HTML CSS Y Librerias (framework CSS)

IS093 – DESARROLLO DE APLICACIONES WEB

JAIME SUASNABAR TERREL

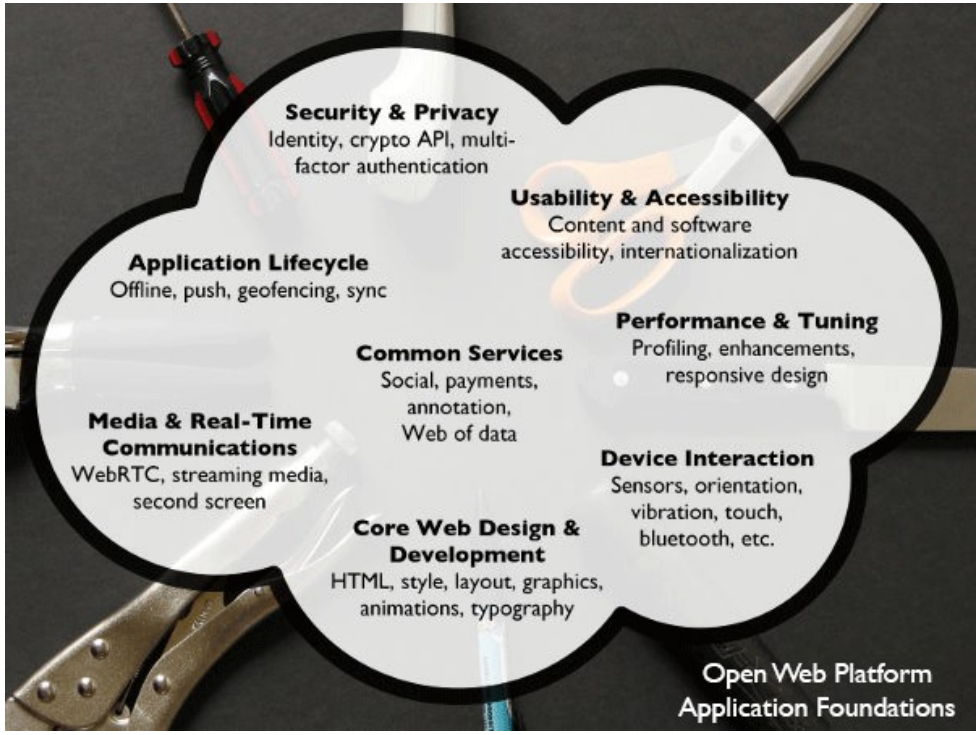
Temas Principales

- Open Web Platform
- HTML
  - ¿Cómo funciona HTML?
  - Principales etiquetas del Lenguaje HTML 5
  - Árbol DOM
  - Ciclo de Vida de una Página Web
- CSS
  - ¿Qué es CSS?
  - ¿Dónde y cómo utilizar CSS?
  - Selectores
    - De tipo
    - De clase
    - De id
    - Selector universal
    - Selector de atributo
    - Combinador de hermanos adyacentes
    - Combinador General de hermanos
    - Combinador de hijo
    - Combinador de descendientes
- Unidades de longitud
  - Unidades absolutas
  - Unidades relativas
- Layout
  - Flexbox / Grid
- Posicionamiento
  - Static / Relative / Absolute
- Diseño fluido
- Diseño responsivo
- Accesibilidad
  - Soluciones tecnológicas AT
    - Visuales
    - Auditivas
    - Motrices
    - Cognitivas
  - Normativas y estándares
    - WCAG 2.1

Open Web Platform - Web Abierta

¿Qué es?
Open Web Platform es una colección de tecnologías abiertas y estándares desarrollados por organismos como W3C, Unicode Consortium, Internet Engineering Task Force, y ECMA International. Algunas de las tecnologías que cubre son HTML5, CSS, SVG, MathML, WAI-ARIA, ECMAScript, WebGL, etc..

- PRINCIPIOS:
- **Evita ser controlado** por ninguna empresa.
- **Aporta transparencia**, haciendo visibles todos los niveles desde el origen de documento hasta las URLs y las capas HTTP.
- **Capacidad de integración**. Fuentes externas.
- **Documentación y especificaciones abiertas**.
- **Libertad de Uso**.
- **Cadena de favores**. Ser integrante de la Open Web significa compartir lo aprendido con blogs, conferencias o el uso de tecnologías abiertas.



HTML HyperText Markup Language

El HTML, es un lenguaje de marcado, *define la semántica y estructura al contenido* de las páginas web. Se basa en ETIQUETAS “<“ “>”

- Se guarda en *archivos con extensión* .html o .htm y se ve a través de cualquier navegador.
- Es parte del estándar SGML (Standard Generalized Markup Language o lenguaje de marcado generalizado estándar)

HISTORIA

El HTML fue inventado por **Tim Berners-Lee** (físico del CERN) en 1989. Se le ocurrió la idea de un sistema de hipertexto (enlaces a contenido) en Internet.

- Necesitaba crear documentos con referencias a otros para facilitar el acceso y la colaboración de otros equipos.

Estructura de Una Página Web

- Toda página web tiene la siguiente estructura:
- DOM Document Object Model

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Estructura</title>
</head>
<body>
    <h1>Titulo</h1>
    <p>parrafo</p>
</body>
</html>
```

¿Cómo funciona HTML?

Un conjunto de *etiquetas*

Tipo de documento

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Estructura</title>
</head>
<body>
    <div>
        <header></header>
        <nav></nav>
        <section>
            <h1>Titulo</h1>
            <p>parrafo</p>
        </section>
        <footer></footer>
    </div>
</body>
</html>
```

Metadata de la página

Contenido de la página

Tags apertura y cierre

Jerarquía de etiquetas:
- Bloques de contenido
- Encabezados
- Párrafos

Etiquetas VOID
Solo tienen Tag de apertura
<hr/>

Principales etiquetas del Lenguaje HTML 5

Elemento	Descripcion
<!doctype html>	Define que el documento esta bajo el estandar de HTML 5
<html>	Representa la raíz de un documento HTML o XHTML. Todos los demás elementos deben ser descendientes de este elemento.

Elemento	Descripción
<head>	Contiene metadatos de la página, incluyendo definiciones de, scripts y hojas de estilo.
<title>	Define el título del documento, el cual se muestra en la barra de título del navegador o en las pestañas de página.
<base>	Define la URL base para las URLs relativas en la página.
<link>	Usada para enlazar CSS externos.
<meta>	Define los metadatos de la Página
<style>	Etiqueta de estilo usada para escribir CSS en línea.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Estructura</title>
</head>
<body>
    <div>
        <header></header>
        <nav></nav>
        <section>
            <h1>Titulo</h1>
            <p>parrafo</p>
        </section>
        <footer></footer>
    </div>
</body>
</html>
```

Principales etiquetas del Lenguaje HTML 5

Elemento	Descripción
<body>	Representa el contenido principal de un documento HTML.
<section>	Define una sección en un documento.
<nav>	Contiene enlaces de navegación
<article>	Define contenido autónomo que podría existir independiente del resto del contenido.
<aside>	Define algunos contenidos vagamente relacionados con el resto del contenido de la página. Si es removido, el contenido restante seguirá teniendo sentido
<h1>,<h2>,<h3>, <h4>,<h5>,<h6>	Titulos de cabecera
<header>	Define la cabecera de una página o sección. Usualmente contiene un logotipo, el título del sitio Web y una tabla de navegación.
<footer>	Define el pie de una página o sección. Usualmente contiene un mensaje de derechos de autoría, algunos enlaces a información legal o direcciones para dar información de contacto.
<address>	Define una sección que contiene información de contacto.
<main>	Define el contenido principal o importante en el documento. Solamente existe un elemento <main> en el documento.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Estructura</title>
</head>
<body>
    <div>
        <header></header>
        <nav></nav>
        <section>
            <h1>Titulo</h1>
            <p>parrafo</p>
        </section>
        <footer></footer>
    </div>
</body>
</html>
```

XML Lenguaje Marcado eXtensible

- XML o Lenguaje de Marcado Extensible es **un estándar que se utiliza para almacenar e intercambiar información estructurada entre diferentes plataformas. Es un** documento que contiene configuraciones, transacciones o simplemente datos.
- Es un lenguaje de marcado que define la estructura y el significado de los datos.
- Un documento XML se divide en dos partes: *prolog* y *body*.
  - La parte *prolog* consiste en metadatos administrativos, como declaración XML, instrucción de procesamiento opcional, declaración de tipo de documento y comentarios.
  - La parte del *body* se compone de dos partes: estructural y de contenido (presente en los textos simples).

Árbol DOM

¿Qué es?
DOM (Document Object Model) **es una interfaz para documentos HTML y XML** que se representa como un árbol de elementos. ***Permite leer y manipular el contenido, la estructura y los estilos de la página*** con un lenguaje de scripting como JavaScript.

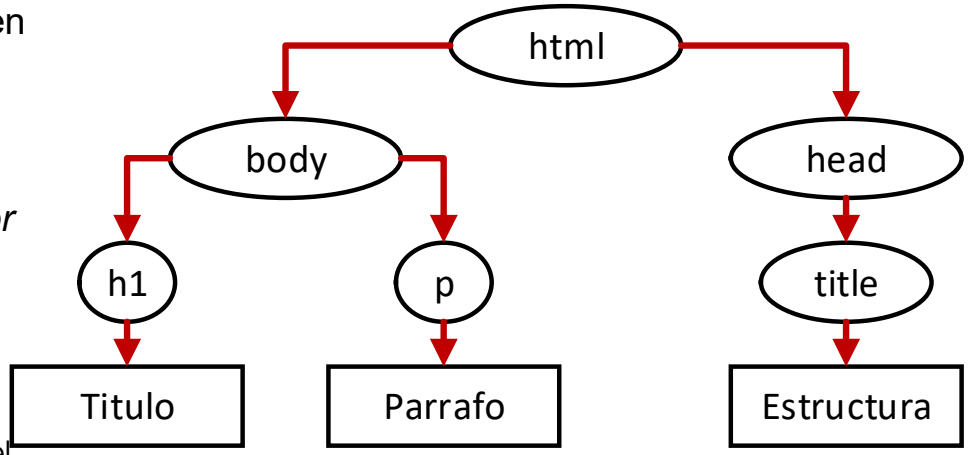
RENDER TREE

- La forma de un documento ***HTML***, muestra estilos e interactividad, se denomina *Critical Rendering Path*. Primero se establece que se va a renderizar y se denomina ***render tree*** (DOM + CSSOM). Luego, el navegador realiza el renderizado.
  - CSSOM: representa los estilos asociados a los elementos.
  - DOM: representa los elementos.
- El *render tree* excluye los elementos que no están visibles como por ejemplo, los que tienen el estilo display: none. El DOM si lo incluiría en su árbol de nodos.

ÁRBOL DE NODOS

- ***El objetivo del DOM*** *es convertir la estructura y el contenido del documento HTML en un modelo de objeto que puede ser utilizado por varios programas*. La estructura del documento es conocida como un árbol de nodos (node tree). El elemento raíz es la etiqueta html, las ramas son los elementos anidados y las hojas serían el contenido de esos elementos.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Estructura</title>
</head>
<body>
    <h1>Titulo</h1>
    <p>parrafo</p>
</body>
</html>
```



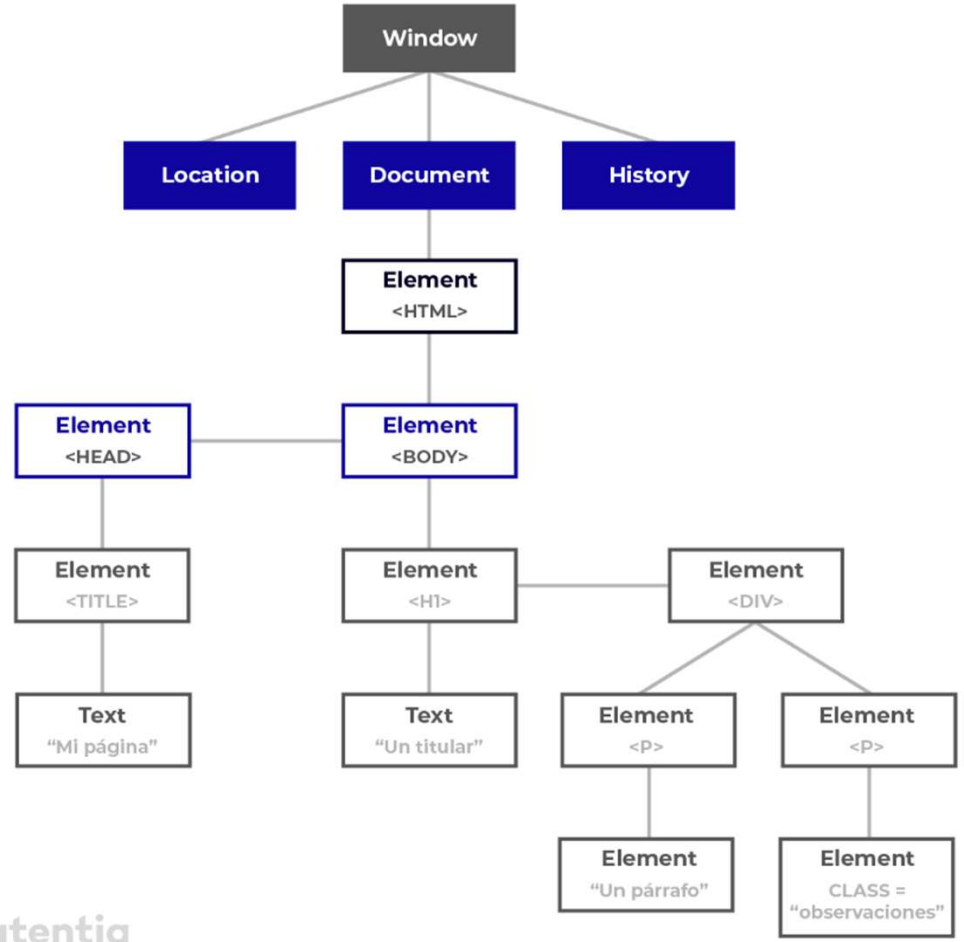
BOM Browser Object Model

El BOM es el api de la ventana gráfica de visualización del navegador tiene estructura jeráquica tipo árbol cuya raíz es el *objeto global window*.

- El objeto Window implementa la interfaz de comunicación con la ventana del navegador. Esta interfaz conecta al navegador con Javascript.

```
window.document.
window.storages.
window.screen.
window.history.
window.location.
```

- Este objeto posee todos los métodos, propiedades y eventos que maneja javascript. Bien sea directamente o a través de sus nodos.



¿Cómo manipular el DOM?

- DOM (Document Object Model) es la interfaz que permite que lenguajes de programación como JavaScript, modifiquen la estructura, el estilo o el contenido de la página web.
  - A través de la etiqueta <script>, se puede comenzar a manipular el documento o los elementos de la página.
  - El DOM nos ofrece una multitud de propiedades y métodos para acceder a los diferentes elementos y poder modificarlos.

```
document.getElementById(“IdName");

document.getElementsByTagName("span");

document.getElementsByClassName(“className");

document.querySelector(".container");
```

```
> document.getElementById("a");
<-   <div id="a">A</div>
> document.querySelector('#a');
<-   <div id="a">A</div>
> a
<-   <div id="a">A</div>
>
```

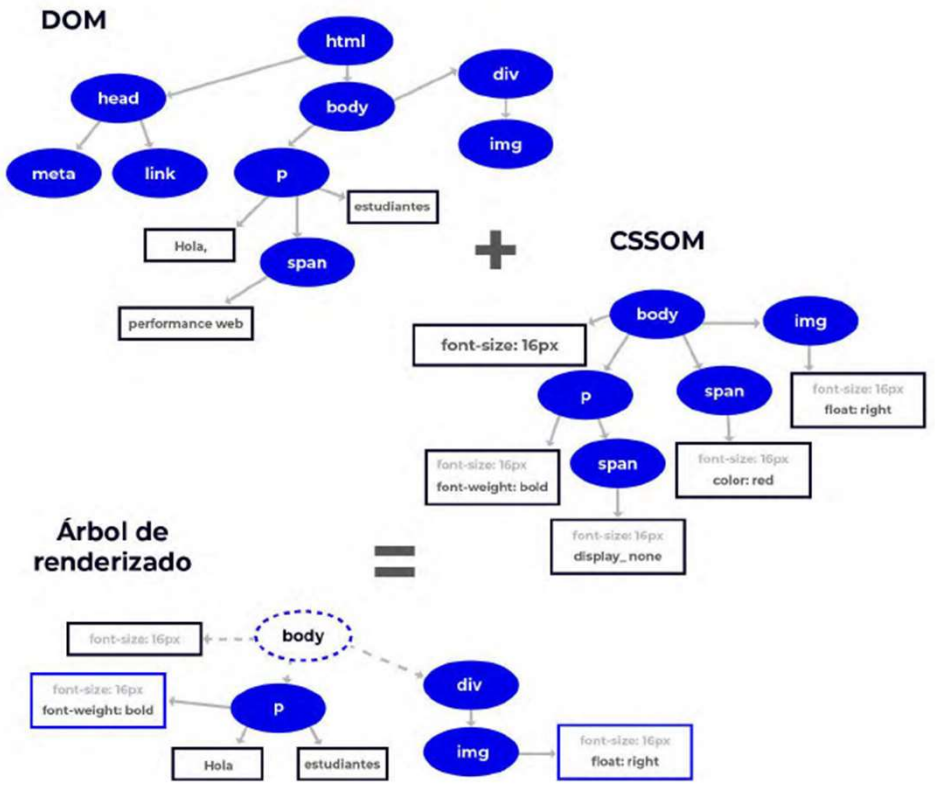
Ciclo de Vida de una Página Web

¿Qué es?
También referido como ***Critical Rendering Path*** (ruta de renderizado crítica) es la **secuencia de pasos** que sigue el navegador **para convertir HTML, CSS y JavaScript en píxeles en la pantalla**.
Esta secuencia de pasos es realizada por el **motor de renderizado** del navegador.

¿EN QUÉ CONSISTE?

Una solicitud de una página web o aplicación comienza con una petición HTML. Al realizar una solicitud, el servidor devuelve los encabezados y datos de respuesta..

1. Se crea el Document Object Model (DOM) a partir de la respuesta HTML. También inicia solicitudes cada vez que encuentra enlaces a recursos externos, ya sean hojas de estilo, scripts o referencias de imágenes incrustadas.
   - Algunas solicitudes son bloqueantes, lo que significa que el parseo del resto del HTML se detiene hasta que se cargue el recurso.
2. Se crea el CSS Object Model (CSSOM).
3. Cuando tiene el DOM y el CSSOM listos, se combinan en el árbol de renderizado (Render Tree), obteniendo los estilos para todo el contenido visible.
4. Una vez que se completa el árbol de procesamiento, el diseño calcula la posición y el tamaño exactos de cada objeto (layout) del árbol de procesamiento.
5. Una vez completado, se procede a la representación o “pintado”, que consiste en tomar el árbol de renderizado final y renderizar los píxeles en la pantalla.



Ciclo de Vida de una Página Web

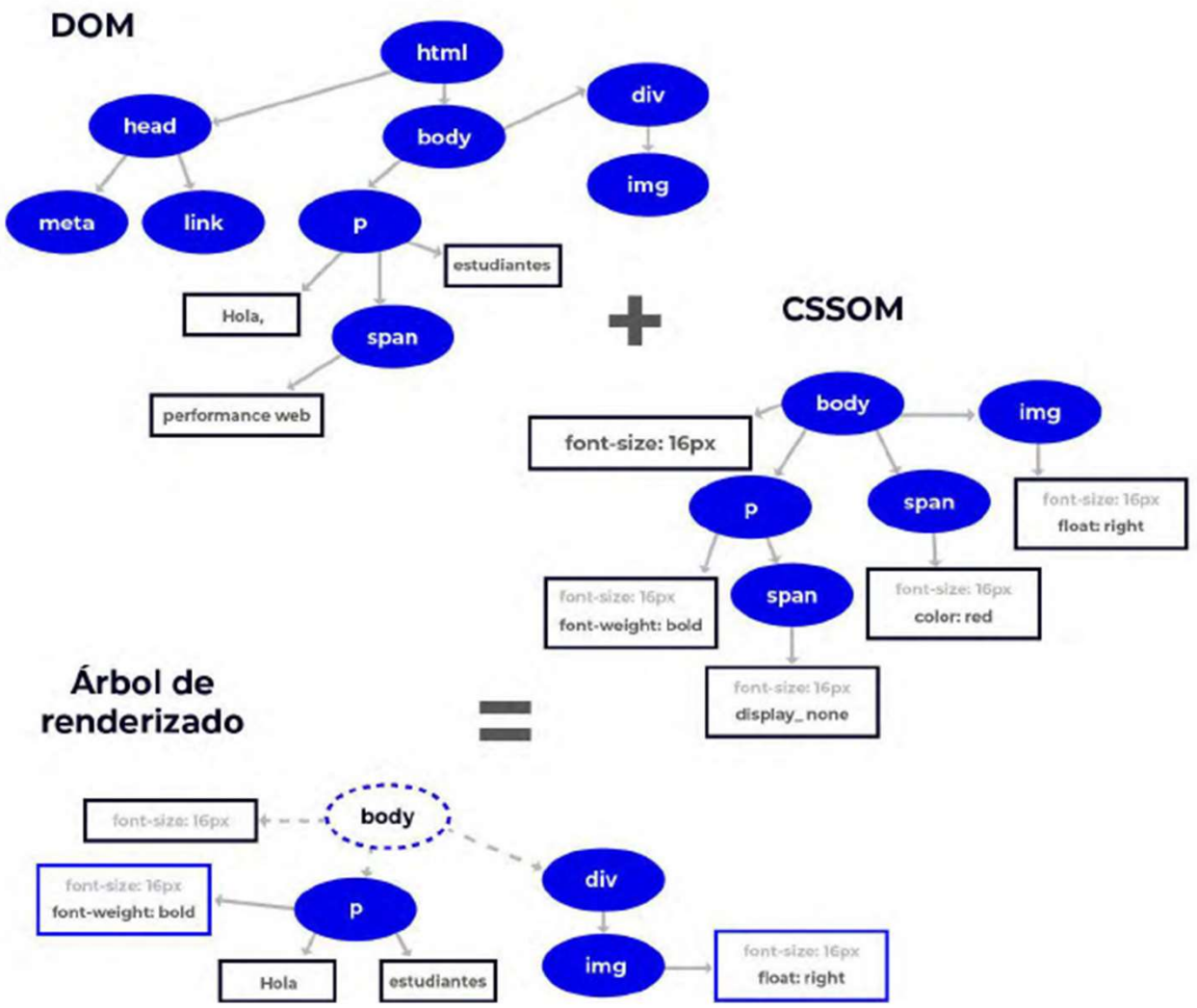
¿Qué es?

También referido como ***Critical Rendering Path*** (ru
navegador **para convertir HTML, CSS y JavaScrip**
Esta secuencia de pasos es realizada por el **motor**

¿EN QUÉ CONSISTE?

Una solicitud de una página web o aplicación comienza con u
HTML. Al realizar una solicitud, el servidor devuelve los enca
de respuesta..

1. Se crea el Document Object Model (DOM) a partir de la
   También inicia solicitudes cada vez que encuentra enlac
   externos, ya sean hojas de estilo, scripts o referencias d
   incrustadas.
   - Algunas solicitudes son bloqueantes, lo que significa
     del resto del HTML se detiene hasta que se cargue e
2. Se crea el CSS Object Model (CSSOM).
3. Cuando tiene el DOM y el CSSOM listos, se combinan e
   renderizado (Render Tree), obteniendo los estilos para t
   visible.
4. Una vez que se completa el árbol de procesamiento, el c
   posición y el tamaño exactos de cada objeto (layout) del
   procesamiento.
5. Una vez completado, se procede a la representación o "
   consiste en tomar el árbol de renderizado final y renderiz
   la pantalla.



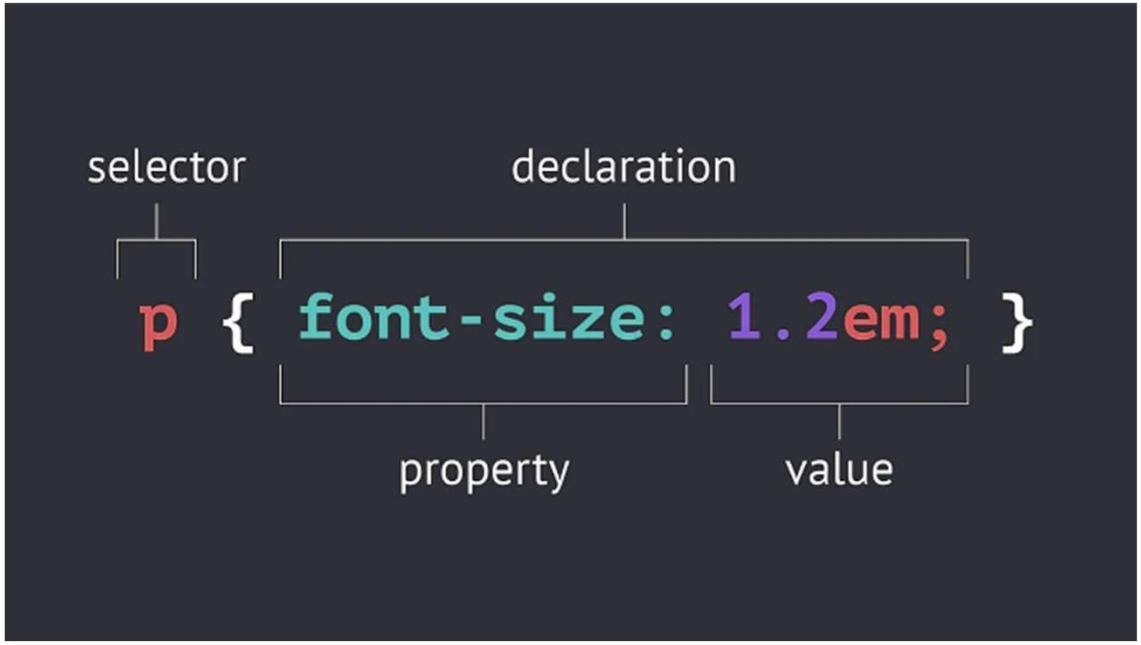
Recursos de Aprendizaje

- Para un aprendizaje más detallado de HTML recomendamos la siguiente dirección:
  - https://developer.mozilla.org/es/docs/Learn/HTML/Introduccion_a_HTML/iniciar

- Una guía de referencia de HTML puedes encontrar en:
  - https://developer.mozilla.org/en-US/docs/Web/HTML/Reference

Hojas de Estilo

- CSS, Cascading Style Sheets, Hojas de Estilo en Cascada Es un lenguaje utilizado para la presentación (diseño o aspecto visual) del contenido de una página web. Utilizando CSS daremos estilo a cualquier tag de HTML.



- USO DE ESTILOS
- Estilos en una misma página
  - En <HEAD> agregar la etiqueta STYLE.

```
<style type="text/css">
    Lista de estilos
</style>
```

- Estilos en un archivo CSS
  - La etiqueta LINK, permite indicar el archivo externo que contendrá los estilos. Su sintaxis es:

```
<link rel="stylesheet"
HREF="miweb.css" TYPE="text/css">
```

Estilos con CLASS

- Permite definir estilos reutilizables.
- DEFINIR LA CLASE

```
p.destacado {color: red; background:yellow;}
*.diminuto {font-size: 8px;}
.grande {font-size: 12px;}
```

- USAR LA CLASE
- En un tag usar el atributo CLASS al HTML, así.

```
<p class="destacado">Un párrafo destacado</P>
<p class="diminuto">Un párrafo diminuto</P>
<p class="grande">Un párrafo grande</P>
<p class="destacado">Otro destacado</P>
```

Estilos con ID

- Permite definir estilos no reutilizables (idealmente una vez). Utilizar el símbolo numeral (#) y un identificador (que corresponde al valor del atributo "id").

```
P#parrafo1 {color: green; background:yellow;}
*#parrafo2 {color: blue; background:yellow;}
#parrafo3 {color: red; background:yellow;}
```

- A continuación, podemos otorgar a algún elemento de HTML dicha identidad mediante el uso del atributo ID:

```
<P>Un párrafo normal</P>
<P ID=“parrafo1”>Parrafo1</P>
<P ID=“parrafo2”>Parrafo2</P>
<P ID=“parrafo3”>Parrafo3</P>
```

Selectores

Los selectores sirven para definir los elementos sobre los que se van a aplicar reglas CSS. Hay distintos tipos de selectores que se pueden combinar utilizando unos operadores (llamados combinadores) para hacer selecciones más complejas.

Selector de tipo

```
div {
// El estilo se aplicará a los elementos div.
}
```

Selector de clase

```
.example {
// El estilo se aplicará a todos los elementos que
tengan la
// clase 'example'.
}
```

Selector de id

```
#example {
// El estilo se aplica al elemento con el id
'example'.
}
```

Selector universal (Usar con combinadores)

```
* {
// El estilo se aplica a todos los elementos.
}
```

Selector de atributo

```
[attr] {
// El estilo se aplica a todos los elementos con ese
atributo.
}
```

```
[attr=value] {
// El estilo se aplica a todos los elementos con ese
atributo y // valor.
}
```

Combinador de hermanos adyacentes

```
h1 + p {
// El estilo se aplica a todos los p que estén
inmediatamente a // continuación de un h1.
}
```

Combinador general de hermanos

```
h1 ~ p {
    // El estilo se aplica a los p que son hermanos
del h1 que les // precede.
}
```

Combinador de hijo

```
div > p {
// El estilo se aplica los elementos p que son hijos
de un div.
}
```

Combinador de descendientes

```
div li {
// El estilo se aplica a los li que estén dentro de
un div, da // igual si son hijos directos o no.
}
```

Unidades de Medida

¿Qué es?

Definen la altura, anchura y margen de los elementos a través de un valor numérico (entero o decimal), seguido de una unidad de medida. Para establecer la disposición de los elementos en el documento.

UNIDADES RELATIVAS

- **em**: unidad relativa a la propiedad *font-size* del elemento padre.
  - `html { font-size: 13px }`
  - `h1 { font-size: 2em } // 13px * 2 = 26px`
- **rem**: igual que la anterior, pero esta es relativa con respecto al *font-size* del elemento *root* (*root* es la etiqueta html). **En caso de no definirlo, toma el valor por defecto que son 16px**. En el ejemplo anterior, 1rem = 13px. Esto es muy útil porque si algún usuario decide cambiar el tamaño de letra por defecto del navegador, todos los elementos del árbol serán flexibles al cambio.
- **ch**: relativa al ancho del cero (0).
- **vw**: relativa al 1% del ancho del *viewport*. El *viewport* es el tamaño de la ventana
- del navegador.
- **vh**: igual que la anterior, pero relativa al 1% del alto del viewport.
- **%:** relativa al elemento padre.

UNIDADES ABSOLUTAS

- Tamaño fijo no cambia
  - px (píxeles).
  - cm (centímetros).
  - mm (milímetros).
  - in (pulgadas): equivalente a 25,4 milímetros.
  - pt (puntos): equivalente a 0,35 milímetros.
  - pc (picas): equivalente a 4,23 milímetros.

Flexbox

¿Qué es?

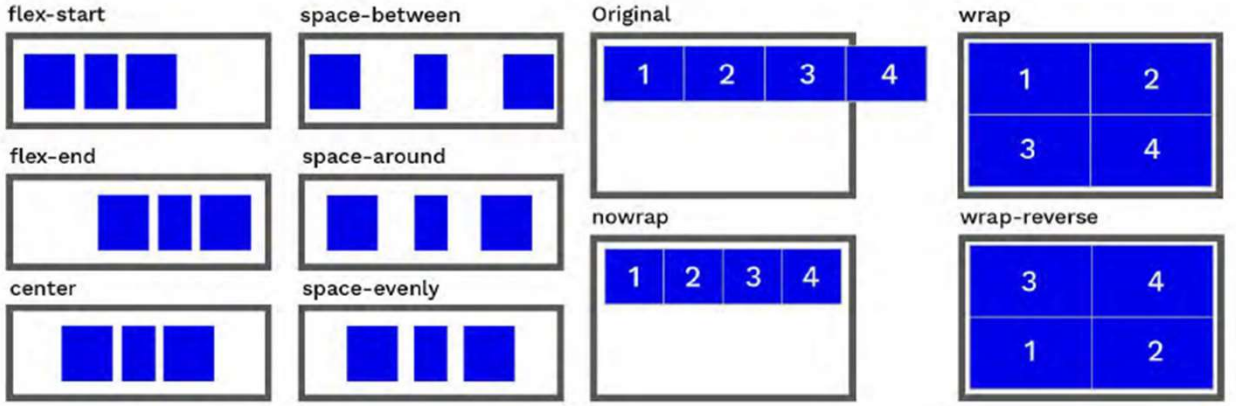
Estilo CSS utilizado para **establecer el espacio de los elementos en filas o columnas de una forma dinámica y sencilla** y que permite **desarrollos responsive** gracias a elementos flexibles que se adaptan automáticamente al contenedor.

PREVIO A FLEX

- Antes se usaban
  - En línea (display:inline).
  - En bloque (display:block).
  - En tabla (display:table).
  - Position (static, relative, absolute, etc.).
  - Float (right, left, inherit, etc.).
- Flex es una mezcla de estas propiedades en cuanto a cómo afecta a la disposición de los elementos contenidos en un contenedor.
- Un diseño Flexbox consiste en un flex container que contiene elementos flexibles (flex items).

PROPIEDADES MÁS USADAS

- Para que un contenedor sea flexible, se usa la propiedad display: flex.
- Al usar flexbox, el eje principal es el horizontal en caso de que flex-direction sea una fila, y vertical en caso una columna. El eje secundario será el perpendicular al principal.
  - Alineamiento a lo largo del eje principal: justify-content: flex-start | flex-end | center.
  - Alineamiento a lo largo del eje secundario: align-items: flex-start | flex-end | center.
  - Dirección de los elementos (de izquierda a derecha, de arriba a bajo, etc.): flex-direction: column | row | row-inverse
  - Por defecto, Flex intenta ajustar los elementos en una fila pero esto se puede modificar: flex-wrap: wrap | no-wrap | wrap-inverse.



Grid Layout

CSS Grid o Grid Layout, es un estándar de las Hojas de Estilo en Cascada que nos **permite maquetar contenido ajustándose a una rejilla** en dos dimensiones totalmente configurables mediante estilos CSS.

CONCEPTOS BÁSICOS

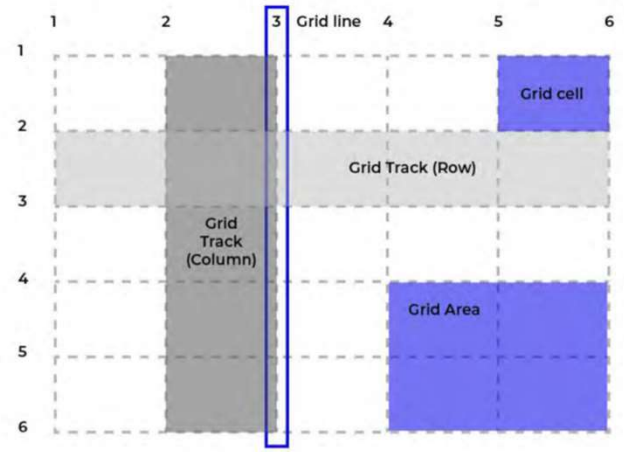
Grid Layout se compone de líneas horizontales (para las filas) y verticales (para las columnas).

El espacio delimitado entre dos líneas consecutivas se le llama track.

Una vez que especificamos el número de filas y columnas, Grid Layout numera las líneas automáticamente.

Una celda es el espacio que define la intersección de las líneas verticales y horizontales, teniendo el tamaño 1x1 en nuestra rejilla.

Un Grid es el espacio que ocupa más de una celda en nuestra rejilla.



CARACTERÍSTICAS

- Es parte de la especificación de CSS, por lo que **no hay problemas de incompatibilidades**.
- Los ítems cuya posición no se especifique **se colocaran solos** (de manera automática), gracias al algoritmo de **auto placement**, ya que Grid es una rejilla, **no es una tabla**.
- Nos ofrece una **sintaxis muy extensa** en su especificación.
- Nos facilita la creación de diseños complejos con layouts.
- **Grid Layout y Flexbox se pueden combinar**.
- Permitiéndonos contener dentro de un Grid una estructura hecha en Flexbox que sólo crece en una dirección.
- Un contenedor en Flexbox es el conjunto de ítems en una dirección, a diferencia de CSS Grid en el que **cada Grid es un contenedor**.

Recursos de Aprendizaje

- Para aprender mas de CSS sugerimos la siguiente dirección:
- https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics
-
- También la guía de referencia de CSS
- https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

Fin