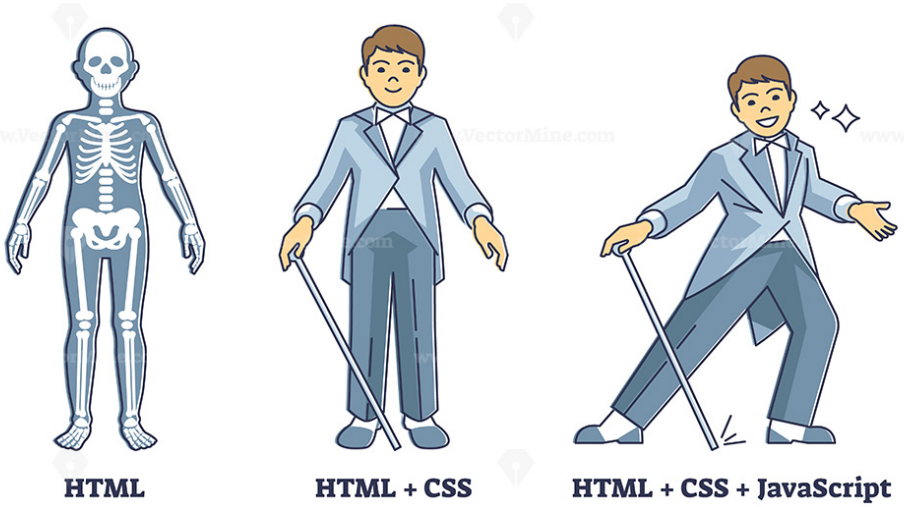
Curs de HTML5, CSS3 y JavaScript

Sergi Gotarra - Setembre-Octubre 2022

A partir del curs openwebinars:
https://openwebinars.net/academia/aprende/curso-html5-css3-javascript/



HTML5 **1**

CSS **19**

Flexbox 70

Bootstrap 82

Javascript **95**

HTML5

La Web semantica HTML5

Las etiquetas estructurales descritas anteriormente proporcionan una visualización específica de su contenido, aunque ésta puede variar ligeramente para cada navegador.

HTML5 introduce ademas etiquetas semánticas, que no aportar de base ningún comportamiento visual adicional, pero que nos permiten por un lado definir de forma semántica el significado de su contenido, lo que será muy útil para el SEO (la optimización de contenidos para buscadores). Posteriormente podremos aportar estilos adicionales a estas etiquetas mediante CSS. Las etiquetas más utilizadas son:

Estructura de cuerpo del documento:

- < header >: Grupo de elementos de introducción de un documento (como los h1, h2). No confundir con < head >, la cabecera HTML que contiene metainformación no visible.

- < nav >: Enlaces de navegación. Podría contener un menú de navegación horizontal, o lateral.

- < section >: Define una sección en un documento. Por ejemplo, la sección central con el conjunto de artículos de un blog, o podríamos tener una sección por cada categoría de artículos, con un listado de artículos dentro.

- < aside >: Contenidos vagamente relacionados con el resto del contenido de la página. Si no es visualizado, el contenido restante seguirá teniendo sentido. Por ejemplo, anuncios, u otros contenidos.

- < footer >: Pie de una página o sección.

Dentro del cuerpo:

- < article >: Contenido con identidad propia, que podría existir aportando información de manera independiente del resto del documento.

Otros elementos:

- < cite >: Título de una publicación

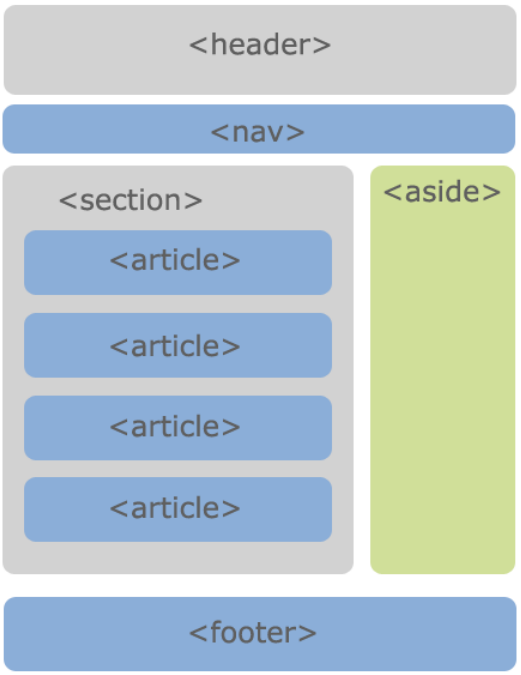
- < address >: Dirección física

- < time >: Fecha y hora

Como referencia para profundizar en este tema, además de las mencionadas añadimos la web de Mozilla Developer Network, que es bastante clara y concisa: https://developer.mozilla.org/es/docs/HTML/HTML5/HTML5_lista_elementos

Siempre que queramos añadir un bloque con contenido a nuestra página, es útil comprobar si tiene un significado semántico de entre los incluidos en las etiquetas HTML5. En caso de no coincidir con ninguno, siempre podemos utilizar el bloque de contenido neutro < div > o el de texto de linea neutro < span >, que estilaremos igualmente por CSS.

Un ejemplo de estructura de página utilizando etiquetas semánticas HTML5 sería el siguiente



Exercici 1

```
<!DOCTYPE html>

<!-- document base html5-->

<html>

    <head>

    </head>

    <body>

        <h1>Hola món</h1>

    </body>

</html>
```

Exercici 2

```
<!DOCTYPE html>

<!-- document base html5-->

<html>

    <head>

        <title>Titol del document</title>

        <!-- Indiquem quin tipus de charset farem servir-->

        <meta charset="utf8">

    </head>

    <body>

        <h1>Titol principal</h1>

        <h2>Subtitle</h2>

        <a href="http://inscastellet.cat/" >Link a inscastellet</a>

        <!-- un link pot tenir molts paràmetres com obrir-se a una
altra pàgina, a la mateixa, etc-->

    </body>

</html>
```

Exercici 3

```
<!DOCTYPE html>

<!-- document base html5-->

<html>

    <head>

        <title>Titol del document</title>

        <!-- Indiquem quin tipus de charset farem servir-->

        <meta charset="utf8">

    </head>

    <body>

        <h1>Titol principal</h1>

        <h2>Subtitle</h2>

        <a href="http://inscastellet.cat/" >Link a inscastellet</a>

        <!-- br introdueix una nova línia adicional (no requereix
tancar-se)-->

        <br>

        <!-- tenim una imatge que es diu castellet.jpg -->

        <!-- alerta als directoris de treball -->

        <img src="castellet.jpg" alt="Logo ins castellet"></img>

        <!-- <p> defineix un paràgraf, tot el que hi ha dins queda
diferenciat entre paragrafs <\p> -->

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

```
        sunt in culpa qui officia deserunt mollit anim id est
laborum.</p>

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

        sunt in culpa qui officia deserunt mollit anim id est
laborum.</p>



    </body>

</html>
```

Exercici 4

```
<!DOCTYPE html>

<!-- document base html5-->

<html>

    <head>

        <title>Titol del document</title>

        <!-- Indiquem quin tipus de charset farem servir-->

        <meta charset="utf8">

    </head>

    <body>

        <h1>Titol principal</h1>

        <h2>Subtitle</h2>

        <a href="http://inscastellet.cat/" >Link a inscastellet</a>

        <!-- br introdueix una nova línia adicional (no requereix
tancar-se)-->

        <br>

        <!-- tenim una imatge que es diu castellet.jpg -->

        <!-- alerta als directoris de treball -->

        <img id="logo-lloc-web"  class="imatges" src="castellet.jpg"
alt="Logo ins castellet"></img>

        <!-- id identifica un únic element del DOM-->

        <!-- class identifica tots els elements del tipus que
vulguem-->

        <!-- D'aquesta manera quan faci un canvi a una classe canviaran
tots els elements-->



        <!-- Llista Ordenada-->
```

```
        <ul>
            <li>Element 1</li>
            <li>Element 2</li>
        </ul>

        <!-- podem anidar una llista dins d'una altra-->
        <ul>
            <li>Element 1</li>
            <li>Element 2</li>
            <ul>
                <li>Element pepe</li>
                <li>Element juan</li>
            </ul>
        </ul>

        <!-- si volem una llista ordenada canviem ul Unordered List per
ol Ordered List-->
        <ol>
            <li>Element 1</li>
            <li>Element 2</li>
            <ul>
                <li>Element pepe</li>
                <li>Element juan</li>
            </ul>
        </ol>
```

```
        <!-- si volem que al picar un element sigui un enllaç, molt
fàcil -->

        <!-- Si li piquem va a l'inici de la pàgina perquè no està
direccionat només posa #-->

        <ol>

            <li><a href="#">Element 1</a></li>

            <li>Element 2</li>

            <ul>

                <li>Element pepe</li>

                <li>Element juan</li>

            </ul>

        </ol>


        <!-- els enllaços poden ser absoluts o relatius -->

        <!-- absolut: http:\\.... -->

        <!-- relatius: arxiu2.html -->

        <!-- relatius: ../subcarpeta2/arixu3.html -->

        <!-- a un punt especial de la pàgina:    -->


        <ol>

            <li><a href="4.html#identificador2">Element 1</a></li>

            <li>Element 2</li>

            <ul>

                <li>Element pepe</li>

                <li>Element juan</li>
```

```
            </ul>
        </ol>

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

            sunt in culpa qui officia deserunt mollit anim id est
laborum.</p>

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

            sunt in culpa qui officia deserunt mollit anim id est
laborum.</p>

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

                sunt in culpa qui officia deserunt mollit anim id est
laborum.</p>

        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit
```

```
in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident,

            sunt in culpa qui officia deserunt mollit anim id est
laborum.</p>



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

            sunt in culpa qui officia deserunt mollit anim id est
laborum.</p>



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

                sunt in culpa qui officia deserunt mollit anim id est
laborum.</p>



        <h2  id="identificador2"> CAPÇALERA 2 </h2>

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

                sunt in culpa qui officia deserunt mollit anim id est
laborum.</p>
```

```
    </body>

</html>
```

Exercici 5

```
<!DOCTYPE html>

<html>

    <head>

        <title>Titol del document</title>

         <meta charset="utf8">

    </head>

    <body>

        <!-- header defineix una zona a dins del body, sempre a la part
superior-->

        <header>

            <h1>Titol principal</h1>

            <h2>Subtitle</h2>

            <!-- definim un navegador, com si fora una llista per ara
les referencies no apunten enlloc-->

            <nav>

                <li><a href="#">Inici</a></li>

                <li><a href="#">Productes</a></li>

                <li><a href="#">Serveis</a></li>

                <li><a href="#">Contacte</a></li>


            </nav>


        </header>


        <!-- deifnim una nova etiqueta section (similar al div)-->
```

```
        <section>
            <!-- dins d'una secció podem definir diferents articles-->
            <article>
                <h1>Casa lloguer 1</h1>
            </article>
            <article>
                <h1>Casa lloguer 1</h1>
            </article>
            <article>
                <h1>Casa lloguer 1</h1>
            </article>
            <article>
                <h1>Casa lloguer 1</h1>
            </article>
        </section>

    </body>
</html>
```

Exercici 6

```
<!DOCTYPE html>

<html>

    <head>

        <title>Titol del document</title>

         <meta charset="utf8">

    </head>

    <body>

        <!-- header defineix una zona a dins del body, sempre a la part
superior-->

        <header>

            <h1>Titol principal</h1>

            <h2>Subtitle</h2>


            <nav>

                <li><a href="#">Inici</a></li>

                <li><a href="#">Productes</a></li>

                <li><a href="#">Serveis</a></li>

                <li><a href="#">Contacte</a></li>

            </nav>


        </header>


        <!-- deifnim una nova etiqueta section (similar al div)-->

        <section>
```

```
            <!-- afegim uns links per a anar a ampliar informació-->
            <article>
                <h1>Casa lloguer 1</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article>
                <h1>Casa lloguer 1</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article>
                <h1>Casa lloguer 1</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article>
                <h1>Casa lloguer 1</h1>
                <a href="#">Ampliar informació</a>
            </article>
        </section>

        <aside>
            <!-- indica elements relacionats pero no sempre
necessaris-->
            <nav>
                    <h1>Links relacionats</h1>
                    <ul>
```

```
                        <li><a
href="http://science.nasa.gov/astrophysics/">Astrofísica - NASA
Science</a></li>

                        <li><a
href="http://www-astro.physics.ox.ac.uk/">Astrofísica | Universidad de
Oxford</a></li>

                        <li><a href="http://aston.blogejemplo.com/">El
blog de astrofísica de Aston</a></li>

                    </ul>

                </nav>

        </aside>

        <!-- footer sempre anirà al final de la pàgina-->

        <footer>

            &copy;2022 EL meu lloguer

            <!-- &copy és una de les moltes entitats que es poden fer
servir -->

        </footer>



    </body>

</html>
```

CSS

Los **selectores** son lo que definen **sobre qué elementos se aplican las reglas**.

Pueden ser una combinación de:

- Nombre de etiquetas HTML
- Nombres de clases, precedidos por carácter “.”
- Identificador de elemento, precedido por carácter “#”

```
a {

        color: orange;

}
```

```
.example_extract {

        background-color: #ccc;

}
```

```
#example_list {

        border: 1px solid #3e2f2d;

}
```

En el ejemplo anterior, podemos ver las siguientes reglas:

- Todas las etiquetas < a > serán de color naranja (orange).
- Todos los elementos de clase **example_extract**tendrán un color de fondo de código #ccc (gris)
- El elemento con el identificador **example_list** tendrá un borde sólido de 1 pixel de ancho, de color #3e2f2d

```
a.selected {

        color: blue;

}
```

En este ejemplo, la regla afectará a todas las etiquetas < a > que tengan especificadas la clase **selected**.

Otra opción es definir en el selector condiciones de anidamiento, por ejemplo, es decir, que la regla afecta a ciertos elementos dentro de otros elementos, separados por espacios. A su vez, esto se puede combinar con múltiples selectores, dando lugar a reglas como esta:

```
#example_list a.button {
        background-color: blue;
        color: white;
        border-radius: 7px;
}
```

Esta regla indica que todos los elementos < a > que tengan especificada la clase **button**, y que estén dentro del elemento con identificador **example_list**, tendrán de color de fondo azul (blue), el color de texto será blanco (white) y los bordes serán redondeados con un radio de 7 píxeles.

Si tuviéramos a su vez esta regla junto a las anteriores, tendríamos dos sitios donde se define el color de ciertos enlaces. Cuando esto sucede, el orden de precedencia es:

- Son más prioritarias las reglas más específicas
- En igualdad de condiciones, se aplican en orden secuencial, siendo la última en aparecer la que finalmente se visualiza

Hemos visto también en estos ejemplos que existen propiedades múltiples, donde tras el nombre de la propiedad se especifican tras los dos puntos varios valores.

Si quisiéramos crear varias reglas que apliquen las mismas propiedades, pero a diferentes selectores, podemos hacerlo separando los selectores diferentes por comas, como en el siguiente ejemplo.

```
section.projects h1, section.projects h2, section.projects h3 {
        color: #222;
}
```

Esta regla (o más correctamente, conjunción de reglas) establece que el color de texto de los elementos < h1 >, < h2 >, < h3 > que se encuentren dentro de una etiqueta < section > con el atributo **class** con valor **projects**.

Otra característica que podemos utilizar en los selectores son los **pseudo clases** mediante el carácter ":", que nos permiten seleccionar solo los elementos en un estado concreto, o las **especificaciones de valores de atributos**, que se incluyen entre corchetes "[" y "]".

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

a:hover {
        text-decoration: underline;
}

input[type=button], input[type=submit], submit {
        font-size: 15px;
        font-family: sans-serif;
}
```

En el ejemplo anterior, hemos conseguido que las etiquetas < a > no vengan subrayadas por defecto. Pero cuando pasemos el ratón sobre ellas (la pseudo clase **hover**), si se aplicará la propiedad de subrayado.

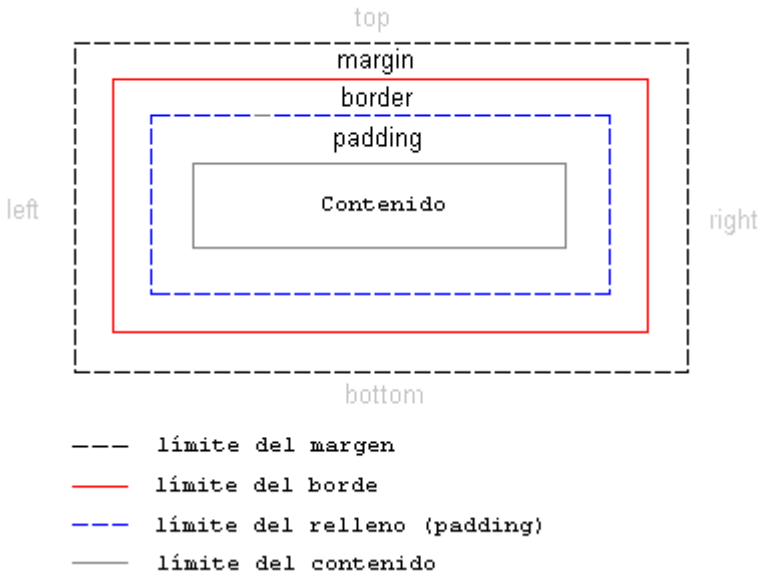
También hemos hecho que las etiquetas < input > cuyo atributo type tenga el valor button o submit, así como la etiqueta < submit > (los tres casos son casi equivalentes), tengan siempre un texto de tamaño 15 píxeles, y un tipo de letra generico sans-serif (sin bordes en los extremos).

Una buena guía de todas las propiedades y pseudo clases disponibles en CSS es la disponible en Mozilla Developer Network:
https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

Un aspecto importante de las propiedades de los elementos HTML que merece una explicación detallada es el llamado modelo de caja.

Cada elemento HTML ocupa un tamaño en la página web, que viene definido por:

- margin: el margen exterior del elemento
- border: el tamaño del borde del elemento
- padding: el margen interior del elemento



Por otro lado, tenemos que hablar de cómo se posicionan los elementos en la página. Hemos visto cómo algunos elementos, como **h1**, ocupan por defecto el 100% del ancho del contenedor donde se encuentren. Estos elementos tienen una propiedad CSS llamada **display** cuyo valor es de tipo **block**, y entre ellos tenemos:

- h1, h2, h3…
- Etiquetas semánticas como header, nav, section, article
- Etiqueta de bloque genérico div

Otros elementos de tipo inline como el texto se van acumulando en la misma línea, hasta llegar al final de esta y continuar en la línea siguiente. Se comportan de este modo:

- El texto
- Etiqueta de contenido inline generico span

Algunos elementos, como las imágenes, tienen un comportamiento de tipo **inline-block**, de manera que por defecto se muestran insertadas en medio del texto.

Estos valores por defecto pueden ser alterados, de manera que convirtamos un elemento que por defecto era de tipo **block** en **inline-block**. Al hacer esto, en lugar de ocupar el

100% del ancho del contenedor, pasará a ocupar el mínimo ancho posible para albergar su contenido, a menos que especifiquemos un ancho específico con la propiedad CSS **width**.

También se comportará como las imágenes, de manera que aparecerá incluido en medio de la línea de texto donde se haya definido. Si a continuación del elemento teníamos algún texto, si no añadimos un retorno de línea o nuevo párrafo, aparecerá inmediatamente después del elemento.

Otra propiedad CSS que puede alterar el posicionamiento de los elementos es **float**, que al definirse con valores **right** o **left**, provocará que el elemento afectado omita totalmente su posición actual para alinearse lo más posible al lado indicado dentro de su misma línea, acumulando el resto de elementos al lado opuesto. Además, para elementos tipo block, pasarán a comportarse como **inline-block** respecto a su ancho, es decir, éste se ajustara o bien al definido explícitamente, o al mínimo según los elementos que contenga.

Si varios elementos consecutivos se definen con el mismo valor de **float**, estos empezarán a acumularse (separados por sus márgenes), hasta forzar una o varias nuevas líneas. En la práctica, es más deseable conseguir este efecto utilizando la propiedad **display** con valor **inline-block**, ya que existen ciertos fallos en los navegadores antiguos (Firefox en este caso) que hace que el comportamiento de elementos **float** no sea igual que en el resto de navegadores.

Por último, tenemos la propiedad **position**, que podemos definir con el valor **fixed** para que la posición del elemento se extraiga del flujo natural del documento, y pase a ser definido directamente en un tamaño en pixel relativo a los lados de la ventana del navegador mediante las propiedades **top, bottom, left o right.**

También podemos posicionar el elemento relativo a los bordes de otro elemento contenedor. Para ello, definiremos la propiedad CSS **position** como **relative** en el elemento contenedor, de manera que cualquier elemento interior tendrá la propiedad position como **absolute**.

¿Todo esto resulta confuso? Veámoslo mejor con un ejemplo completo. Sobre el mismo documento con la lista de ejemplos que preparamos para las etiquetas semánticas HTML5, vamos a añadir la siguiente hoja de estilos:

```
/* Reglas generales */

* { font-family: sans-serif; }

body { margin: 0 }

a { text-decoration: none; }

a:hover { text-decoration: underline; }

/* Cabecera y navegación */
```

```
header {
    background-color: #DFD493;
    color: white;
    position: fixed;
    top: 0;
    width: 100%
}
header h1 {
    margin: 0; padding: 10px 10px;
    font-size: 25px;
}
nav {
    background-color: #EAE2B8;
    padding: 10px 10px;
    width: 100%
}
nav ul {
    list-style-type: none;
    display: inline-block;
    margin: 0; padding: 0;
}
nav ul li {
    display: inline-block;
    margin: 0 6px;
}
/* Secciones centrales */
```

```
section {
    width: 980px;
    margin: 30px auto;
    padding-top: 60px;
}
article {
    display: inline-block;
    background: #ccc;
    width: 240px; height: 240px;
    margin: 2px 2px;
    padding: 15px 15px;
}
article h1 {
    font-size: 15px;
}
/* Pie */
footer {
    background-color: #CBCD00;
    padding: 10px 10px;
    margin: 0;
}
```

Y el resultado es el siguiente:

Fichero de ejemplo "Etiquetas semánticas HTML5" del curso sobre HTML, CSS y Javascript

Un cambio bastante radical frente al mismo documento sin estilos. Vemos como un buen código HTML se la base adecuada para poco a poco tener un aspecto sofisticado a base de ir definiendo reglas CSS.

Noves propietats CSS3

Con la llegada de HTML5 también llega una nueva versión de CSS3 que, además de estandarizar mucho más el efecto de reglas ya conocidas, introduce algunas nuevas enormemente útiles para conseguir resultados muy efectistas de manera sencilla con muy poco código.

- **border-radius:**
- Hasta hace poco implementada como un idiom de cada navegador, especifica el radio para el que los bordes de un elemento será redondeado. **opacity:**
- Indica en tanto por uno el nivel de opacidad (o transparencia) del elemento. **transition-duration:**
- Cuando definamos con una pseudo clase como hover alguna propiedad diferente, la transición al pasar el ratón sobre el elemento no será instantánea, sino que se realizará de manera fluida según la duración indicada. **box-shadow:**
- Define la sombra del elemento **text-shadow:**
- Define la sombra del texto contenido en el elemento **transform:**
- Define una transformación sobre el elemento original, que puede tener como valores:
    - **scale:**
    - escalado **rotate:**
    - rotación **skew:**
    - inclinación **translate:**

traslación

Otras adiciones interesantes son sobre las propiedades que define colores. Con CSS3 podemos definir de color de fondo complejos gradientes, mediante el valor de propiedad gradient sin tener que utilizar ninguna imagen para ello. Construir el código para estos gradientes de forma visual sin es sencillo utilizando servicios como los de la siguiente página: http://www.colorzilla.com/gradient-editor/.

Exercici 7

```
<!DOCTYPE html>
<html>
    <head>
        <title>Titol del document</title>
        <meta charset="utf8">
        <!-- aquí hi ha la referència al fitxer de estils-->
        <!-- és la manera més habitual de afegir-los-->
        <link href="7.css" rel="stylesheet" type="text/css">

    </head>
    <body>
        <header>
            <h1 id="tit">Titol principal</h1>
            <h2>Subtitle</h2>

            <nav>
                <li><a href="#">Inici</a></li>
                <li><a href="#">Productes</a></li>
                <li><a href="#">Serveis</a></li>
                <li><a href="#">Contacte</a></li>
            </nav>

        </header>
```

```
<!-- deifnim una nova etiqueta section (similar al div)-->
<section>
    <!-- afegim uns links per a anar a ampliar informació-->
    <article class="casalloguer">
        <h1>Casa lloguer 1</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 1</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 1</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article  class="casalloguer">
        <h1>Casa lloguer 1</h1>
        <a href="#">Ampliar informació</a>
    </article>
</section>


<!-- footer sempre anirà al final de la pàgina-->
<footer>
```

```
            &copy;2022 EL meu lloguer

            <!-- &copy és una de les moltes entitats que es poden fer
servir -->

        </footer>


    </body>

</html>
```

7.css

```
/* això és un comentari en css */


/* a tots els elements de tipus link (a) se'ls aplicarà el que hi ha
dins el parèntesis

    en aquest cas es posaran en color cyan */


a {

    color:cyan;

}




/* com referenciar a un identificador */

#tit {

    color: rgb(255, 0, 102);

}
```

```
/* podem fer que els elements li tinguin un determinat color, marge
exterior i interior (margin i padding)*/


article.casalloguer {

    background-color: tomato;

    color: white;

    border: 2px solid black;

    margin: 20px;

    padding: 20px;

}
```

Exercici 8

```
<!DOCTYPE html>

<html>

    <head>

        <title>Titol del document</title>

        <meta charset="utf8">

        <!-- aquí hi ha la referència al fitxer de estils-->

        <!-- és la manera més habitual de afegir-los-->

        <link href="8.css" rel="stylesheet" type="text/css">




    </head>

    <body>

        <header>

            <h1 id="tit">Titol principal</h1>

            <h2>Subtitle</h2>


            <!-- possem un id al menú per a poder aplicar css-->


            <nav id="menu-principal">

                <li><a href="#">Inici</a></li>

                <li><a href="#">Productes</a></li>

                <li><a href="#">Serveis</a></li>

                <li><a href="#">Contacte</a></li>

            </nav>
```

```
</header>


<!-- definim una nova etiqueta section (similar al div)-->
<section>
    <!-- afegim uns links per a anar a ampliar informació-->
    <article class="casalloguer">
        <h1>Casa lloguer 1</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 1</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 1</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article  class="casalloguer">
        <h1>Casa lloguer 1</h1>
        <a href="#">Ampliar informació</a>
    </article>
</section>
```

```
        <footer>
            &copy;2022 EL meu lloguer
        </footer>

    </body>
</html>
```

8.css

```
/* això és un comentari en css */

/* a tots els elements de tipus link (a) se'ls aplicarà el que hi ha
dins el parèntesis
    en aquest cas es posaran en color cyan */

a {
    color:rgb(18, 32, 229);
}

/* com referenciar a un identificador */
#tit {
    color: rgb(255, 0, 102);
```

```
}



/* es pot referenciar només amb el punt perquè sigui global a tot el
dom */

.casalloguer {

    background-color: tomato;

    color: white;

    border: 2px solid black;

    margin: 20px;

    padding: 20px;

}


#menu-principal{

    background-color: rgb(194, 192, 255);

    color: blue;

    border: 2px solid rgb(37, 169, 240);

    margin: 20px;

    padding: 20px;

}


/* possem els menús en horitzontal, i sí, podem possar a la vegada més
d'una referència */

ul, li {

    padding: 0,0;

    margin: 2px ,2px;
```

```
    /* atenció la característica display que és molt potent */
    display: inline-block;
}
```

Exercici 9

```
<!DOCTYPE html>

<html>

    <head>

        <title>Titol del document</title>

        <meta charset="utf8">

        <link href="9.css" rel="stylesheet" type="text/css">




    </head>

    <body>

        <header>

            <h1 id="tit">Titol principal</h1>

            <h2>Subtitle</h2>


            <!-- possem un id al menú per a poder aplicar css i posar
el menú horitzontal-->


            <nav id="menu-principal">

                <li><a href="#">Inici</a></li>

                <li><a href="#">Productes</a></li>

                <li><a href="#">Serveis</a></li>

                <li><a href="#">Contacte</a></li>

            </nav>
```

```
</header>


<!-- Ara el contingut s'adapta al tamany de la pantalla !-->
<section>
    <article class="casalloguer">
        <h1>Casa lloguer 1</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 2</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 3</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article  class="casalloguer">
        <h1>Casa lloguer 4</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 5</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
```

```
                <h1>Casa lloguer 6</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article class="casalloguer">
                <h1>Casa lloguer 7</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article  class="casalloguer">
                <h1>Casa lloguer 8</h1>
                <a href="#">Ampliar informació</a>
            </article>
        </section>




        <footer>
            &copy;2022 EL meu lloguer
        </footer>

    </body>
</html>
```

9.css

```
/* això és un comentari en css */


/* a tots els elements de tipus link (a) se'ls aplicarà el que hi ha
dins el parèntesis

    en aquest cas es posaran en color cyan */


a {

    color:rgb(18, 32, 229);

}




#tit {

    color: rgb(255, 0, 102);

}




.casalloguer {

    background-color: tomato;

    color: white;

    border: 2px solid black;

    margin: 10px 10px;

    padding: 10px 10px;

     /* atenció amb inline-block */
```

```
    display: inline-block;
    width: 30%;
    height: 200px;
    /* podem definir tamanys amb píxels o percentatges */
}

#menu-principal{
    background-color: rgb(194, 192, 255);
    color: blue;
    border: 2px solid rgb(37, 169, 240);
    margin: 20px;
    padding: 20px;
}

ul, li {
    padding: 0,0;
    margin: 2px ,2px;
    display: inline-block;
}

/* veure model https://lenguajecss.com/css/modelo-de-cajas/que-es/   */
```

Exercici 10

```
<!DOCTYPE html>

<html>

    <head>

        <title>Titol del document</title>

        <meta charset="utf8">

        <link href="10.css" rel="stylesheet" type="text/css">




    </head>

    <body>

        <header>

            <h1 id="tit">Titol principal</h1>

            <h2>Subtitle</h2>


            <!-- possem un id al menú per a poder aplicar css i posar
el menú horitzontal-->


            <nav id="menu-principal">

                <li><a href="#">Inici</a></li>

                <li><a href="#">Productes</a></li>

                <li><a href="#">Serveis</a></li>

                <li><a href="#">Contacte</a></li>

            </nav>
```

```
</header>


<!-- Ara el contingut s'adapta al tamany de la pantalla !-->
<section>
    <article class="casalloguer">
        <h1>Casa lloguer 1</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 2</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 3</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article  class="casalloguer">
        <h1>Casa lloguer 4</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 5</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
```

```
                <h1>Casa lloguer 6</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article class="casalloguer">
                <h1>Casa lloguer 7</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article  class="casalloguer">
                <h1>Casa lloguer 8</h1>
                <a href="#">Ampliar informació</a>
            </article>
        </section>




        <footer>
            &copy;2022 EL meu lloguer
        </footer>

    </body>
</html>
```

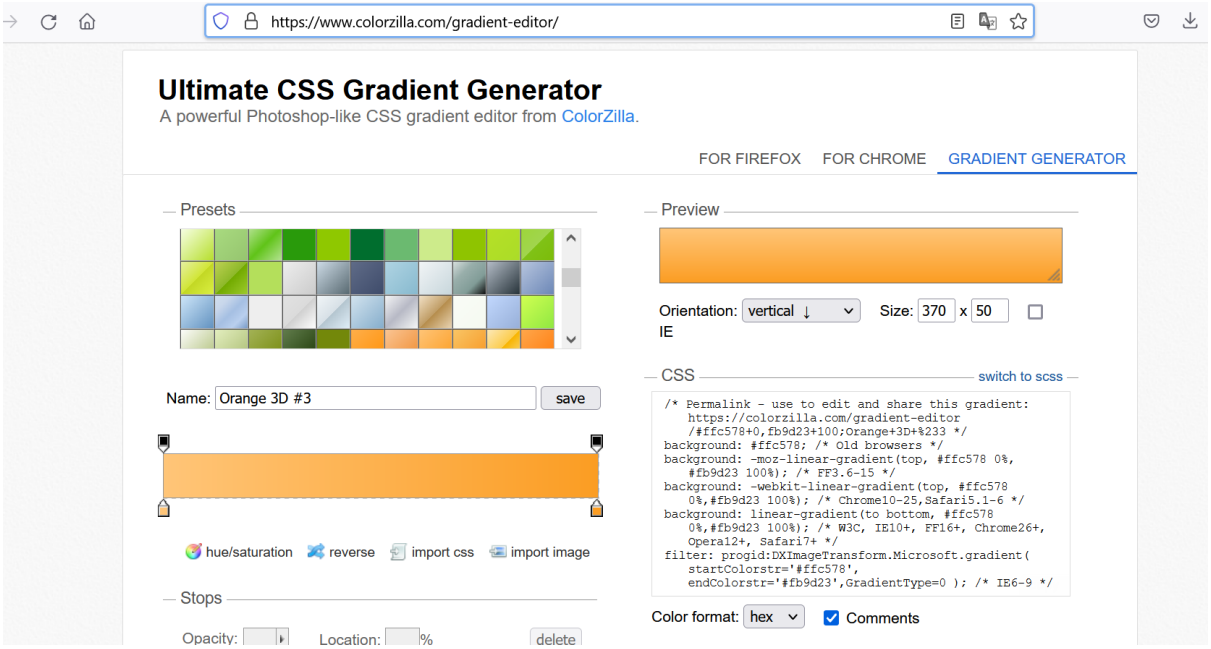
10.css

```
a {
    color:rgb(18, 32, 229);
}



#tit {
    color: rgb(255, 0, 102);
}



.casalloguer {
    background-color: tomato;
    color: white;
    border: 2px solid black;
    margin: 10px 10px;
    padding: 10px 10px;
     /* atenció amb inline-block */
    display: inline-block;
    width: 30%;
    height: 200px;
     /* noves funcions */
    opacity: 0.6;
    border-radius: 20px;
```

```
    box-shadow: 3px 3px 10px rgb(109, 58, 58);

}

#menu-principal{
    background-color: rgb(194, 192, 255);
    color: blue;
    border: 2px solid rgb(37, 169, 240);
    margin: 20px;
    padding: 20px;
}

ul, li {
    padding: 0,0;
    margin: 2px ,2px;
    display: inline-block;
}

/* footer  */
footer {
    background-color: rgb(7, 218, 218);
    padding: 10px, 10px;
    margin: 0;

}
```

Fem un exemple amb gradiants de colors

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



Formularis

Los formularios web permiten introducir datos y enviarlos mediante el navegador. Solo con conocimientos de HTML, CSS y JavaScript no podemos hacer casi nada con los datos de estos formularios, pero darles estilo, y validar su contenido previamente a su envío es una labor fundamental que hacer con estas tecnologías.

Todo formulario debe tener sus elementos en una etiqueta < form >, cuyos principales atributos son:

- **action:** URL destino que cargará el navegador al enviar el formulario, pasando a ésta el valor de todos los campos rellenados.
- **method:** Método de envío, que puede tener valor **get** para codificar los datos del formulario de manera visible en la URL destino, o **post** para enviarse durante la conexión de manera no tan visible por el usuario (aunque no totalmente privada).
- **enctype:** Codificación a realizar en el envío del formulario. Se especifica con el valor multipart/form-data para el envío de formularios con campos fichero.

Dentro de la etiqueta < form >, nos encontraremos las siguientes etiquetas:

- **< input >**: Control de formulario:

    - **type=”text”:** Permite introducir una línea de texto
    - **type=”radio”:** Elemento que permite marcar uno solo de un conjunto que tenga el mismo valor en el atributo name.
    - **type=”checkbox”:** Elemento que permite ser marcado o desmarcado de forma autónoma.
    - **type=”hidden”:** Elemento invisible, que puede almacenar un valor y será enviado al enviar el formulario. Puede ser cambiado mediante programación con JavaScript.
    - **type=”button”:** Botón, sin ningún comportamiento al pulsarlo por defecto, pero que puede ser cambiado mediante programación con JavaScript.
- **< textarea >:** Permite introducir varias líneas de texto. Los atributos cols y rows indican las filas y columnas de tamaño
- **< select >:** Selector entre una de múltiples opciones
    - **< option >:** Especifica una de las opciones seleccionables. Si tiene definido el atributo **default**, será la por defecto nada más cargar el formulario. Si tiene definido el atributo value, el valor de éste será el enviado en los datos del formulario aunque se muestre otro texto.
- **< button> :** Equivalente a < input type=”button” >
-
- **< submit >:** Muestra un botón que, al pulsarlo, ocasiona que se envíe el formulario. En todos los botones al especificar el parámetro value se establece el texto dentro del mismo. Equivalente a < input type=”submit” >

Trabajaremos más a fondo con los elementos de formulario en la sección sobre JavaScript.

11.html

```
<!DOCTYPE html>

<html>

    <head>

        <title>Titol del document</title>

        <meta charset="utf8">

        <link href="11.css" rel="stylesheet" type="text/css">




    </head>

    <body>
```

```
<header>
    <h1 id="tit">Titol principal</h1>
    <h2>Subtitle</h2>

    <nav id="menu-principal">
        <li><a href="#">Inici</a></li>
        <li><a href="#">Productes</a></li>
        <li><a href="#">Serveis</a></li>
        <li><a href="#">Contacte</a></li>
    </nav>

</header>

<!-- Ara el contingut s'adapta al tamany de la pantalla !-->
<section>
    <article class="casalloguer">
        <h1>Casa lloguer 1</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 2</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 3</h1>
```

```
            <a href="#">Ampliar informació</a>
        </article>
        <article  class="casalloguer">
            <h1>Casa lloguer 4</h1>
            <a href="#">Ampliar informació</a>
        </article>
        <article class="casalloguer">
            <h1>Casa lloguer 5</h1>
            <a href="#">Ampliar informació</a>
        </article>
        <article class="casalloguer">
            <h1>Casa lloguer 6</h1>
            <a href="#">Ampliar informació</a>
        </article>
        <article class="casalloguer">
            <h1>Casa lloguer 7</h1>
            <a href="#">Ampliar informació</a>
        </article>
        <article  class="casalloguer">
            <h1>Casa lloguer 8</h1>
            <a href="#">Ampliar informació</a>
        </article>
    </section>

    <!-- afegim un video-->
```

```
        <video controls>
            <source src="video.mp4"/>
        </video>


        <!-- afegim un formulari
        es pot enviar amb el mètodo post o amb get (post es invisible, get es veu a la url)
        action és qui reb el resultat del formulari
        -->
        <form method="post" action="https://result.php">
            <!-- elements del formulari-->
            <br>
            <input type="text" />
            <br>
            <input type="checkbox"/>
            <input type="button" value="pulsar"/>
            <br>
            <select>
                <option>opcio 1</option>
                <option>opcio 2</option>
            </select>
            <textarea>HOLA QUE TAL NO TAN MAL</textarea>
            <input type="submit" value = "enviar">
            <input type="hidden" name="ocult" value="una cosa que s'enviara " />
```

```
            <!-- el hidden no es veu pero podem enviar el valor de
value quan enviem el formulari-->


        </form>


        <footer>

            &copy;2022 EL meu lloguer

        </footer>


    </body>

</html>
```

```
a {
    color:rgb(18, 32, 229);
}


#tit {
    color: rgb(255, 0, 102);
}


.casalloguer {

    color: white;
    border: 2px solid black;
    margin: 10px 10px;
    padding: 10px 10px;
     /* atenció amb inline-block */
    display: inline-block;
    width: 30%;
    height: 200px;
     /* noves funcions */
    opacity: 0.6;
    border-radius: 20px;

    box-shadow: 3px 3px 10px rgb(109, 58, 58);
```

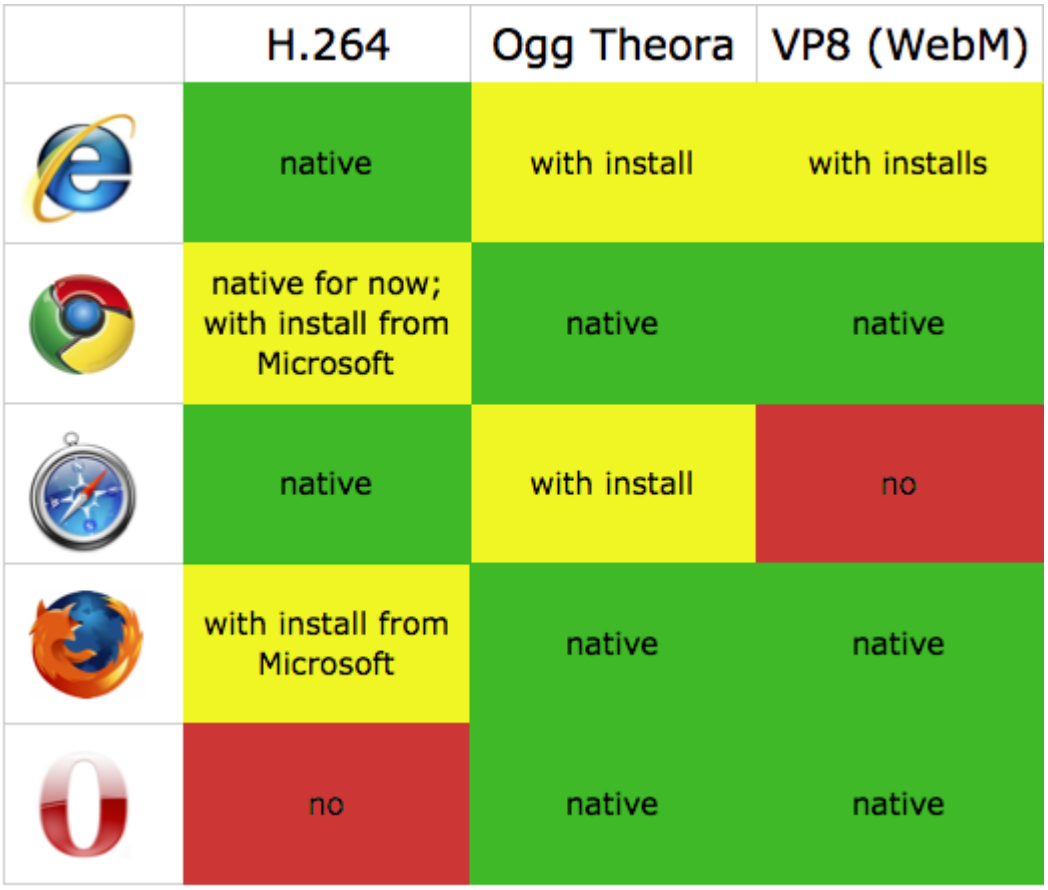
```
    /* Obrim ultimate css gradient generator    */

    /* Permalink - use to edit and share this gradient:
https://colorzilla.com/gradient-editor/#ffc578+0,fb9d23+100;Orange+3D+%
233 */

    background: #ffc578; /* Old browsers */

    background: -moz-linear-gradient(top,  #ffc578 0%, #fb9d23 100%);
/* FF3.6-15 */

    background: -webkit-linear-gradient(top,  #ffc578 0%,#fb9d23 100%);
/* Chrome10-25,Safari5.1-6 */

    background: linear-gradient(to bottom,  #ffc578 0%,#fb9d23 100%);
/* W3C, IE10+, FF16+, Chrome26+, Opera12+, Safari7+ */

    filter: progid:DXImageTransform.Microsoft.gradient(
startColorstr='#ffc578', endColorstr='#fb9d23',GradientType=0 ); /*
IE6-9 */




}



#menu-principal{

    background-color: rgb(194, 192, 255);

    color: blue;

    border: 2px solid rgb(37, 169, 240);

    margin: 20px;

    padding: 20px;

}



ul, li {
```

```
    padding: 0,0;
    margin: 2px ,2px;
    display: inline-block;
}

/* footer  */
footer {
    background-color: rgb(7, 218, 218);
    padding: 10px, 10px;
    margin: 0;

}

/*estilem els botons */
/* només s'aplica als elements que tenen un atribut en concret */

input[type=button]{
    font-size: 40px;
    padding: 30px 30px;
}
```

Video i audio:

HTML5 ha incorporado con algo de controversia etiquetas < video > y < audio > para incrustar elementos multimedia en las páginas web sin necesidad de recurrir a plugins de terceros como Flash.

El problema es que los responsables de navegadores no se han puesto de acuerdo en cuanto a qué formato de codificación deben tener esos ficheros. La situación al respecto en estos momentos en la siguiente:



Fuente: https://msdn.microsoft.com/es-es/hh552485.aspx

En Septiembre de 2015 Microsoft ha anunciado que dotará en el futuro de soporte nativo a VP8 en su navegador Edge, lo que da buenas esperanzas para la estandarización mediante este formato.

La solución pasa por incluir en la etiqueta < video > o < audio > varias fuentes con diferentes codificaciones, pero el mismo contenido, de manera que el navegador seleccione el que sea compatible. Un ejemplo de cómo se consigue esto es el siguiente:

```
< video controls >
  < source src="devstories.webm"
          type='video/webm;codecs="vp8, vorbis"'/>
  < source src="devstories.mp4"
          type='video/mp4;codecs="avc1.42E01E, mp4a.40.2"'/>
< /video >
```

Por lo demás, a estas alturas el uso general de etiquetas y atributos debe ser bastante conocido, por lo que si queremos hacer un uso complejo de < video > o < audio >, solo nos queda investigar las referencias a todos los atributos específicos de estas:

- https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Using_HTML5_audio_and_video
- http://www.html5rocks.com/en/tutorials/video/basics/

Las tipografías web requieren aclarar ciertos conceptos previos. En principio, podemos definir que un texto utilice una tipografía mediante la propiedad CSS font-family. Pero solo tendrá en cuenta tipografías muy genéricas, o aquellas que ya están instaladas en el ordenador.

Si lo que queremos es utilizar una tipografía original, CSS3 define un estándar para enlazar nuestra web con el fichero tipográfico con la información necesaria para representarla. Pero de nuevo aquí hay varios estándares y no ha habido un acuerdo definitivo sobre qué formato utilizar, por lo que lo más sencillo es delegar este problema en servicios de tipografía web como el gratuito Google Fonts o Adobe Typekit.

- https://www.google.com/fonts
- https://typekit.com/

Utilizando el primero como ejemplo, podemos buscar una tipografía que nos parezca interesante para nuestra página. Para utilizarla, deberemos enlazar en la sección < head > de nuestro documento con la ubicación de la tipografía de la manera:

```
< link href='https://fonts.googleapis.com/css?family=Lobster'
rel='stylesheet' type='text/css' >
```

Y luego en el fichero CSS, tendremos que agregar la siguiente regla en el selector al que queramos dotar de esta tipografía:

```
font-family: 'Lobster', cursive;
```

El especificar cursive como fuente alternativa permitirá que si por alguna razón no se puede utilizar la primera opción indicada, haya una segunda con una visualización diferente a la que tienen por defecto otros elementos de la página.

CSS Responsive

Necesitamos en primer lugar conocer las diferentes resoluciones de pantalla de los dispositivos que van a ser nuestro objetivo. Podemos consultar una extensa tabla en la siguiente referencia: http://dpi.lv/

Comparemos algunas características de dispositivos como los siguientes:

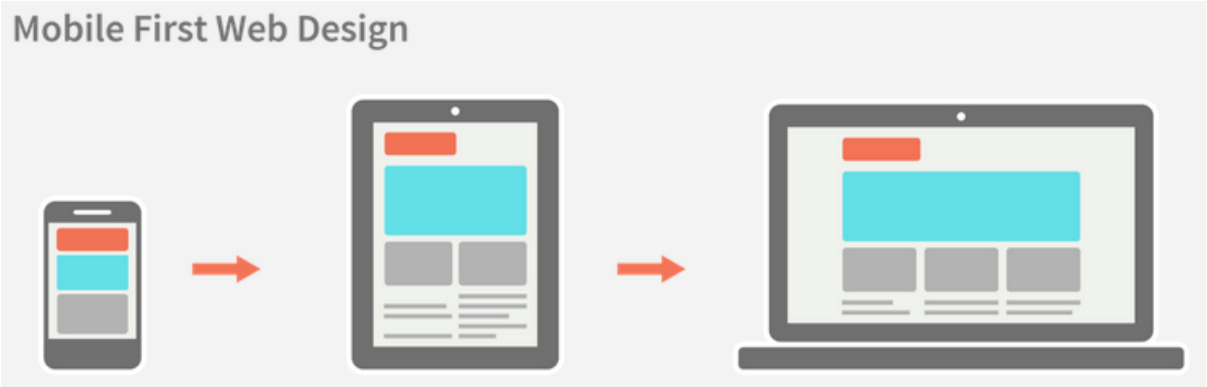
	Tamaño diagonal pantalla	Resolución	Densidad Píxeles por Pulgada (PPI)
Monitor PC convencional	21"	1920 x 1080	105
iPhone 6 Plus	5,5"	1080x1980	401
Samsung Galaxy S6	5.1"	1920x1080	532
iPad Air	9,7”	2048 x 1536	264

Nos damos cuenta de que tan solo fijándonos en la resolución, no tenemos una información fidedigna de las características del sistema utilizado por el usuario. Un monitor de un PC convencional puede tener la misma resolución que un iPhone 6 o un Samsung Galaxy S6,

pero al ser estas pantallas mucho más pequeñas (lo que se comprueba examinando la densidad de píxeles), al usuario le costaría mucho interactuar con la misma página si se representara de forma equivalente en estos dispositivos.

Además, una página diseñada para ordenador tiene en cuenta que la interacción principal se realizará con el ratón, donde la precisión del cursor es muy grande. En un dispositivo táctil, el tamaño de nuestros dedos sobre un móvil tiene un tamaño considerable, y sería deseable que los elementos a pulsar fueran de un tamaño similar a nuestra huella.

Afrontar este reto de diseño no es fácil. Para ello está demostrado que es mucho más práctico adoptar una filosofía mobile first, donde en primer lugar se diseñe cómo se va a visualizar el contenido en un dispositivo móvil, de manera conceptual y creando el código HTML y CSS; para luego especificar cómo alteramos el interfaz para aprovechar las características de la visualización en escritorio.



Fuente: http://metamonks.com/mobile-first-vs-responsive/

En general, el diseño mobile first debe tener en cuenta algunos patrones muy comunes de usabilidad:

- La cabecera no debe ser muy grande, de manera que el contenido no quede muy abajo.
- Los menús de navegación deberán aparecer condensados, desplegándose solo al pulsar alguna opción para ello.
- Un conjunto de artículos es apropiado mostrarlos de arriba a abajo, ocupando cada uno el 100% del ancho (o el 50% en dispositivos como tablets con algo más de tamaño).
- Cualquier contenido lateral no relacionado con el contenido principal, es mejor trasladarlo a que se muestre al pie del documento.
- Cualquier enlace que se podría mostrar como un texto, es preferible hacerlo como un botón que tenga un tamaño fácil de pulsar.
- Si un artículo se puede expandir para visitar su contenido ampliado, hacer tanto el título como la imágen y el texto resumen del artículo “pulsables” para visitar la versión expandida de la misma.

Conseguir todo puede parecer muy complicado, pero en realidad se consigue siempre de la misma manera. Utilizaremos una nueva característica de CSS3 llama **media queries**.

Utilizando la palabra clave **media**, podemos establecer entre corchetes que un conjunto de reglas solo se tenga en cuenta para un medio en concreto (la pantalla, o la versión de impresión), o para un rango de resolución específico. En nuestro ejemplo podemos utilizar unas reglas como estas:

```
@media screen and (max-width: 300px) {
    nav { display: none; }
    article {
        width: 100%;
        height: 100px;
    }
}
```

En el resultado comprobamos al reducir el tamaño del navegador que el comportamiento de la lista de ejemplos ya no es conformar una red de 3 elementos de ancho, sino una lista donde todos tiene un ancho del 100%. También hemos ocultado el menú superior, aunque sería necesario añadir un menú alternativo desplegable.

Otro factor que tenemos que tener en cuenta, es que los dispositivos táctiles al facilitar el hacer zoom pellizcando la pantalla, pueden mostrar un ancho de ésta diferente del ancho del documento. Para forzar a que el documento se muestre exactamente con el ancho de la pantalla, utilizaremos una etiqueta en la cabecera como esta:

```
< meta name="viewport" content="width=device-width" >
```

El ejemplo anterior nos ha funcionado reduciendo el tamaño del navegador, y funcionaría con pantallas de poca resolución. Pero los dispositivos de pantalla tipo “retina”, con muchos pixels por pulgada, requerirán reglas en las que utilicemos este factor en lugar del ancho en píxeles.

```
@media
  only screen and ( min-device-pixel-ratio: 2),
```

```
  only screen and ( min-resolution: 192dpi),
  only screen and ( min-resolution: 2dppx) {
    /* Reglas para dispositivos retina */
}
```

Hemos visto cómo afecta tanto el ratio de píxeles de la pantalla como la densidad para la experiencia a la hora de construir CSS. Pero, ¿qué pasa con las imágenes?

En una pantalla de mucha resolución pero pequeño tamaño, vamos a querer ampliar la imagen respecto al tamaño de píxel de un ordenador de escritorio. Pero si lo hacemos utilizando la misma imagen original, estaremos desaprovechando la mayor resolución a nuestra disposición.

La solución está en utilizar ficheros de imagen alternativos para pantallas de alta densidad. Cuando se definan mediante CSS y media queries, tenemos que especificar en estas reglas alternativas los nuevos ficheros como cualquier otra regla particular para esa resolución. Más referencias en: http://www.html5rocks.com/en/mobile/high-dpi/

Cuando los utilicemos en etiquetas , podemos emplear la propiedad srcset para especificar un conjunto de alternativas a la imagen original definida en la propiedad src.

Un ejemplo de uso de srcset sería el siguiente:

```
< img alt="my awesome image"
  src="banner.jpeg"
  srcset="banner-HD.jpeg 2x, banner-phone.jpeg 640w, banner-phone-HD.jpeg 640w 2x" >
```

12 html

```
<!DOCTYPE html>

<html>

    <head>

        <title>Titol del document</title>

        <meta charset="utf8">

        <link href="12.css" rel="stylesheet" type="text/css">




    </head>

    <body>

        <header>

            <h1 id="tit">Titol principal</h1>

            <h2>Subtitle</h2>


            <nav id="menu-principal">

                <li><a href="#">Inici</a></li>

                <li><a href="#">Productes</a></li>

                <li><a href="#">Serveis</a></li>

                <li><a href="#">Contacte</a></li>

            </nav>


        </header>


        <!-- Ara el contingut s'adapta al tamany de la pantalla !-->
```

```
<section>
    <article class="casalloguer">
        <h1>Casa lloguer 1</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 2</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 3</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article  class="casalloguer">
        <h1>Casa lloguer 4</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 5</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 6</h1>
        <a href="#">Ampliar informació</a>
    </article>
```

```
            <article class="casalloguer">
                <h1>Casa lloguer 7</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article  class="casalloguer">
                <h1>Casa lloguer 8</h1>
                <a href="#">Ampliar informació</a>
            </article>
        </section>


        <footer>
            &copy;2022 EL meu lloguer
        </footer>

    </body>
</html>
```

12.css

```
a {
    color:rgb(18, 32, 229);
}



#tit {
    color: rgb(255, 0, 102);
}



.casalloguer {

    color: white;
    border: 2px solid black;
    margin: 10px 10px;
    padding: 10px 10px;
     /* atenció amb inline-block */
    display: inline-block;
    width: 30%;
    height: 200px;
     /* noves funcions */
    opacity: 0.6;
    border-radius: 20px;
```

```
    box-shadow: 3px 3px 10px rgb(109, 58, 58);
    background: #ffc578; /* Old browsers */



}

#menu-principal{
    background-color: rgb(194, 192, 255);
    color: blue;
    border: 2px solid rgb(37, 169, 240);
    margin: 20px;
    padding: 20px;
}

ul, li {
    padding: 0,0;
    margin: 2px ,2px;
    display: inline-block;
}

/* footer  */
footer {
    background-color: rgb(7, 218, 218);
    padding: 10px, 10px;
    margin: 0;
```

```
}



/* Set the background color of body to tan */
body {
    background-color: tan;
  }

  /* On screens that are 992px or less, set the background color to
blue */
  @media screen and (max-width: 992px) {
    body {
      background-color: blue;
    }
  }

  /* On screens that are 600px or less, set the background color to
olive */
  @media screen and (max-width: 600px) {
    body {
      background-color: rgb(0, 128, 2);
    }
    .casalloguer{
        display: block;
```

```
        background-color: blue;
    }
    nav {
        display:none;
    }
}
```

Flexbox

Contenedor e items

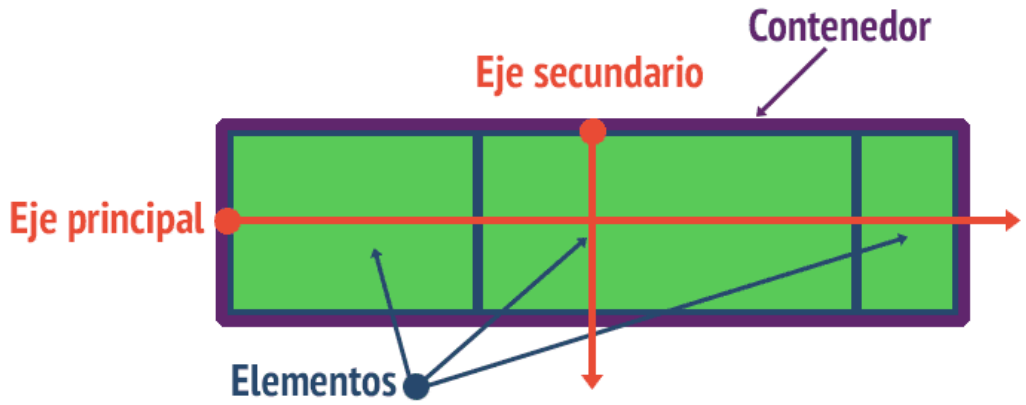
Las cosas que se pueden hacer con Flexbox involucran a dos partes fundamentales. Una es el contenedor y otra los ítems que están dentro de él.

Más adelante detallaremos todo esto, pero puedes ir entendiendo que habrá propiedades CSS pensadas para alterar el modo en el que un contenedor se va a comportar y otras propiedades que sirven para indicar cómo se deben representar sus ítems internos. Por ejemplo, en el contenedor principal podremos decir que los elementos los queremos en la horizontal, o en la vertical. Mientras que en un item interno podremos indicar cosas como el tamaño que deben ocupar en relación a otros o el orden en el que deben aparecer.

Los ejes en Flexbox

En flexbox vamos a tener dos ejes. El eje principal, por defecto, es el eje horizontal y el eje secundario que es el eje vertical. Por defecto significa que nosotros como diseñadores podremos cambiar este comportamiento, de modo que el eje principal sea el vertical y el secundario el horizontal.

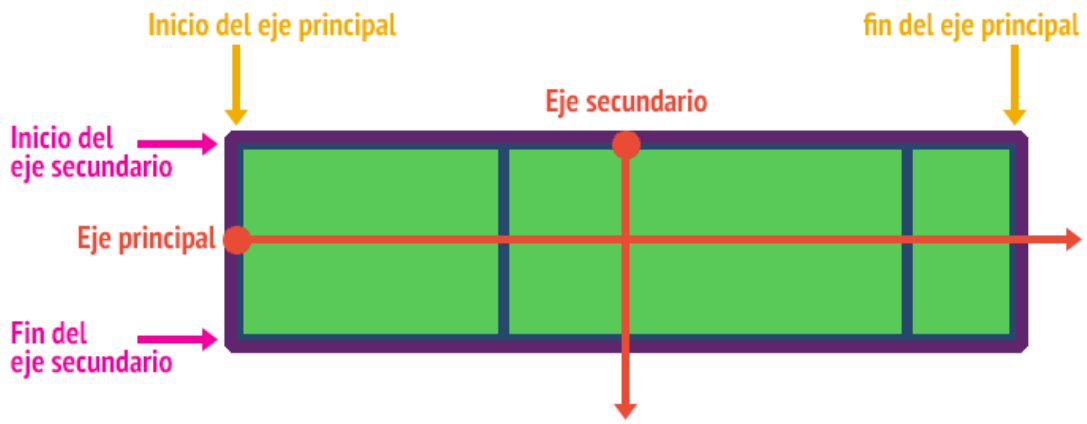
Estos ejes, principal y secundario, implican el modo en el que los ítems se van a posicionar. Todo es configurable, pero para que nos hagamos una idea con un ejemplo, si el eje principal es la horizontal, los ítems se pondrán uno al lado del otro. Si el eje principal fuera la vertical, los ítem se colocarían uno debajo (o arriba) del otro.



Nota: El atributo flex-direction es el que nos permitirá intercambiar el eje principal y secundario. Más adelante lo veremos con ejemplos. Inicio y fin de los ejes.

A la hora de alinear un texto, por ejemplo, con CSS tenemos los conceptos left y right, indicando que queremos una alineación a izquierda o derecha. Esto no funciona justamente igual en Flexbox. En este caso tenemos los conceptos de inicio y fin (start / end).

Como se ha dicho, en flex tenemos dos ejes: principal y secundario, pues existirá un inicio y un fin para cada uno de los ejes.



Dimensiones del contenedor

Aquí las dimensiones continúan siendo la altura y la anchura, definidas con width y height. Sin embargo, la principal de estas dimensiones dependerá de cuál de sus ejes sea el principal. Date cuenta que, si la dirección es de arriba a abajo, entonces la dimensión principal será la altura.

Ejercicio práctico con Flexbox

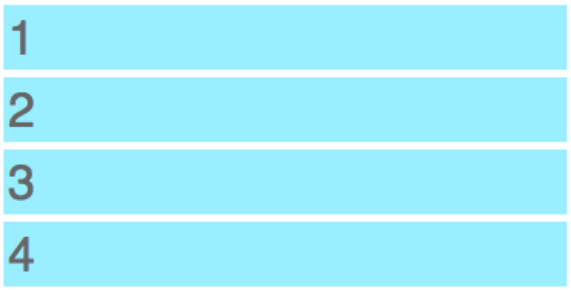
Vamos a partir de un HTML sencillo y aplicaremos varios cambios con Flexbox para ver la transformación a la hora de representarse en la página.

```
<section>
  <article>1</article>
  <article>2</article>
  <article>3</article>
  <article>4</article>
</section>
```

Partimos también de un CSS básico solamente para darle un poco de color a los elementos y poder verlos un poco separados en la página.

```
body {
  font-size: 2em;
  font-family: sans-serif;
  color: #666;
}
article {
  background-color: #9ef;
  margin: 5px;
  padding: 3px;
}
```

Tal cual están estos elementos, su representación en el navegador sería más o menos esta:



Ahora vamos a empezar a aplicar algo de flexbox. Lo primero que tenemos que hacer es decirle al contenedor principal (el SECTION) que debe comportarse como un elemento "flex". Esto lo conseguimos con el atributo "display", aplicando el valor "flex".

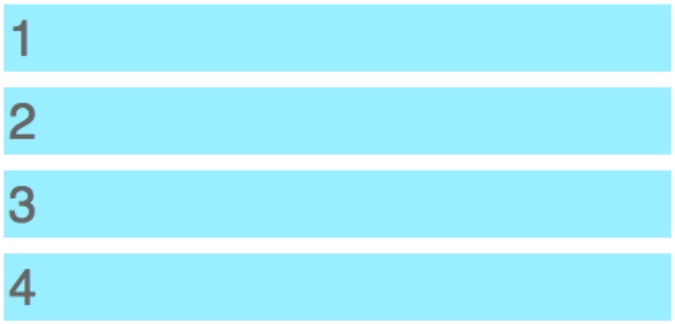
```
section {
  display: flex;
}
```

Solo por haber añadido este comportamiento, nuestros elementos van a colocarse en la página de otra manera totalmente distinta.

Como no hemos indicado nada, el eje predeterminado es el horizontal y por ello es que aparecen uno al lado del otro. Pero podríamos indicar que se colocasen uno debajo del otro si cambiamos el atributo "flex-direction".

```
section {
  display: flex;
  flex-direction: column;
}
```

Ahora los elementos se muestran como puedes ver en la siguiente imagen:



Vamos a hacer una pequeña alteración en nuestro código para conseguir que, estando dispuestos en el eje horizontal, los elementos tengan espacio entre ellos de modo que se distribuyan uniformemente en todo el contenedor.

```
section {
  display: flex;
  flex-direction: row;
  justify-content: space-around;
}
```

Ahora para acabar este primer acercamiento a Flexbox, vamos a ver cómo podríamos conseguir que los ítem tengan una anchura que permita ocupar todo el espacio disponible en el eje principal. Lo conseguimos con "flex-grow", pero ten en cuenta que esta propiedad ya no depende del contenedor, sino de los item, elementos internos, en nuestro caso los ARTICLE.

Nota: Creo que es obvio, pero lo menciono por si acaso. En mi ejemplo he decidido usar etiquetas SECTION y ARTICLE, pero estas propiedades de flexbox puedes aplicarlas a todo tipo de elementos, independientemente de la etiqueta escogida.

```
article {
  background-color: #9ef;
  margin: 5px;
  padding: 3px;
  flex-grow: 1;
}
```

Habiendo colocado a todos los elementos el mismo valor de flex-grow (1) estamos produciendo que el tamaño que van a ocupar sea idéntico para todos. Se verán como la siguiente imagen.

Display flex

Al contenedor principal en un esquema Flexbox es al que le asignamos "display: flex". Esta propiedad hace que cambien las reglas con las cuales sus hijos van a ser representados en la página.

```
.contendor-flex {

  display: flex;

}
```

Tan sencillo como eso! a partir de este momento, todos los elementos en la página con la clase "contenedor-flex" se comportarán según las reglas de Flexbox. Ésto implica que sus

hijos se van a posicionar de una manera distinta a la habitual. Por lo tanto, debe quedar claro que el propio contenedor no se va a ver afectado, sólo sus hijos.

Display inline-flex

Además del display flex tenemos también el valor "inline-flex". Si conocemos los elementos "inline-block", la diferencia fundamental es la misma que tienen respecto a los elementos "block" normales, que se comportan como un bloque, pero no se exanden para ocupar todo el espacio en la horizontal.

```
.contendor-flex {

  display: inline-flex;

}
```

En resumen, con inline-flex es como si tuviéramos un elemento inline-block, donde sus hijos se comportan con las reglas de Flexbox.

Una vez el contenedor es "flex" o "inline-flex" puedo aplicarle toda una serie de propiedades adicionales para personalizar todavía más su comportamiento. Las veremos a continuación.

Propiedad flex-direction

Esta propiedad nos sirve para definir la dirección del flujo de colocación de los elementos. Tiene que ver con los ejes que conocimos en el artículo anterior, pudiendo marcar si los elementos se van a colocar todos en la misma fila, o si se van a colocar en una columna, pero además también permite indicar el orden de los item, normal o reverso.

Permite usar estos valores:

- row (valor predeterminado): Indica que los elementos se colocan en una fila, uno al lado del otro, de izquierda a derecha.
- row-reverse: se colocan en una fila, pero con orden de derecha a izquierda.
- column: se colocan uno debajo del otro, en orden los primeros arriba.
- column-reverse: se colocan en una columna, pero los primeros aparecerán abajo.

Podemos experimentar con esta propiedad con un HTML como este. (De hecho verás que el HTML es muy parecido al del ejercicio anterior y es que con esto nos es suficiente para probar las distintas facetas de Flexbox).

```
<div class="flex-container">

  <div class="item">1</div>

  <div class="item">2</div>

  <div class="item">3</div>

  <div class="item">4</div>
```

```
</div>
```

Ahora vamos a aplicar unos estilos. Principalmente lo que nos interesa es aplicar el CSS al elemento con class="flex-container", pero aplicaremos estilos a todos los elementos para que podamos apreciar mejor la posición de las cajas.

```
body {
  font-size: 2.5em;
  font-family: sans-serif;
  color: #ddd;
}
.flex-container {
  display: flex;
  flex-direction: column-reverse;
}
.item {
  background-color: #369;
  margin: 2px;
  padding: 5px;
  flex-grow: 1;
}
```

Como al contenedor flex le hemos colocado flex-direction: column-reverse; observarás que los elementos se colocan uno debajo del otro y con orden contrario al que aparecen en el código HTML.

Propiedad flex-wrap

Sirve para indicar si queremos que haya saltos de línea en los elementos que se colocan en el contenedor, si es que éstos no caben en el espacio disponible.

De manera predeterminada con Flexbox los elementos se colocan en el eje de la horizontal, en una fila. Si los elementos tienen unas dimensiones tales que no quepan en el contenedor, el comportamiento flex hará que se intenten agrupar en la fila de manera que quepan bien sin saltar de línea, pero también podemos configurarlo para hacer que, si no caben, se pasen a la línea siguiente.

- nowrap (predeterminado): hace que nunca se produzcan saltos de línea.
- wrap: hace que si no caben, entonces se coloquen en la siguiente línea.
- wrap-reverse: El salto de línea se producirá al contrario, o sea, hacia arriba.

Nota: Si tenemos configurado "nowrap" el navegador hará lo que pueda para hacer que los elementos quepan en la fila (si es que flex-direction es row o row-reverse), llegando a alterar las dimensiones definidas en el width de los ítem si fuera necesario. Sin embargo, si por mucho que el navegador lo intente, siguen sin caber los elementos ahí dependerá de otros estilos CSS lo que podrá ocurrir. Con otras propiedades de CSS podremos conseguir que los elementos desborden el contenedor, que no se vean los que no quepan, etc. Por ejemplo usarás la propiedad "overflow" de toda la vida.

```
.flex-container {

  display: flex;

  flex-direction: row;

  flex-wrap: nowrap;

}

.item {

  background-color: #369;

  width: 30%;

  padding: 5px;

}
```

En este caso hemos colocado "flex-wrap: nowrap" y además cada uno de los ítem hemos indicado que debe tener una anchura de 30%. Recuerda que en nuestro HTML teníamos 4 ítem dentro del flex-container y sin embargo, cuando veamos el resultado obtendremos algo como esto:

Es obvio que 4 contenedores a 30% cada uno no cabrían en la horizontal, pero con flexbox el navegador hace un esfuerzo para ajustarse en una fila.

Ahora bien, si cambiamos para "flex-wrap: wrap" entonces sí se producirá el salto de línea:

```
.flex-container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
}
```

Nota: Si se te ocurre poner los contenedores a 33% puedes apreciar que aun así no caben 3 en la horizontal, cuando sí debería. Si es el caso, el problema no es Flexbox. Tendrías que ver si acaso por culpa de los margin los elementos no tengan espacio, o dependiendo de la combinación de box-sizing pueden afectar los padding o border. Recuerda la recomendación de usar "box-sizing: border-box".

Propiedad flex-flow

Esta propiedad no aporta nada nuevo, pues simplemente es un atajo para escribir de 1 sola vez flex-direction y flex-wrap. El valor predeterminado es "row nowrap"

```
.flex-container {
  display: flex;
  flex-flow: row wrap;
}
```

Propiedad justify-content

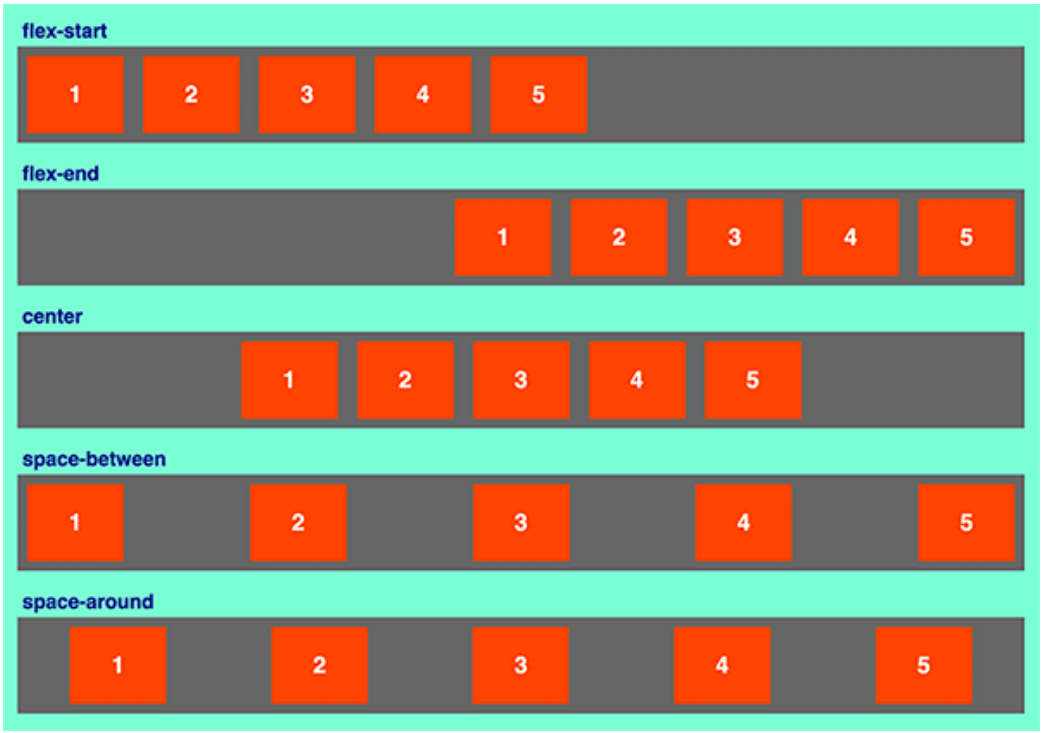
Esta propiedad es muy útil para indicar cómo se van a colocar los justificados y márgenes de los ítems. Puedes indicar que vayan a justificados al inicio del eje o al final del eje o que

a la hora de distribuirse se coloque un espacio entre ellos o un espacio entre ellos y los bordes.

Es interesante como para tratarla de manera independiente y así poder ver varios ejemplos de ella. Veamos simplemente sus posibles valores:

- flex-start: Añade los elementos a partir del inicio del eje principal.
- flex-end: Añade los elementos a partir del final del eje principal.
- center: los elementos se centran en el espacio del contenedor, siempre con respecto al eje principal.
- space-between: hace que los elementos se distribuyan con un espacio proporcional entre ellos, siendo que los ítem de los extremos se sitúan en el borde del contenedor.
- space-around: es parecido a space-between en el sentido de dejar un espaciado proporcional, sin embargo, en esta ocasión se deja también espacio entre el borde del contenedor y los ítem de los extremos.

Lo veremos todo más claro observando la siguiente imagen:



En próximos artículos abordaremos más prácticas de uso de justify-content.

Propiedad align-items

Esta propiedad es muy similar a la propiedad anterior, justify-content, solo que ahora estamos alineando con respecto al eje secundario y no el principal.

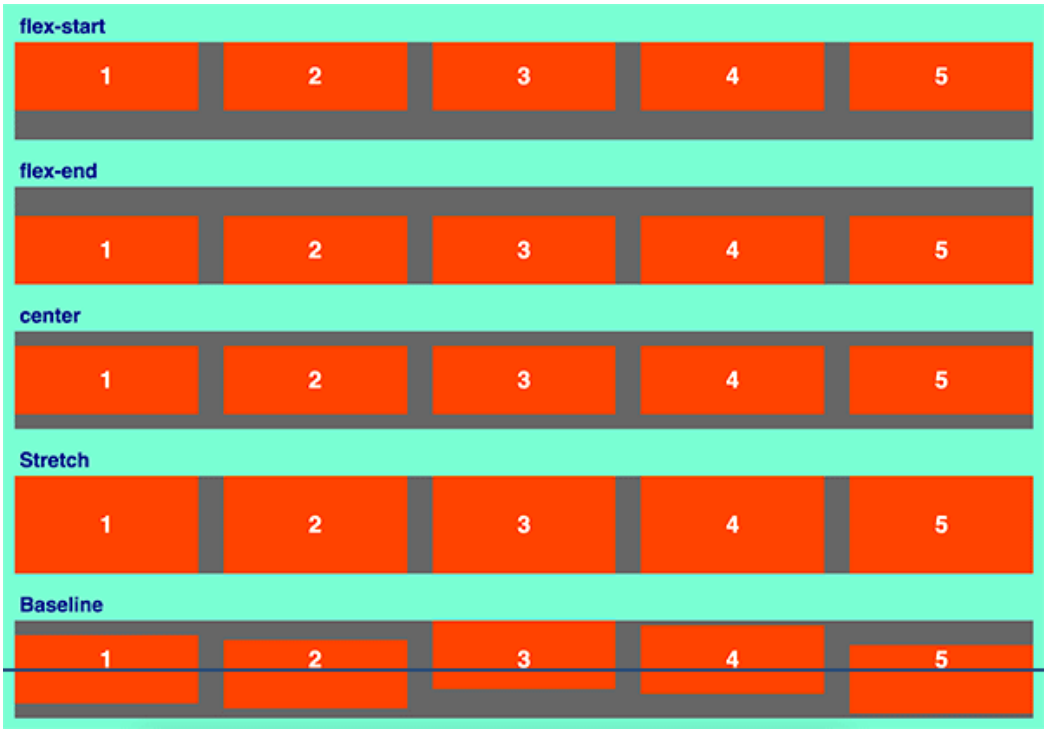
En el caso de un contenedor flex cuyo eje principal está en la horizontal, entonces align-items servirá para obtener el alineamiento en el otro eje (vertical, de arriba a abajo).

En definitiva, align-items nos ofrece el tan deseado alineamiento vertical que hemos echado en falta en CSS históricamente.

También merece hacer ejemplos específicos para verlo con más detalle, por lo que ahora nos limitaremos a enumerar sus posibles valores con una breve descripción.

- flex-start: indica que se posicionarán al comienzo del eje secundario.
- flex-end: se posicionarán al final del eje secundario.
- center: se posicionarán en el centro del eje secundario.
- stretch: ocuparán el tamaño total del eje secundario (a no ser que hayamos marcado que esos elementos tengan un tamaño diferente).
- baseline: para el posicionamiento de los elementos se tendrá en cuenta el texto que hay escrito dentro.

Como una imagen vale más que mil palabras, se pueden ver aquí los distintos efectos con bastante claridad.



Propiedad align-content

Esta propiedad sólo aplica cuando dispones de varias líneas de elementos en el contenedor flexbox. El efecto que conseguiremos será una alineación y separación de las filas en el eje secundario.

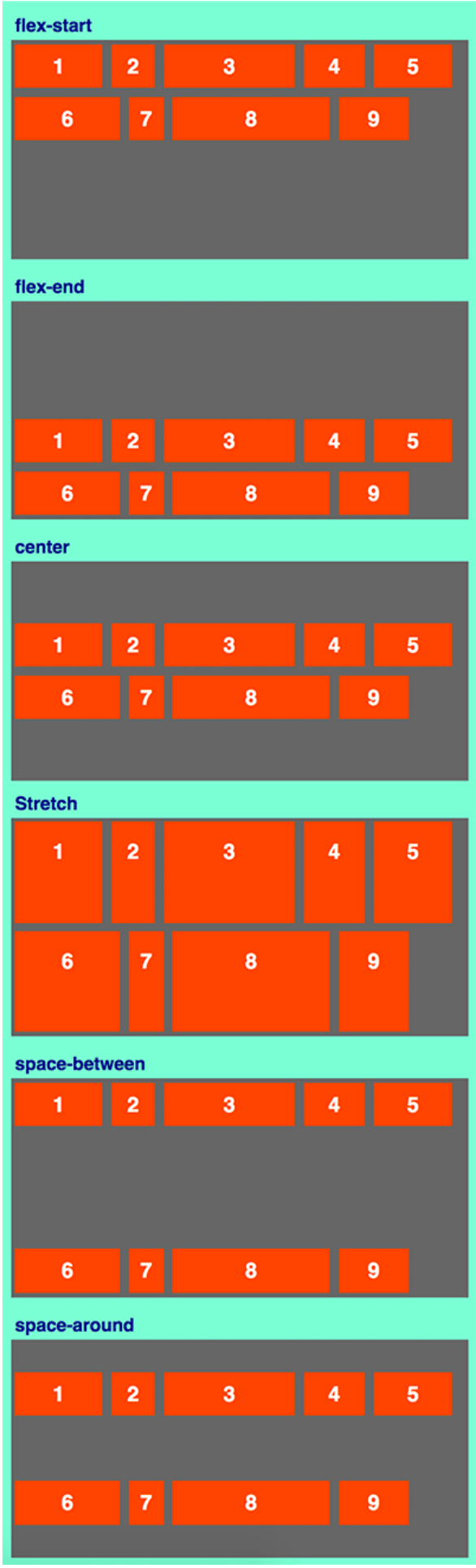
Nota: Para conseguir varias líneas de elementos en el contenedor flex necesitarás aplicarles un "flex-flow: wrap" y por supuesto que haya suficientes elementos, con suficiente anchura, para que se necesiten varias filas para su distribución.

Al final, el efecto que conseguimos con align-content es parecido al que conseguimos con align-items, en el sentido que aplicará al eje secundario su efecto de distribución, solo que

aquí no estamos indicando la colocación de una única fila, sino de todas las filas. Además se parece también a justify-content en el sentido que estamos definiendo la separación entre ítems, pero afectando a filas de ítems en vez de ítems sueltos.

- flex-start: indica que las filas se colocarán todas pegadas entre sí (obviamente no aparecerán exactamente pegadas si le hemos colocado un margin), desde el inicio del eje secundario.
- flex-end: las filas se colocarán pegadas entre sí, pero esta vez pegadas al final del eje secundario.
- center: se posicionarán en el centro del eje secundario, pegadas entre sí.
- stretch: Sus dimensiones crecerán para ocupar todo el espacio disponible (a no ser que se haya colocado una dimensión diferente en los elementos).
- space-between: indica que las filas se separarán entre sí, dejando un espacio proporcional entre ellas.
- space-around: indica que las filas se separarán, dejando un espacio entre ellas proporcional, también con el borde.

Creo que se entenderá mejor a la vista de la siguiente imagen.


flex-start
1
2
3
4
5
6
7
8
9
flex-end
1
2
3
4
5
6
7
8
9
center
1
2
3
4
5
6
7
8
9
Stretch
1
2
3
4
5
6
7
8
9
space-between
1
2
3
4
5
6
7
8
9
space-around
1
2
3
4
5
6
7
8
9

De: https://desarrolloweb.com/articulos/conceptos-principales-flexbox.html

Bootstrap

viewport: la finestra de l'aplicacio

Layout:

Contenedores:

 té diferents tamany depenent del viewport

 sempre utilitza tot el viewport

Exemple:

bootstrap.html

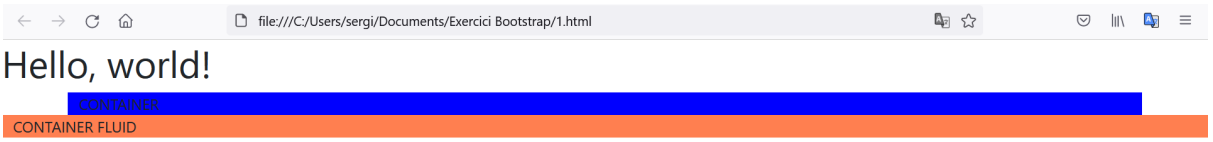
```
<!doctype html>

<html lang="en">

  <head>

    <!-- Required meta tags -->

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1">



    <!-- Bootstrap CSS -->

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.m
in.css" rel="stylesheet"
integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoq
yl2QvZ6jIW3" crossorigin="anonymous">

    <link rel="stylesheet" href="bootstrap.css" type="text/css"
media="all">

    <title>Hello, world!</title>

  </head>

  <body>

    <h1>Hello, world!</h1>
```

```
    <div class="container">CONTAINER</div>

    <div class="container-fluid">CONTAINER FLUID</div>




    <!-- Optional JavaScript; choose one of the two! -->


    <!-- Option 1: Bootstrap Bundle with Popper -->

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bun
dle.min.js"
integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q
+8nbTov4+1p" crossorigin="anonymous"></script>


    <!-- Option 2: Separate Popper and Bootstrap JS -->

    <!--

    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper
.min.js"
integrity="sha384-7+zCNj/IqJ95wo16oMtfsKbZ9ccEh31eOz1HGyDuCQ6wgnyJNSYdr
Pa03rtR1zdB" crossorigin="anonymous"></script>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min
.js"
integrity="sha384-QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmW
l5/YESvpZ13" crossorigin="anonymous"></script>

    -->

  </body>

</html>
```

Bootstrap.css

```
.container {
    background: blue;
}
.container-fluid{
    background: coral;
}
```

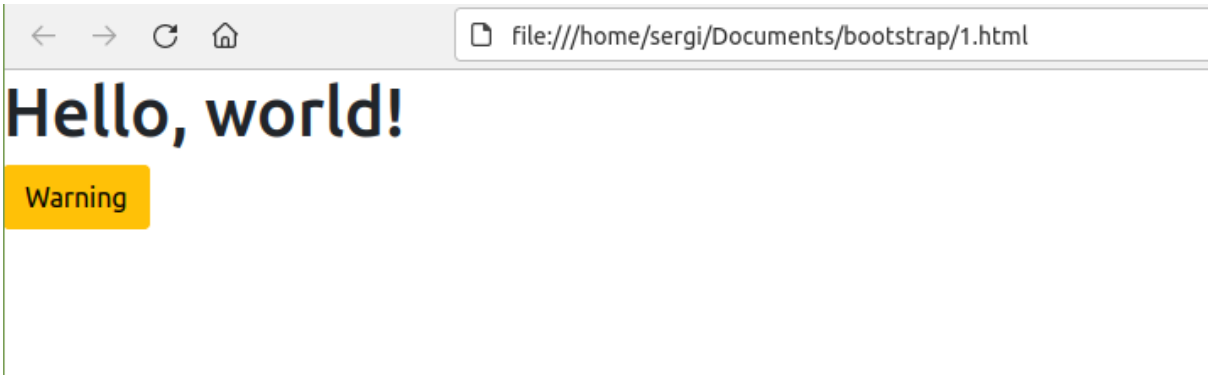
Resultat



bootstrap2.html

```
<!doctype html>

<html lang="en">

 <head>

   <!-- Required meta tags -->

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1">



   <!-- Bootstrap CSS -->

   <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.m
in.css" rel="stylesheet"
integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoq
yl2QvZ6jIW3" crossorigin="anonymous">



   <title>Hello, world!</title>

 </head>

 <body>

   <h1>Hello, world!</h1>

   <button type="button" class="btn btn-warning">Warning</button>

   <!-- Option 1: Bootstrap Bundle with Popper -->

   <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bun
dle.min.js"
integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q
+8nbTov4+1p" crossorigin="anonymous"></script>

 </body>

</html>
```

resultat:

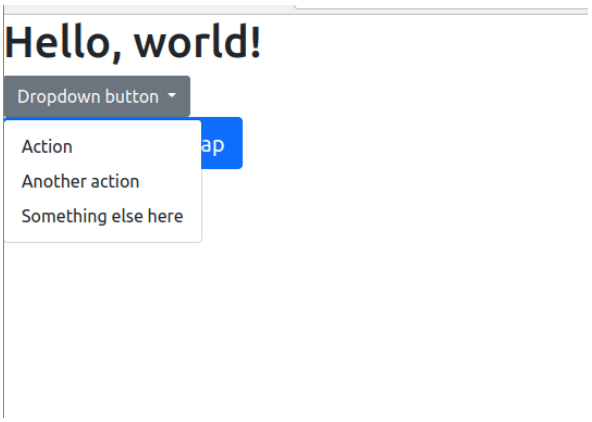

file:///home/sergi/Documents/bootstrap/1.html
Hello, world!
Warning

Bootstrap3.html

```
<!doctype html>

<html lang="en">

 <head>

   <!-- Required meta tags -->

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1">



   <!-- Bootstrap CSS -->

   <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.m
in.css" rel="stylesheet"
integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoq
yl2QvZ6jIW3" crossorigin="anonymous">



   <title>Hello, world!</title>

 </head>

 <body>

   <h1>Hello, world!</h1>



   <div class="dropdown">

 <button class="btn btn-secondary dropdown-toggle" type="button"
id="dropdownMenuButton1" data-bs-toggle="dropdown"
aria-expanded="false">

   Dropdown button

 </button>

 <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton1">
```

```
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
  </ul>
</div>
    <button type="button" class="btn btn-primary btn-lg">
                                            Iniciar amb Boostrap
                        </button>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

  </body>
</html>
```

Resultat:



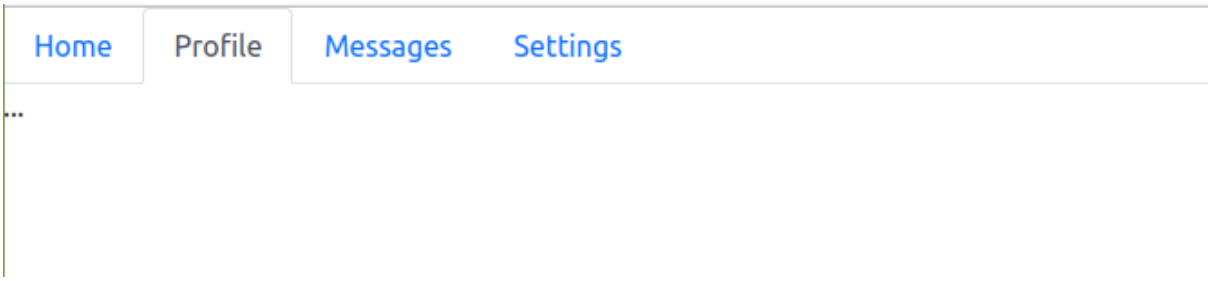
Bootstrap4.html

```
<!doctype html>

<html lang="en">

 <head>

   <!-- Required meta tags -->

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1">



   <!-- Bootstrap CSS -->

   <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.m
in.css" rel="stylesheet"
integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoq
yl2QvZ6jIW3" crossorigin="anonymous">



   <title>Hello, world!</title>

 </head>

 <body>

    <!-- Navegador -->

<!-- Nav tabs -->

<ul class="nav nav-tabs" id="myTab" role="tablist">

 <li class="nav-item" role="presentation">

   <button class="nav-link active" id="home-tab" data-bs-toggle="tab"
data-bs-target="#home" type="button" role="tab" aria-controls="home"
aria-selected="true">Home</button>

 </li>

 <li class="nav-item" role="presentation">
```

```
   <button class="nav-link" id="profile-tab" data-bs-toggle="tab"
data-bs-target="#profile" type="button" role="tab"
aria-controls="profile" aria-selected="false">Profile</button>

 </li>

 <li class="nav-item" role="presentation">

   <button class="nav-link" id="messages-tab" data-bs-toggle="tab"
data-bs-target="#messages" type="button" role="tab"
aria-controls="messages" aria-selected="false">Messages</button>

 </li>

 <li class="nav-item" role="presentation">

   <button class="nav-link" id="settings-tab" data-bs-toggle="tab"
data-bs-target="#settings" type="button" role="tab"
aria-controls="settings" aria-selected="false">Settings</button>

 </li>

</ul>



<!-- Tab panes -->

<div class="tab-content">

 <div class="tab-pane active" id="home" role="tabpanel"
aria-labelledby="home-tab">...</div>

 <div class="tab-pane" id="profile" role="tabpanel"
aria-labelledby="profile-tab">...</div>

 <div class="tab-pane" id="messages" role="tabpanel"
aria-labelledby="messages-tab">...</div>

 <div class="tab-pane" id="settings" role="tabpanel"
aria-labelledby="settings-tab">...</div>

</div>



   <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bun
```

```
dle.min.js"
integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q
+8nbTov4+1p" crossorigin="anonymous"></script>

>

 </body>

</html>
```

Resultat:

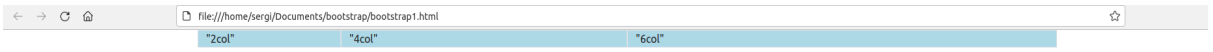


Bootstrap5.html

```
<!doctype html>

<html lang="en">

 <head>

   <!-- Required meta tags -->

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1">



   <!-- Bootstrap CSS -->

   <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.m
in.css" rel="stylesheet"
integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoq
yl2QvZ6jIW3" crossorigin="anonymous">



   <title>Hello, world!</title>

   <style type="text/css">.fondo {background: lightblue;};

        .borde{border:  1px, #000 solid;}

 </style>

 </head>

 <body>



   <div class="container fondo">

     <div class ="row">

         <div class="col-xs-12 col-sm-2 border">"2col"</div>

         <div class="col-xs-12 col-sm-4 border" >"4col"</div>
```

```
        <div class="col-xs-12 col-sm-6 border">"6col"</div>

    </div>

  </div>

 <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

  </body>

</html>
```

Sortida:



Javascript

El lenguaje JavaScript es muy directo y convencional a la hora de definir variables, funciones y operadores. La sintaxis es parecida a C, C++ y C#, e incluso un poco a PHP.

Cada instrucción se termina en “;”, y los caracteres “{“ y “}” se utilizan para delimitar el ámbito de las funciones y otros elementos como objetos JSON (los veremos más adelante).

```
< script >

        function muestrasuma(numero1, numero2) {

                var resultado;

                resultado = numero1 + numero2;

                alert(resultado);

        }


        muestrasuma(10, 2);
```

Las variables en JavaScript no tienen un tipo determinado, por lo que habrá que tener mucho cuidado a la hora de mezclar diferentes tipos de datos en las operaciones.

13.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Titol del document</title>
        <meta charset="utf8">
        <link href="13.css" rel="stylesheet" type="text/css">

        <!-- com incloure js al nostre document?-->
        <!-- opcio 1: amb un fitxer extern-->
        <script src="13.js" type="text/javascript"></script>

        <!-- opcio 2: com un conjunt de funcions directament al head
-->
        <script>
            function mostra (valor){
                alert(valor);
            }
        </script>

    </head>
    <body>
        <header>
```

```
            <h1 id="tit">Titol principal</h1>

            <h2>Subtitle</h2>


            <nav id="menu-principal">

                <li><a href="#">Inici</a></li>

                <li><a href="#">Productes</a></li>

                <li><a href="#">Serveis</a></li>

                <li><a href="#">Contacte</a></li>

            </nav>


        </header>



        <section>

            <article class="casalloguer">

                <h1>Casa lloguer 1</h1>

                <!-- incluim codi javascript-->

                <!-- opció 3: directament al codi on volem fer-lo
correr-->

                <a href="#" onclick="alert('informacio
ampliada');">Ampliar informació</a>

            </article>

            <article class="casalloguer">

                <h1>Casa lloguer 2</h1>

                <!-- aquí cridem a una funció que hem definit abans-->
```

```
                <a href="#" onclick="mostra('hola pepe');">Ampliar
informació</a>


            </article>
            <article class="casalloguer">
                <h1>Casa lloguer 3</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article  class="casalloguer">
                <h1>Casa lloguer 4</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article class="casalloguer">
                <h1>Casa lloguer 5</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article class="casalloguer">
                <h1>Casa lloguer 6</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article class="casalloguer">
                <h1>Casa lloguer 7</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article  class="casalloguer">
```

```
                <h1>Casa lloguer 8</h1>
                <a href="#">Ampliar informació</a>
            </article>
        </section>



        <footer>
            &copy;2022 EL meu lloguer
        </footer>

    </body>
</html>
```

13.css

```
a {
    color:rgb(18, 32, 229);
}



#tit {
    color: rgb(255, 0, 102);
}



.casalloguer {

    color: white;
    border: 2px solid black;
    margin: 10px 10px;
    padding: 10px 10px;
     /* atenció amb inline-block */
    display: inline-block;
    width: 30%;
    height: 200px;
     /* noves funcions */
    opacity: 0.6;
    border-radius: 20px;
```

```
    box-shadow: 3px 3px 10px rgb(109, 58, 58);
    background: #ffc578; /* Old browsers */



}


#menu-principal{
    background-color: rgb(194, 192, 255);
    color: blue;
    border: 2px solid rgb(37, 169, 240);
    margin: 20px;
    padding: 20px;
}
ul, li {
    padding: 0,0;
    margin: 2px ,2px;
    display: inline-block;
}
/* footer  */
footer {
    background-color: rgb(7, 218, 218);
    padding: 10px, 10px;
    margin: 0;


}
```

13.js

```
alert ("hola mon");
```

Variables

14.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Titol del document</title>
        <meta charset="utf8">
        <link href="14.css" rel="stylesheet" type="text/css">

    </head>
    <body>
        <header>
            <h1 id="tit">Titol principal</h1>
            <h2>Subtitle</h2>

            <nav id="menu-principal">
                <li><a href="#">Inici</a></li>
                <li><a href="#">Productes</a></li>
                <li><a href="#">Serveis</a></li>
                <li><a href="#">Contacte</a></li>
            </nav>

        </header>
```

```
        <script>
            function mostra(x){
                alert(x);
            }

            a= "resultat";
            b=": es... ";
            c= 7*9;
            /* es poden sumar cadenes de text i numeros */
            mostra(a+b+c);
            //això també és un comentari

            function dividir (a,b) {
                //alerta amb l'ambit de les variables
                //si definim var result --> variable local
                //si directament fem result =   --> va a buscar una
variable global
                var result=(a/b);
                return result;
            }

            mostra (dividir(10,2));
```

```
</script>



<section>
    <article class="casalloguer">
        <h1>Casa lloguer 1</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 2</h1>
         <a href="#">Ampliar informació</a>

    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 3</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article  class="casalloguer">
        <h1>Casa lloguer 4</h1>
        <a href="#">Ampliar informació</a>
    </article>
    <article class="casalloguer">
        <h1>Casa lloguer 5</h1>
        <a href="#">Ampliar informació</a>
    </article>
```

```
            <article class="casalloguer">
                <h1>Casa lloguer 6</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article class="casalloguer">
                <h1>Casa lloguer 7</h1>
                <a href="#">Ampliar informació</a>
            </article>
            <article  class="casalloguer">
                <h1>Casa lloguer 8</h1>
                <a href="#">Ampliar informació</a>
            </article>
        </section>


        <footer>
            &copy;2022 EL meu lloguer
        </footer>

    </body>
</html>
```

14.css

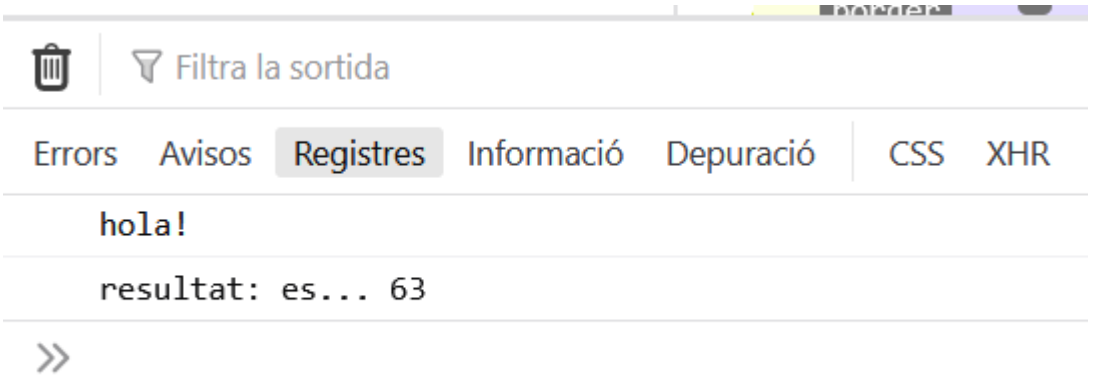
```
a {
    color:rgb(18, 32, 229);
}

#tit {
    color: rgb(255, 0, 102);
}

.casalloguer {
    color: white;
    border: 2px solid black;
    margin: 10px 10px;
    padding: 10px 10px;
    display: inline-block;
    width: 30%;
    height: 200px;
    opacity: 0.6;
    border-radius: 20px;
    box-shadow: 3px 3px 10px rgb(109, 58, 58);
    background: #ffc578; /* Old browsers */
}

#menu-principal{
    background-color: rgb(194, 192, 255);
```

```
    color: blue;
    border: 2px solid rgb(37, 169, 240);
    margin: 20px;
    padding: 20px;
}

ul, li {
    padding: 0,0;
    margin: 2px ,2px;
    display: inline-block;
}

/* footer  */
footer {
    background-color: rgb(7, 218, 218);
    padding: 10px, 10px;
    margin: 0;
}
```

Console log

Envia dades a la consola:

```
a= "resultat";

          b=": es... ";

          c= 7*9;

          console.log("hola!");

          console.log(a+b+c);
```



Las cadenas de texto en JavaScript son objetos complejos, que incorporan toda una serie de métodos para manipularlas. La concatenación de cadenas se realiza con el operador “+”.

Las cadenas pueden estar definidas utilizando comillas simples o dobles, siempre que se utilice el mismo tipo de comillas para abrir y cerrar la cadena.

< script >

var a = "Hola";

var b = "Mundo";

alert(a + " " + b);

< /script >

Condicionals:

```
if (n <50) {
        alert(“n menor que 50);
        }
else    {
        alert (“n major o igual que 100”);
        }
```

FOR: (com el de C)

```
for (var i=0; i<=5; i++) {
        console.log(i);
 }
```

WHILE

```
var i=0;
while (i<10) {
        console.log (i);
        i++;
        }
```

EL DOM (document object model)

En JavaScript ejecutado desde un documento HTML, contamos con toda una serie de objetos y sus métodos predefinidos que representan todos los objetos visuales del documento, de manera que podamos manipularlos con facilidad. A esto se le llama Document Object Model (DOM).

El objeto “document” que representa a todo el documento cuenta con un método getElementById, que devuelve un objeto especificando su atributo id, para que luego podamos manipularlo.

< a href="#" id="**mienlace**" style="background-color: orange;">Enlace demo< /a >
< script >
document.getElementById("**mienlace**").style.backgroundColor="orange";
< /script >

JavaScript es un lenguaje que además de ser funcional (permite la declaración directa de funciones) también es “basado en objetos”.

Para aquellos que tengan nociones de orientación a objetos, decir que la diferencia en ser “basado en objetos”, es que carecemos de mecanismos sencillos en el lenguaje para definir clases y herencias, y exigir comprobaciones de pertenencia a dichas clases.

Si todo esto no te dice nada, no te preocupes. Lo importante es que en JavaScript en el navegador el DOM nos proporciona toda una serie de objetos equivalente a todos los

elementos de la página web, que están atribuidos con sus respectivas propiedades equivalente según se ven en cada momento en la página.

Gracias al **DOM** cualquier cambio que hagamos en estos objetos se traduzca automáticamente en un cambio en la representación visual del mismo en la web.

Pero, ¿cómo se utiliza esto para programar? ¿cómo se acceden a estos objetos y métodos? En primer lugar, partimos de un objeto principal llamado document, que contiene a todo el documento HTML.

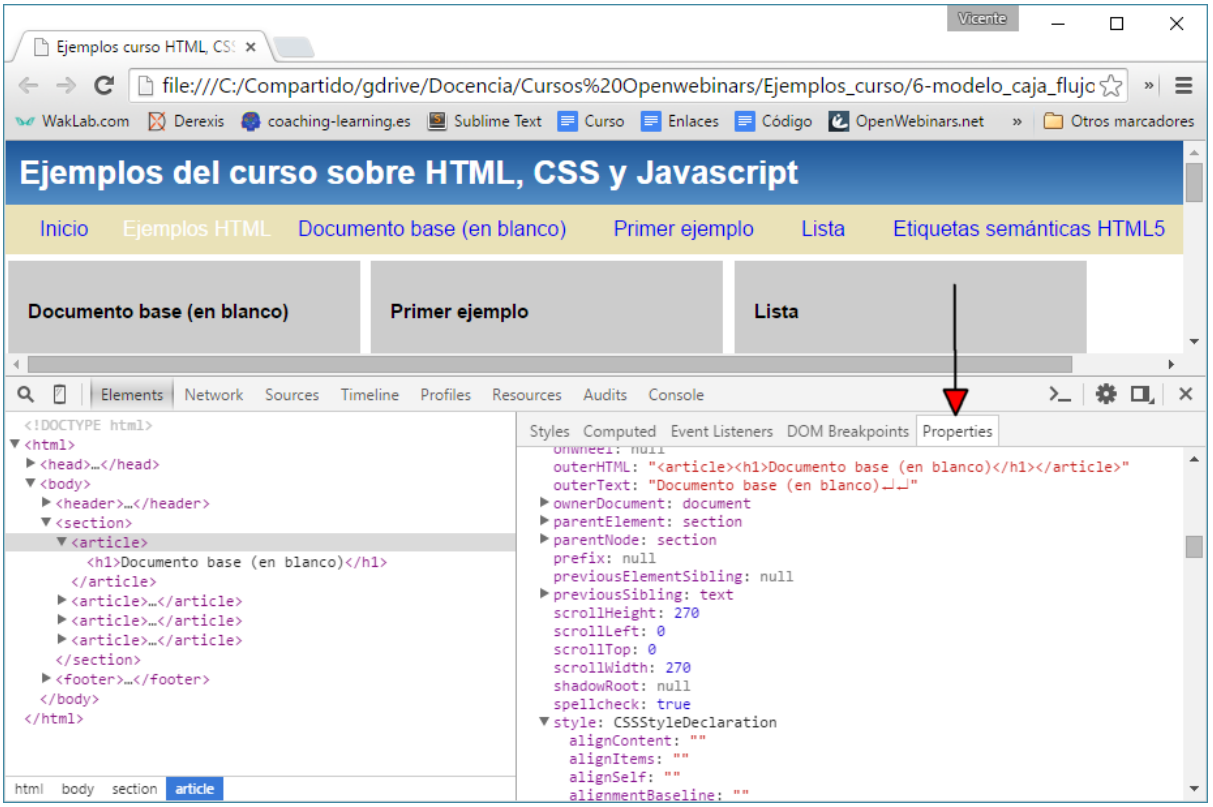
Éste tiene un método **getElementByID** que nos permite obtener un objeto concreto del documento, sabiendo su identificador. A partir de ahí, podríamos por ejemplo acceder a la propiedad **style** que define todos los estilos CSS del objeto, y de nuevo a la propiedad **border**, que permite definir un nuevo estilo para el borde. En conjunto, todo esto se hace con la siguiente linea de código:

```
document.getElementById('menu-principal').style.border = '1px solid red';
```

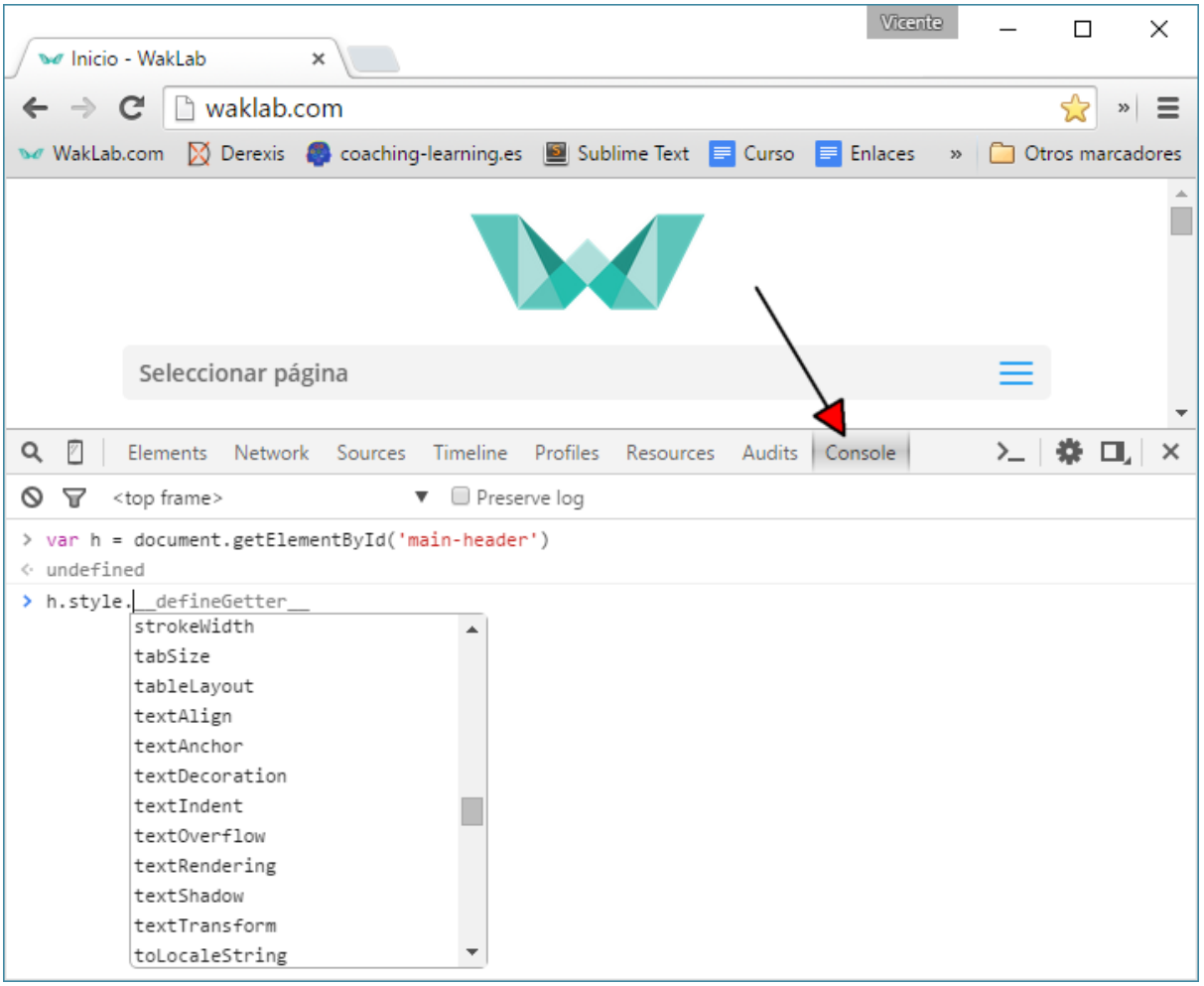
Un truco muy común es cambiar la propiedad CSS display de un objeto de block para que sea visible, a none para que quede oculto. Esto lo podríamos conseguir así:

```
document.getElementById('menu-principal').style.display = 'none';
```

La consola de código de Chrome es muy útil a la hora de explorar las propiedades JavaScript que tiene un objeto, para ello habrá que seleccionar el nodo que queremos inspeccionar, y pulsar en la pestaña de la derecha llamada **Properties**.



Otro truco útil es **utilizar la consola JavaScript** del inspector de código. Cuando en ella escribimos código JavaScript, se ejecuta inmediatamente sobre la página actual. Para los diferentes objetos del DOM, si los almacenamos en una variable nos ofrece además completación inteligente del código.



Unas buenas referencias al DOM las podéis encontrar en las siguientes webs:

- https://developer.mozilla.org/es/docs/DOM http://www.w3.org/DOM/DOMTR http://www.w3schools.com/js/js_htmldom.asp

Métodos para escribir con JS

Usamos Javascript para tratar de hacer las páginas más ligeras y dinámicas además de lograr que se muestren correctamente en cualquier dispositivo.

En JavaScript existen varios métodos que nos permiten escribir y mostrar texto u otro contenido en una página web, éstos son:

- document.write

- document.writeln
- innerHTML
- textContent

Los caracteres de escape funcionan siempre que estén dentro de **elementos pre**. Podemos utilizar carácteres de escape normalmente si empleamos document.write, document.writeln y **template strings** en ES6.

Una ventana alert() tiene formato pre por defecto.

Es decir, si utilizamos tabuladores u otros signos especiales, antes debemos encerrar la salida entre etiquetas pre.

las plantillas de Texto: Template Strings en ES6

Con el nuevo estándar ES6 se han introducido las Plantillas de Texto de forma nativa permitiendo con ello cubrir, al menos parcialmente, una necesidad habitual del desarrollo moderno.

Las plantillas de texto (o Template Strings) son cadenas literales de texto incrustadas en el código fuente que permiten su interpolación mediante expresiones. Históricamente, el nombre que ha recibido esta estructura es la de Quasi Litarals (‘casi literales’).

Su sintaxis es sencilla y consta de dos partes:

- Se utilizan comillas invertidas, es decir, **acentos graves** ` para delimitar las cadenas.
- Para añadir expresiones (variables, sentencias,...) utilizamos el signo de dolar $ seguido de llaves { } entre las que colocamos la expresión.

```
let miTexto = 'En un lugar de la Mancha, ...';
console.log(`Mira esta frase: ${miTexto}`);
//→ Mira esta frase: En un lugar de la Mancha
```

Gracias a las plantillas, ahora es posible disfrutar de cadenas multilínea en Javascript sin necesidad de recurrir a concatenaciones, barras invertidas, unión de arrays u otros malabares habituales.

Es importante recalcar que las plantillas mantienen el formato introducido, incluyéndose los saltos de línea y las tabulaciones, es decir, como el texto que encerramos con el elemento pre.

El método document.write

El método document.write de Javascript permite escribir en una página texto, el resultado de una funcion o las dos cosas. El resultado aparecerá en el lugar exacto de la página donde se inserte el código. El método document.writeln es semejante salvo que inserta un saldo de línea al final (si trabajamos con etiquetas pre).

Sobra decir que este código debe ir encerrado entre dos etiquetas script para que el navegador lo interprete como tal (si lo colocamos en el propio archivo HTML). El texto

encerrado entre comillas será escrito literalmente. Si no se usan comillas se asumirá que es una variable.

La mayoría de las veces se utiliza Javascript para escribir texto cuando está acompañado de una variable que cambiará su valor dependiendo de lo que se exprese con ella.

Dentro del texto que queremos mostrar podemos incluir código HTML.

Ejemplos:

```
<script>
// Ejemplo 1
document.write("Bienvenido a mi sitio web. Gracias...");
// Ejemplo 2
document.write('<a href=\"http://3con14.pro\">Enlace a una página</a>');
// Ejemplo 3
document.write(new Date());
// Ejemplo 4
document.write('Ancho de pantalla: '+ screen.width +' pixeles');
// Ejemplo 5
document.write('Ultima modificacion: '+ window.document.lastModified);
// Ejemplo 6
document.write('Usando el navegador: '+ navigator.appName);
</script>
// Ejemplo 7
window.onload = document.write('<img src=\"grande.jpg\" alt=\"miImagen\" />');
```

La propiedad textContent

Escribir un texto plano (sin formato HTML) dentro de un determinado elemento HTML.

La propiedad textContent establece o devuelve el contenido de texto del nodo especificado y todos sus descendientes. Si establecemos la propiedad textContent, los nodos secundarios se eliminan y reemplazan por un solo nodo de texto que contiene la cadena especificada.

- **textContent** devuelve el contenido de texto de todos los elementos, mientras que innerText devuelve el contenido de todos los elementos, a excepción de los elementos script y style.
- **innerText** no devolverá el texto de los elementos que están ocultos con CSS (textContent lo hará).

Para establecer o devolver el contenido HTML de un elemento, usamos la propiedad innerHTML.

La propiedad: innerHTML

La usamos para agregar dinamismo a la página, es decir, para agregar texto después de que esté cargada. Mediante ella podemos agregar contenido dentro de un elemento, sin modificar el resto del contenido.

Forma de utilizarla:

1. Colocamos un contenedor en la página HTML, normalmente un div con su id para apuntar a él fácilmente desde JS.
2. Desde JS, se inserta el 'texto' que deseemos en dicho contenedor.
3. Dentro del 'texto' que queremos mostrar podemos incluir código HTML.

Ejemplos:

```
<!-- Un ejemplo -->
document.getElementById("demo").innerHTML = "Esto es un párrafo de ejemplo";
<!-- Otro ejempplo -->
<p id="parrafo1">Esto es el texto del párrafo 1</p>
<button onclick="mostrar()"> Pulsa </button>
<p id="parrafo2"></p>
<script>
function mostrar() {
   var x = document.getElementById("parrafo1").innerHTML;
   document.getElementById("parrafo2").innerHTML = x;
}
</script>
```

Algunos caracteres es necesario escribirlos tal y como son y no lo que representan, por eso es necesario anteponer la barra invertida (\) también conocida como signo de escape.

Por ejemplo, para escribir "Z:\tareas" tal cual, dentro un contenedor con id="respuesta" escribimos:

```
respuesta.innerHTML = '\"Z:\\tareas\"';
```

El código JavaScript puede usarse como respuesta a un evento, por ejemplo hacer clic en un botón:

```
<button onclick="EncenderCrono();">Encender</button>
```

16 HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Titol del document</title>
        <meta charset="utf8">
        <link href="15.css" rel="stylesheet" type="text/css">

    </head>
    <body>
        <header>
            <h1 id="tit">Titol principal</h1>
            <h2>Subtitle</h2>

            <nav id="menu-principal">
                <li><a href="#">Inici</a></li>
                <li><a href="#">Productes</a></li>
                <li><a href="#">Serveis</a></li>
                <li><a href="#">Contacte</a></li>
            </nav>

        </header>


        <script>
           function ampliar_info(){
                var ampliat = document.getElementById('titol');
                ampliat.style.display='none';

                var article1 = document.getElementById('article1');
                var pTexto = document.createTextNode("Aqui hi ha
informacio ampliada del exemple");
                article1.appendChild(pTexto);

                //també si no volem crear un fill
                var article2 =document.getElementById('article2');
                article2.innerHTML="<p> HOLA!  </p>";

           }

        </script>
```

```
        <section>
            <article id="article1" class="casalloguer">
                <h1 id="titol">Casa lloguer 1</h1>
                <a href="#" onclick="javascript:
ampliar_info();">Ampliar informació</a>
            </article>
            <article id="article2" class="casalloguer">
                <h1>Casa lloguer 2</h1>
                 <a href="#">Ampliar informació</a>

            </article>
                   </section>


        <footer>
            &copy;2022 EL meu lloguer
        </footer>

    </body>
</html>
```