consultar en w3schools.

Ej: `a{ color:blue;}`

### COLOR RGB DECIMAL

Puede resultar la más fácil de usar ya que nuestra cultura numérica está desarrollada, en su mayor parte, bajo un sistema decimal. Cada color va de 0 a 255, en el que el 0 representa la total ausencia de un color y el 255, su total presencia. Se puede representar de esta forma absoluta o también relativa, en porcentajes de 0 a 100.

Para declarar un color de esta forma, lo haremos de la siguiente manera:

***rgb( cantidad de rojo, cantidad de verde, cantidad de azul );*** → cada cantidad en cifras de 0 a 255

***rgb( cantidad de rojo, cantidad de verde, cantidad de azul );*** → cada cantidad en porcentajes de 0 a 100

Ej: div{ background:rgb( 200, 100, 50 ); }
div{ background: rgb( 80%, 40%, 20% ); }

Además, **esta forma permite añadirle también el conocido como "canal alfa"**, que indica la opacidad. La opacidad se determina de 0 a 1 y señala un "porcentaje".

***rgba(cantidad de rojo, cantidad de verde, cantidad de azul, grado de opacidad)***

Ej: rgba(100, 50, 20, 0.6);

Este último, indicaría que sería la combinación de los tres primeros colores con una opacidad del 60%

**COLOR RGB HEXADECIMAL**

Representaría también las cantidades de color de rojo, verde y azul, al igual que el anterior, pero expresadas en forma hexadecimal. Eso quiere decir que van del 0 al 15. Para expresar los números del 0 al 9, se utilizarían estas mismas cifras. Pero del 10 al 15 se realizaría de la siguiente manera:
**10=A | 11=B | 12=C | 13=D | 14=E | 15=F**

Si queremos convertir un número a lenguaje hexadecimal, lo haríamos dividiéndolo por 16 para obtener las "dieciseisenas" y lo que sobre y no llegue a completar una "dieciseisena", sería una unidad. Por ejemplo:

Decimal: 50 → 16*3=48 +2 → Hexadecimal: 32
Decimal: 60 → 16*3=48 +12 → Hexadecimal: 3C

En lenguaje CSS se representan los 3 colores precedidos de una almohadilla y si una "dieciseisena" es igual a su unidad, puede escribirse sólo una
#rojoverdeazul

Ej: header{ background: #2C443A;}

Aunque es más difícil de conceptualizar el color así y no tiene "canal alfa", lo cierto es que está en base binaria (la propia de la informática) y muchos sistemas "traducen" lo que declaramos en decimal a hexadecimal, por lo que nos lo podemos encontrar frecuentemente declarado de esta forma.

## PROPIEDADES DE CSS EN LAS QUE SE PUEDE UTILIZAR COLOR

### color

Se refiere al color de la tipografía. Con lo cual se puede utilizar en párrafos (p), Cabeceras (h1-h6), enlaces (a) y elementos de línea (span), además para definir los  y 

Ej:
```
p{ color:black; }
span{ color:rgb (255, 50, 50); }
strong{ color: #009; }
```

### background

Se refiere al color del fondo del elemento. Se puede utilizar en prácticamente cualquier elemento, tanto de línea como de bloque y puede tener muchas propiedades "asociadas".

### background-color

Señala cuál es el color de fondo. Se declara de forma similar al color

### background-image

Indica una imagen de fondo, a través de su url

Ej:
```
div { background: url(images/foto.png); }
div { background: url(http://www.miweb.com/images/foto.png); }
```

### background-position

Indica la "posición" del fondo, a través de porcentajes de sus ejes X e Y.

Ej:
```
div#uno{ background-image: url(img/playa.jpg); background-position: 0% 50%; }
```

Así conseguiríamos que la imagen de fondo se situase en el borde izquierdo del div, pero que comenzase a la mitad vertical del div.

**background-repeat**

Sirve para indica si la imagen de fondo se repite. Por defecto es no-repeat, es decir, por defecto los fondos no se repiten. Las otras opciones son repeat-x (se repite siguiendo el eje horizontal, repeat-y (siguiendo el eje vertical) y repeat (se repite por todos lados).

**background-origin**

Señala 3 formas diferentes de que el fondo ocupe un espacio:

*background-origin: border-box;* → el fondo también ocupa el espacio de los bordes.
*background-origin: padding-box;* (por defecto)→ el fondo ocupa todo el espacio del contenedor, pero sin los bordes.
*background-origin: content-box;* → el fondo tiene en cuenta el padding existente para ocupar el fondo, y no ocupa la zona del padding.

**background-size**

Señala el tamaño de la imagen de fondo. Especialmente útil si queremos ajustar el tamaño de la imagen del fondo a la del contenedor. Se puede declarar en tamaño fijo (pixels) y porcentaje. Primero la anchura y después la altura

Ej. div{ background-size: 30px 100px;}
div{background-size: 100% 100%; } → así ocuparía todo el fondo y sin pasarse

**Si queremos declarar varias propiedades relacionadas dentro de la misma propiedad background, lo haremos en el siguiente orden:**

***background: color image position/size repeat origin***
***Ej: background: red 0% 10% no-repeat padding-box;***

**border**

Sirve para dotar de un borde al contenedor o elemento sobre el que declaramos la propiedades. Se hace de la siguiente forma:

***border: anchura tipo color;***

Los tipos de bordes pueden ser:
*none* → ninguno

*dotted* → de puntos
*dashed* → línea discontinua
*solid* → línea continua
*double* → línea doble
*groove* → tridimensional "con la luz procedente de abajo"
*ridge* → tridimensional "con la luz procedente de arriba"

Si utilizamos la propiedad border, el borde se aplica a los 4 lados del rectángulo. Si queremos que el borde se aplique a sólo un lado, lo haremos con estas propiedades, según la posición del borde:
*border-top, border-bottom, border-left, border-right.*

Ej: img{ border:2px solid grey; border-bottom:4px dashed rgb(30,30,100); }

De esta forma, todo los bordes de la imagen serán iguales menos el de abajo, que tendrá sus propias características.

**box-shadow**

Sirve para darle sombre y, con ella, un cierto efecto de profundidad y tridimensionalidad al elemento. Su nomenclatura es la siguiente:

***box-shadow: desplazamiento-horizontal(px) desplazamiento-vertical(px) extensión del difuminado(px) color***

Ej: box-shadow: 2px -2px 4px rgba(0,0,0,0.6)***;***

## 3. 4. CSS3. Texto

Las tipografías y lo que se puede hacer con ellas tienen en css dos propiedades principales a las que se les asocian otras muchas: ***font*** y ***text***.

***FONT* Y PROPIEDADES ASOCIADAS**

**font** determina una serie de propiedades que podemos utilizar para actuar sobre las letras a nivel tipográfico. Por lo tanto, los elementos con los que podemos utilizarlas serán: ***p, a, h1-h6*** y ***span***

**font-family**

Es el nombre de la familia tipográfica que vamos a utilizar. Puede estar en el sistema o puede ser una de las que leen los navegadores por defecto. Si queremos en otra variante, negrita o cursiva, debemos asegurarnos que la familia que cargamos incluya esas variedades o incuirlas también

Las fuentes seguras son courier, monaco, tahoma, palatino, arial, impact, verdana, georgia y times new roman (otras, dependen del sistema operativo)

Ej. `a{ font-family:verdana, georgia, sans-serif;}`

**font-size**

Tamaño de la tipografía. En *px* (píxels) o en *em.*

Ej. `a{ font-size:14px;}`

**font-weight**

"Peso" de la tipografía. Se puede expresar de *100* a *900*. Tiene sus equivalencias: *400* es *normal* y *700* es *bold*; es decir, normal o negrita.

Ej. `span{ font-weight: bold; }`

**font-style**

En este caso, se trata del "estilo" de la tipografía. Su uso está determinado para definir si la letra es *normal*, *italic* (cursiva) u *oblique* (similar a la cursiva, en muchas familias tipográficas no se distingue).

**font-stretch**

Determina lo "apretadas" o separadas que están las letras dentro de la misma palabra. A día de hoy, las versiones actuales de la mayoría de navegadores no la interpretan, pero sigue siendo una propiedad css, así que es posible que sea interpretada por versiones de los navegadores. Sus posibilidades son:

*ultra-condensed, extra-condensed, condensed, semi-condensed, normal, semi-expanded, expanded, extra-expanded, ultra-expanded* (de más "apretadas" a más separadas)

**font-variant**

Simplemente sirve para que las letras sean todas mayúsculas y que la primera letra de cada palabra sea "más mayúscula" o más grande que el resto de mayúsculas. En Este caso la propiedad se declararía así:

```
h2{ font-variant:small-caps; }
```

## *TEXT* Y PROPIEDADES ASOCIADAS

**text**, al igual que font, actúa sobre el texto, pero más a nivel de conjunto que de elementos tipográficos.

**text-align**

Determina la alineación del párrafo o línea. Al igual que en un editor de texto, las opciones pueden ser izquierda, derecha, centrado (cada línea se sitúa en el centro, teniendo por tanto CADA LÍNEA un mismo margen izquierdo y derecho) y justificado (todas las líneas, menos la última, llegan hasta el final). En CSS se escriben así:

*left, right, center, justify*

**text-align-last**

Se declara igual que *text-align*, pero para sólo afecta a la última línea del párrafo.

**text-decoration**

Sirve para poner una líneal texto. Puede ser *underline* (subrayado), *overline* (suprarrayado) y l*ine-through* (la línea lo atraviesa)

**text-decoration-color**

Color de la línea del *text-decoration*. Se declara como cualquier forma de color

**text-decoration-style**

Por defecto, la línea de text-decoration es *solid* (continua), pero hay otras formas: *dotted* (de puntos), *dashed* (línea discontinua), *double* (doble), *wavy* (ondulada)

**text-indent**

Sangrado de la primera línea. Se determina en *nºpx*

```
p{ text-indent:50px; }
```

**text-overflow**

Cuando el texto se encuentra dentro de un div, puede ocurrir que ocupe más de la altura del div y sobresalga sel espacio que debería ocupar. Entonces primero tendríamos que decir que

div{ overflow-y:hidden; } (para que no aparezca) o bien

div{ overflow-y:scroll; } (para que aparezca si hacemos scroll)

Pero también existe la opción de hacerlo con la propiedad *text-overflow*. Entonces tenemos dos opciones:

p{ text-overflow:clip; } → Se adapta al tamaño del div

p{ text-overflow: elipsis} → Se adapta al tamaño del div y deja unos puntos suspensivos para indicar que el texto sigue.

**text-transform**

Como su nombre indica, sirve para "transformar" el texto, al igual que haría un editor. Las posibilidades son *uppercase* (todo mayúsculas), *lowercase* (todo minúsculas) o *capitalize* (mayúscula, la primera de cada palabra).

**line-height**

Determina la altura de la línea, teniendo en cuenta la que ya ocupa el propio texto. Es decir, que la línea ya de por sí ocupa 1 ó 100%. En este caso la siguiente línea comenzaría justo donde acaba la línea de texto anterior. Para conseguir el interlineado debemos declarar una altura mayor a 1 ó 100%. Un interlineado estándar estaría en el 140% ( la línea ocupa su texto y deja debajo un 40% más)

```
p{ line-height:140%;}
```

## 4. Bootstrap

Bootstrap(9) es un *framework*, un conjunto de herramientas y librerías para el desarrollo de sitios web. Aunque tiene muchas aplicaciones, lo vamos a utilizar principalmente para automatizar el proceso de dimensionar y distribuir los elementos dentro del conjunto espacial del documento para cada tipo del dispositivo. Es decir, nos va a facilitar la tarea de hacer nuestro documento *responsive* (adaptativo), ya que ayuda a situar los elementos, en cuanto a espacio y lugar que ocupan dentro de un documento en cada dispositivo.

### ¿CÓMO FUNCIONA?

Primero hay que tener en cuenta que **Bootstrap tiene múltiples utilidades**. Nos vamos a circunscribir, para el objetivo que tenemos planteado, solamente a las relativas a CSS. Y, en especial, a las relativas a su *grid system* (sistema de rejilla).

El ***grid system*** de Bootstrap se basa en que divide la anchura de la pantalla o de un <*div*> en 12 columnas, independientemente del dispositivo que se utilice. En cuanto a la distribución vertical, se pueden añadir el número de filas que se desee y dentro de cada fila, se pueden incluir tantos elementos mientras que la suma de las columnas que ocupen dichos elementos no supere 12, ya que si se supera, dichos elementos comenzarían a formar una fila debajo.

La potencialidad del *grid system* de Bootstrap es que se puede asignar a cada elemento un número de columnas, que ocupará dependiendo del dispositivo con el que accede un usuario y que se puede decidir qué elementos van al lado de otros y cuáles encima o debajo.

Una distribución que se "condiciona" a los diferentes tipos de dispositivo. En el apartado 3.1 *(CSS: Dimensiones),* se señalan las tres anchuras (768px, 992px y 1200px) que delimitan cuatro opciones condicionales (menos de 768px, 768px-991px, 992px-1199px y 1200px-∞) que, de forma general, corresponderían a dispositivo móvil, tablet, PC portátil, y PC de sobremesa, respectivamente.

| | **Extra small devices** Phones (<768px) | **Small devices** Tablets (≥768px) | **Medium devices** Desktops (≥992px) | **Large devices** Desktops (≥1200px) |
|---|---|---|---|---|
| **Grid behavior** | Horizontal at all times | Collapsed to start, horizontal above breakpoints | | |
| **Container width** | None (auto) | 750px | 970px | 1170px |
| **Class prefix** | `.col-xs-` | `.col-sm-` | `.col-md-` | `.col-lg-` |

## UTILIZACIÓN

Antes de nada, hay que integrar Bootstrap en nuestro sistema. Para ello, debemos introducir una línea de código dentro de la etiqueta <head>...</head>. La línea de código que hay que introducir es la siguiente:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7" crossorigin="anonymous">
```

(en realidad, es una sola línea)

En primer lugar, para que el contenido del documento ocupe la totalidad de la anchura (los márgenes se crean automáticamente), se crea un ... . Entre la etiqueta de apertura y cierre se escribirá todo el código del documento (no es necesario hacerlo en una misma línea).

En segundo lugar, se van creando filas (row). Por defecto, un *row* ocupa las 12 columnas. Esto se consigue con cada div de clase row ( div class="row" ). Dentro de cada row (fila), se van incluyendo los elementos que ocupen la misma fila y se les añaden los distintos condicionales, en cuanto a cuantas columnas ocuparán según cada dispositivo. Por ejemplo, si integramos una imagen en un documento, para un PC de sobremesa puede resultar suficiente con que ocupe sólo el 25% de la anchura de la pantalla del dispositivo (3 columnas); sin embargo, en un móvil, lo conveniente sería que ocupase toda la anchura de la pantalla (12 columnas).

Como tercer paso, se debe crear un div para cada elemento o grupo de elementos con sentido unitario (como una foto y un pie de foto). A cada div se le asigna un atributo *class="..."*. Posteriormente, como contenido del atributo *class*, se añadirán diferentes códigos según cuántas columnas (de las 12 de anchura totales) ocupen en cada uno de los dispositivos. Lo mismo se puede hacer con cada elemento que se incluya en cada div, realizando el mismo proceso.

Dichos códigos están compuestos de tres partes, separadas por guiones:

- Un **prefijo** col, común a todos, como indicativo.
- Un **sufijo** que indica de que tipo de pantalla:
  I. Extra-small (móvil) → xs
  II. Small (tablet) → sm

III. Medium (PC portátil) → md

IV. Large (PC sobremesa) → lg

- Una **cifra**, que indica cuantas columnas ocupa en dicho caso

Por último, hay que tener en cuenta que no es necesario indicar los 4 tamaños para cada elemento. Si no se indica para un rango de anchura, para dicho rango ocupará las columnas del inmediatamente inferior que esté indicado. (si solo está indicado para *xs* y *md*, *sm* ocupará las mismas columnas que *xs* y *lg* ocupará las mismas que *md*.

**importante**

**Para un elemento que esté incluido en un "contenedor" (div), dicho elemento toma como referencia a su contenedor, a la hora de contabilizar 12 columnas. Es decir, si tenemos un div padre que ocupa 6 columnas (50% del** div class="row" **) y dentro de él, algún elemento que ocupa 6 columnas también (50% del div que lo contiene), dicho elemento ocupará realmente 3 columnas del total (la mitad de la mitad, el 25%).**

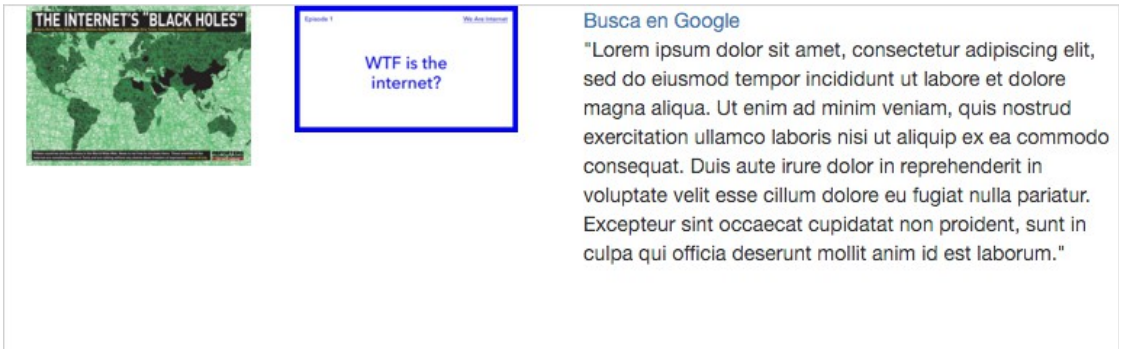
## ej ejemplo

**En el siguiente ejemplo se puede apreciar un fragmento de código utilizando la herramienta grid system de Bootstrap y cuál sería el resultado visual, primero en un móvil y debajo, en una tablet.**

```
<div class="row">
        <div class="col-xs-12 col-sm-6 col-md-8">
                <img class="col-xs-12 col-sm-6 col-md-4" src="https://farm2.staticflickr.com/1437/1286617555_b2e5da513f_o_d.jpg" >
                <img class="col-xs-12 col-sm-6 col-md-4" src="https://i.vimeocdn.com/video/540843914_1280x720.jpg" >
        </div>
        <div class="col-xs-12 col-sm-6 col-md-4">
                <a href="http://google.es" target="_blank"><span>Busca en Google</span></a>
        </div>
</div>
```

**Resultado visual (móvil):**

**Resultado visual (tablet):**




**importante**

**Existen herramientas que nos permiten apreciar cómo se vería cualquier página web dependiendo el dispositivo con el que se accede a ella. Uno de ellos es Responsinator (http://responsinator.com)**

## tareas

**Siguiendo los pasos desarrollados en los ejemplos durante los subtemas 4.2, 4.3 y 4.4, el proyecto final de este taller es el desarrollo de un documento HTML multimedia. Este debe incluir, al menos, algún recurso como imagen, vídeo, audio, mapa, gráfica u otro tipo de contenido incrustado.**

**En el tema 4.2 ya vimos cómo se integra un elemento dentro de la estructura HTML, en el tema 4.3, cómo se edita visualmente un elemento y en el 4.4, cómo se hace *responsive.* Ahora es el momento de poner estos tres pasos en práctica.**

**Para ello, lo primero que hay que hacer es entrar a vuestro proyecto de Cloud9 o crear uno nuevo para este fin y crear un nuevo archivo, al que podemos llamar proyecto.html (o cualquier otro nombre, pero es necesario que tenga extensión .html). Posteriormente, iniciar la estructura e integrar los diferentes elementos, darles formato gráfico y visual. Finalmente, desarrollar el *grid system* de filas (*row*) y columnas para hacerlo *responsive.***

**Como recurso se puede utilizar el código que viene en el documento *"Esquema de estructura de un documento HTML responsive"* para tener una base con la que comenzar. Resulta muy importante que copiéis todo este código y empecéis con el. O al menos, incluir en el <head> todos estos métodos, necesarios para que el documento "funcione" de forma correcta.**

**Para ver el resultado, recordad que se debe tener el proyecto en modo *Run Project*. Después seleccionar *Share* y, abrir la url correspondiente a files. Cuando se abra, seleccionad el archivo correspondiente al proyecto (proyecto.html, por ejemplo). Según vais realzando modificaciones, tenéis que ir guardando el documento (ctrl-s) para que los cambios se hagan efectivos. Después se actualiza la ventana del navegador donde esté abierto el archivo correspondiente para ver los cambios realizados (también se puede visualizar el resultado en una pestaña dentro de Cloud9, pero el resultado no será tan versátil).**

**Para comprobar si se está realizando correctamente la parte de adaptación a diferentes dispositivos (*responsive design* utilizando *Bootstrap*), se pueden utilizar herramientas online como *responsinator.com***

### ENTREGA DE PROYECTO

**Hay tres opciones para entregar el proyecto:**

**- Descargar la carpeta del proyecto de Cloud9 y subir a la plataforma el archivo comprimido (.tar.gz) correspondiente.**

**- Enseñar la url del proyecto en el navegador. Tiene un riesgo y es que si el proyecto no está ejecutándose o la sesión está cerrada, no se podrá acceder.**

**- La opción más recomendable es que se invite al proyecto al tutor (Share > Invite > f.aguaza@gmail.com), ya que así se puede descargar los archivos necesarios directamente, visualizarlo en cualquier momento y corregir cualquier duda o fallo que se produzca cuando se esté aún desarrollando.**

## referencias


1) ***El medio es el mensaje*** es el primer capítulo del estudio más influyente de Marshall McLuhan: *Understanding Media.The Extensions of Man.* Fue publicado por primera vez en 1964, en un contexto donde se comenzaba a vislumbrar el poder de magnetismo social que tenían los medios de comunicación audiovisuales. Identifica medio con mensaje porque precisa que las propias cualidades del medio no son solo circunstancias a las que el mensaje, sino que son parte inherente del mismo y fundamentales para entender el grado de efectividad de un acto comunicativo. Se puede ampliar información sobre este paradigma en el siguiente artículo de Lance Strate, publicado en InfoAmérica: http://www.infoamerica.org/icr/n07_08/strate.pdf

2) En una entrevista publicada por clasesdeperiodismo.com: http://www.clasesdeperiodismo.com/2015/09/21/gaston-roitberg-las-redacciones-van-a-ser-cada-vez-mas-lo-parecido-a-un-staff-tecnologico/

3) **Usabilidad** es un término que, en el ámbito web, se relaciona con las "facilidades" que ofrece una interfaz gráfica para la interacción del usuario con ella. Está muy relacionado con la estandarización en la posición de algunos elementos (como los menús), los pasos que ha de dar el usuario para llegar a su destino, referencias heredadas de la maquetación en papel y un aspecto que resulte agradable para quien lo esté visualizando y adecuado al contenido.

4) Un **editor de texto plano** es aquel que no introduce ningún tipo de formato al texto (como puede hacer un editor de textos normal, del tipo word), algo muy necesario para la generación de un código "limpio". Un ejemplo clásico es el "bloc de notas" de Windows, aunque hay editores de texto plano orientados al diseño web, con herramientas que facilitan esta tarea (como Notepad++ o el editor de Cloud9).

5) Un **framework** es un conjunto de herramientas que facilitan la tarea de desarrollo de una web o una aplicación, mediante la automatización o simplificación de algunos de sus procesos. Se podría comparar como una cierta "industrialización" en el proceso "artesanal" de introducir códigos.

6) La **maquetación** es una disciplina técnica relativa al aspecto visual de una publicación, muy relacionada con la edición y orientada, sobre todo, a organizar la disposición visual de los elementos que componen una unidad de contenido, dentro de una armonía estética y una coherencia semántica.

7) ***Internet de las cosas*** es un concepto relativamente nuevo que se refiere a la interconexión digital de objetos cotidianos con internet.

8) ***Responsive design*** o diseño adaptativo es una forma de concebir la maquetación web que surge en respuesta al desarrollo de diferentes dispositivos para acceder a internet, como tablets y móviles. Se basa en la

posibilidad de que cada elemento pueda tener unas características u otras, según la anchura en px de la pantalla del dispositivo con el que se accede. Esto se realiza mediante unas reglas condicionales denominadas media query. En este enlace de w3schools, se explica su definición y cómo se implementa: http://www.w3schools.com/html/html_responsive.asp.

9) **Bootstrap:** Framework orientado al desarrollo web con múltiples herramientas y que se ha popularizado como un referente. Es de código abierto y se puede implementar en cualquier web. Tiene utilidades basadas en CSS, JavaScript y una librería de iconos, entre otras. Todas las instrucciones y recursos necesarios para instalar y utilizar Bootstrap se pueden encontrar en su web oficial: getbootsrap.com.

10) El W3C (www Consortium) fija las reglas lingüíticas de los lenguajes de la web, como **HTML5** y **CSS3**. Su web oficial es: www.w3c.es. En www.w3schools.com hay un recurso educativo oficial para aprender todos los lenguajes de la web, a través de explicaciones y ejercicios pequeños ejercicios prácticos.

## IDEAS CLAVE

- **HTML5** es la 5ª versión del Lenguaje de Marcas de Hipertexto (Hypertext Markup Language). En el ámbito de la www y los documentos multimedia, **su función es semántica**. Su unidad fundamental es la etiqueta, que sirve para indicar partes de la propia estructura, una agrupación de elementos o un elemento individual de contenido.
- Una **etiqueta** puede contener un número indefinido de etiquetas, que a su vez pueden contener otras y así sucesivamente, con lo que se producen jerarquías entre los diferentes niveles. Una etiqueta sólo está relacionada con sus "hermanas" y sus "hijas". El contenido de una etiqueta es todo lo que se encuentra entre su apertura y su cierre.
- Un **atributo** de una etiqueta determina ciertas características de un elemento. Los 3 fundamentales para la edición web son ***src***, que señala la fuente de recursos como imágenes, vídeos, audios o contenido incrustado; ***id***, que puede dar a un elemento carácter de único, al otorgarle un código identificativo que no puede repetirse; ***class***, que sirve para otorgar cualidades comunes a diferentes elementos.
- **CSS3** es la 3ª versión del lenguaje de Hoja de Estilos en Cascada (Cascade Style Sheet). **Su función es forma**l. Sirve para otorgar cualidades visuales y gráficas a prácticamente cualquier elemento, representado por una etiqueta html, una etiqueta incluida dentro de otra, una id o una class.
- El ***responsive design*** es la concepción de una web cuya distribución y cualidades visuales están optimizadas para cualquier dispositivo con el que se accede. Esto se puede conseguir gracias a unas reglas condicionales llamadas ***media query***, que permiten señalar diferentes cualidades a cada elemento, según la anchura de la pantalla del dispositivo con el que accede el usuario.
- **Bootstrap** es un ***framework*** con múltiples funcionalidades y herramientas. Orientado al CSS, permite automatizar el proceso de responsive design, gracias a su ***grid system*** (sistema de rejilla)

## REFERENCIAS BIBLIOGRÁFICAS


- Gastón Roitberg y Franco Piccato (coords.). ***Periodismo Disruptivo, dilemas y estrategias para la innovación.*** Buenos Aires, Argentina: La Crujía Ediciones, 2015.
- Steve Krug. ***No me hagas pensar (or. Don´t make me think. )***. EE.UU: New Riders Press, 2000.