RESUMEN WEB I

HTML

Fue creado para dar estructura y contenido.

El CSS es un lenguaje de presentación creado para dar estilo al contenido.

Ambos lenguajes son independientes uno del otro.

Tag:

- son los que definen los objetos dentro de la página, h1, h2, p, a, div, html, head, etc.

Atributos:

- Son propiedades usadas para agregar instrucciones adicionales a los elementos

DocType:

- Es usado para decirle al navegador que versión de HMTL va a usar el documento
- **No es un tag HTML**

HTML – Body:

- Utilizado para poner todo el contenido visible de la página web.
- Puede haber solo uno par archivo HTML.

<h1>…<h6>:

- Usar **solo para títulos**.
- Evitar saltar niveles.

Head:

- Cualquier cosa que se ponga dentro de este tag no se ve en la pag.

- Usado para colocar la metadata de la página.

Imágenes:

- El atributo “**alt**” se utiliza para poner un texto alternativo, si no se puede mostrar la imagen.
- Buena práctica para la gente no vidente.

CSS

Selectores:

- Determina el estilo que se le va a aplicar a un elemento.
- **Clase, id o tipo** que depende del tag.

Propiedades:

- El estilo que se aplica puede ser una o varias propiedades.

Valor:

- Define el valor que toma la propiedad.

Layouts

Un **layout** define la estructura básica de la interfaz de usuario en una aplicación (es el esqueleto de la página).

Box Model:

El concepto dice que cada elemento en una página se representa mediante una caja rectangular (contenedor).

- El CSS permite controlar el aspecto y ubicación de las páginas.
- Todos los HTML son cajas.
- Fondo y borde pueden ser transparentes.

El **box model** consta de 4 partes:

- CONTENT: alto y ancho de un elemento
- PADDING: genera espaciado **interior**
- BORDER: borde de una caja
- MARGIN: genera **margen exterior**

Contenedores:

** & <sapan>**

- Simples contenedores de HTML
- Cajas sin significado semántico
- Se pueden usar id o class
- Ponerle un nombre representativo

Div:

- Elemento que define un bloque
- Pude incluir varios elementos
- Ayuda con el diseño y layout

Span

- Es un elemento “inline”
- Usado para agrupar texto palabras o frases dentro de un párrafo.

BLOCK:

- Las cajas block por defecto se apilan una encima de la otra.

INLINE:

- Las cajas inline no mueven los elementos alrededor de ellas.

Flex

Le da al contenedor la capacidad de alterar las dimensiones y orden de sus ítems para manejar mejor el espacio disponible.

Propiedades del eje principal:

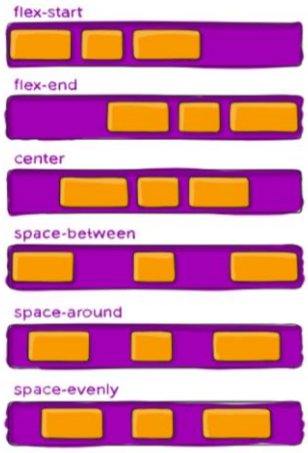
- **Eje principal** definido por la propiedad *flex-direction*
- **Eje transversal** es perpendicular al principal

Flex-direction:

- row
- columna
- row-reverse
- columna-r

Flexbox alineación:

La propiedad ***justify-content*** define la alineación de los componentes **a lo largo del eje principal**

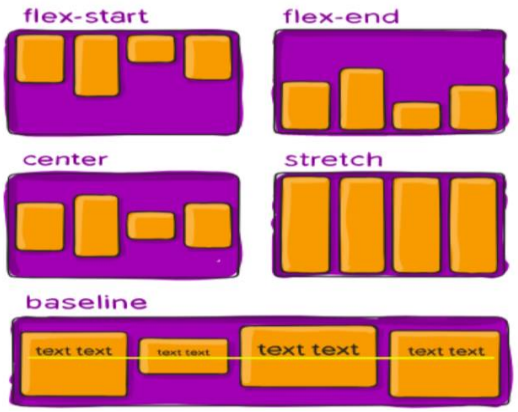


Ayuda a distribuir el espacio libre entre los items.

La propiedad ***flex-wrap*** especifica si los elementos flexibles deben ajustarse o no al contenedor.

- nowrap
- wrap
- wrap-reverse

Align-items: define la alineación de los componentes a lo largo del eje perpendicular



Propiedades para cada elemento interior:

- order: posicion de cada uno.
- flex-grow: cuando crecera en relacion con el resto
- flex-shrink: cuanto encogera en relacion con el resto
- flex-basis: especifica la longitud inicial de un elemento.

align-self: similar a *align-item* pero para un elemento particular:

- stretch
- flex-start
- flex-end
- center

position: sirve para posicionar un elemento dentro de la pagina.

- static → valor por default. Mantiene el flujo normal.
- absolute → permite usar *top, right, bottom* y *left* para mover el elemento.
- relative → rompe el flujo normal.
- fixed → rompe el flujo. Establece una posicion fija en la pantalla.
- sticky → es posicionado en base al scroll del usuario.

Tablas y formularios

TABLE:

- Filas → <**tr**>
- Celdas → <**td**> (se puede colocar texto, img, etc).

FORMS:

- Son usados para pasar info al servidor.
- Se utiliza el tag <**form**> para definirlo.
- Dentro de ese tag se puede tener diferentes tags

<**input**>

- Texto
- Pasword
- Checkbox
- Radio
- Botones
- Botones de submit
- Labels

Text → se usa para capturar texto.

Password → se usa para poner una contraseña.

Radio → se usa para seleccionar una opcion de un grupo.

Checkbox → se usa para seleccionar una o mas opciones.

Select → despliega un menu de opciones.

Text area → se usa para que el usuario escriba un mensaje.

<**label**> → se usa para poner un mensaje en pantalla antes del input.

Hyperlink

- Se define con el tag <**a**>
- Se lo conoce como anchor

Dentro del **href** = “link a la página”

target = “..”

- _**blank** → abre en una nueva ventana
- _**parent** → abre la ventana en el frame padre
- _**self** → abre en la misma ventana

JavaScript

- **No tiene relación con JAVA.**
- Uno de los lenguajes más utilizados
- Principal uso, por el lado del cliente
- Valida formularios
- Reacciona a lo que hace el usuario
- Cambiar al pasar el mouse
- Hacer cálculos
- Carga dinámica de contenido (AJAX)
- Partes de las páginas que se muestras/ocultan

Funciones en JS:

- Dividen un problema grande en varios problemas chicos
- Facilitan la lectura del código
- Se ahorra repetir código

Eventos en js:

- Un evento es algo que ocurre en el sistema, originado por el usuario u otra parte del sistema y que avisa al sistema.

DOM

- Representación estructurada del documento
- Permite modificar el contenido
- Conecta a las páginas web con js.
- Es un árbol de objetos.
- Las propiedades del DOM se pueden leer/editar desde js.

Operadores de comparación:

- Igualdad → ==, (!=)
- Identidad → ===, (!==) **recomendable a veces**

CSS AVANZADO

*{…} → **Selector universal**

- Se utiliza para seleccionar todos los elementos de la pag.

HERENCIA

- Mecanismo mediante el cual determinadas propiedades de un elemento padre se transmiten a sus hijos.
- No todas las propiedades CSS son heredadas, porque algunas de ellas no tendrían sentido que lo fueran.

- Todos los elementos de un documento HTML heredan todas las propiedades heredables de su padre excepto el elemento raíz (html), que no tiene padre.
- El valor por defecto es *inherit.*

Para que sirve la herencia?

Si ponemos *font-family* al elemento HTML

- Todos los elementos lo heredan
- No tengo que reescribir código para que otros lo tengan
- Mas mantenible
- Menos código repetido

SELECTORES COMBINADOS

Se pueden combinar selecciones para hacerlas mas especificas.

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

Elige los parrafos que tengan la clase “destacado”.

SELECTORES ANIDADOS

Permite seleccionar elementos contenidos dentro de otros elementos. Asi se puede aumentar el nivel de detalle.

```
ul li a {
   text-decoration: none;
   color: red;
   background-color: yellow;
 }
```

Aplica estilo a los links que se encuentren dentro de un item de lista no ordenada.

GRUPO DE SELECTORES

- Se pueden usar varios selectores juntos
- Permite evitar duplicación de estilos
- Se separan los selectores con ‘,’ creando un grupo de selectores con propiedades en común.

UN ESPACIO O UNA COMA PUEDEN HACER LA DIFERENCIA

CASCADA

- La cascada es el mecanismo que controla el resultado final cuando se aplican varias declaraciones CSS contrapuestas al mismo elemento.

Hay tres conceptos principales que controlan el orden en el que se aplican las declaraciones de CSS:

- IMPORTANCIA
- ESPECIFICIDAD
- ORDEN EN EL CODIGO FUENTE

IMPORTANCIA

La palabra reservada **!important,** sobrescribe toda la cascada.

HTML

```
<p>Este es un parrafo</p>
```

CSS:

```
* {
  color: red !important;
}
 p {
  color: blue;
}
```

ESPECIFICIDAD

- Mas especifico es el selector, entonces ese es el estilo que se aplica.
- Lo que tiene clases es mas especifico que lo que no.
- En un selector anidado, si tienen la misma cantidad de clases, gana el que tiene mas elementos para cumplir.

ORDEN

Si dos declaraciones afectan al mismo elemento, tienen la misma importancia y la misma especificidad, la selección final es el orden en las fuentes.

La declaración que se ve después en las hojas de estilo “ganara” a las anteriores.

Pseudo-clases

Las pseudo-clases se utilizan para definir un estado o comportamiento especial de un elemento.

Cambia el “cuando” (solo cuando paso el mouse)

Pseudo-elemento

Un pseudo-elemento de CSS es usado para dar un estilo a una parte de un elemento. Pueden usarse combinados.

Cambia “el que elegimos” (solo el primer elemento)

HTML5 Y CSS3

Las paginas en HTML5 funcionan en múltiples dispositivos

- Desktops
- Mobiles
- Tablets

Como expresa mejor semántica, es mejor para los lectores de pantalla para personas no videntes

IFrame

- Permite incluir otra pagina HTML dentro de mi página.
- Se usa para contenido que en general no es de mi web, pero lo quiero incluir.
- Videos de YouTube, Mapas de Google, el clima, etc.
- Tiene una propiedad src que apunta a la página que queremos incluir.

```
<iframe width="560" height="315" src="https://www.youtube.com/embed/mzPxo7Y6JyA"
frameborder="0" allowfullscreen></iframe>
```

RESPONSIVE

Es una tecnica de diseño web que busca la correcta visualizacion de una misma pagina en distintos dispositivos.

Implica:

- Cambio de dimensiones
- Distribución dinámica de los elementos

Ventajas de Responsive:

- Mismo sitio para todos los dispositivos
- Mejora la experiencia del usuario
- Menor costo de desarrollo
- Menor costo de mantenimiento

Media Queries

- Es un recurso de css que permite asignar diferentes estilos para tamaños y resoluciones de pantalla
- Nos da la posibilidad de entregar distintas apariencias para diferentes dispositivos
- Es la base del diseño responsive
- Se utilizan **breackpoints** para indicar que ciertas partes del diseño se comportan diferentes bajo ciertas condiciones

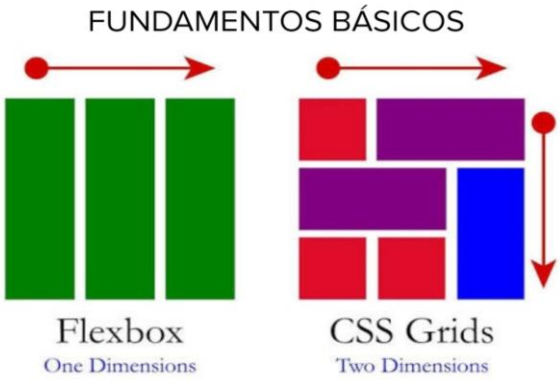
@media only screen and (min-width: 600px) {…} **mobile first**

@media only screen and (max-width: 600px) {…} **desktop first**

Mobile First

1. Desarrollar el diseño móvil primero
    - Programas una pantalla pequeña con menos elementos
2. Luego ajustarlo a las grandes pantallas
    - Agregas otros elementos a medida que hay más resolución.

CSS GRID



CSS Grid permite dividir un contenedor en varias secciones, permitiendo posicionar y alinear items en columnas y filas. Se puede colorcar items donde quieras, en cualquiera de las celdas que lo forman.

Grid: filas y columnas

grid-template-columns

grid-template-rows

- La propiedad especifica el numero y ancho de las columnas y filas en un diseño de cuadricula
- Los valores son una lista separada por espacios, donde cada valor especifica el tamaño de la columna respectiva.

```
/* agregamos un contenedor grid */
.middle-wrapper {
  display: grid;
  grid-template-columns: 1fr;
}

/* solo para pantallas superiores a 660px */
@media only screen and (min-width: 660px) {
   .middle-wrapper {
        grid-template-columns: 3fr 1fr;
    }

  }
```

CAMBIAMOS LA CANTIDAD DE COLUMNAS

Grid Areas

grid-template-areas

- Especifica nombres para cada uno de las secciones del grid.
- Repetición del área permite que el contenido abarque múltiples celdas.
- La sintaxis visualiza la estructura de la cuadricula

grid-area

- Asocia el nombre de área a un ítem. Especifica en que área se posiciono el ítem

```
/* Page Layout */
 .container {
    display: grid;
    grid-template-areas:
         "header"
         "main"
         "sidebar"
         "footer";
  }
 /* posicionamos los elementos donde queremos */
  header { grid-area: header; }
  .main { grid-area: main; }
  .sidebar { grid-area: sidebar; }
  footer { grid-area: footer; }

/* solo para pantallas superiores a 660px */
@media only screen and (min-width: 660px) {
  /* Page Layout */
  .container {
    display: grid;
    grid-template-columns: 4fr 1fr;
    grid-template-areas:
         "header header"
         "main sidebar"
         "footer footer";
  }
}
```

grid-column-gap

grid-row-gap

- Define el tamaño del espacio entre las filas y las columnas en un diseño de cuadricula.

Flexbox vs Grid

CSS Grid es un sistema bidimensional, lo que significa que puede manejar columnas y filas.

Flexbox es un sistema unidimensional que define un eje y el otro actúa en función a este.

Semántica

La semántica se refiere a todo aquello que esta vinculado o pertenece a la significación de las palabras. La misma esta asociada al significado, interpretación y sentido de las palabras.

Beneficios de usar código semántico:

- Darle a computadoras, buscadores y otros dispositivos la capacidad de leer y entender páginas web
- Fácil de trabajar con el código semántico
- Fácil de manejar
- Comprender para que esta cada porción de contenido.

<header>

- Usado para identificar la cabecera de una pagina, articulo, sección
- Va a incluir un texto de introducción o navegación
- Se puede usar mas de un header por pagina

<nav>

- Utilizado para identificar la sección donde están los links de navegación de la pagina
- No todos los links tienen que estar dentro de un nav
- Tiene que ser reservado solo para la navegación primaria

<section>

- Es definido para representar una sección de un documento o una aplicación
- No hay que usarlo solo para poner estilos
- Puede haber más de un section
- Es más fácil de confundirlo con un div
- Se usa para agrupar elementos que conceptualmente están relacionados

<article>

- Este elemento es parecido al div o section
- Definido para incluir contenido independiente que puede ser distribuido y reutilizado
- En un blog, es una publicación y cada comentario también
- El contenido del articulo tiene que tener sentido por si solo
- Puede haber más de uno

<time>

- Indica la fecha de publicación
- Refiere al artículo que lo contiene

<aside>

- Contenido relacionado al documento o a la sección que se encuentra alrededor.
- Puede ser utilizado muchas veces por pagina

<footer>

- Es igual que el header
- La diferencia es que es usado para la parte mas baja de la página.

<figure>

- Se crea un contenedor para imágenes/video/audio
- Introducido para dar más semántica
- Permite más de una imagen/video/etc.

<figcaption>

- Sirve para ponerle título a una imagen

```
<figure>
  <img src="https://bit.ly/1YPriWf" alt="McFLy" />
  <figcaption>Marty McFly</figcaption>
```

Marty McFly

```
    <img
src="https://66.media.tumblr.com/tumblr_lxay62n5JH1qhc4
v3o2_r2_250.jpg" alt="DcBrown" />
    <figcaption>Doctor Brown</figcaption>
  </figure>
```

Doctor Brown

Javascript #2

Variables globales

- En lugar de declarar las variables y funciones como globales podemos incluirlas en un objeto para aislarlas
- Crear un ambiro de todo el documento de JS para que las variables no estén disponibles desde fuera de el.

- Esa función la ejecutamos cuando el DOM se haya cargado

Tipado de variables – tipos

- **El tipado estático** nos obliga a definir desde el principio tipo de una variable. Lenguajes con tipado estático son C++, JAVA, C# entre otros.
- **El tipado dinámico** nos da la facilidad de no definir los tipos al declarar una variable, algunos ejemplos son PHP, JS, Grooby, Python, entre otros.

Tipos

- JS tiene tipos dinámicos.
- Una misma variable puede cambiar de tipo
- Puede causar confusiones

Undefined

- Tipo fundamental en Js
- Las variables sin inicializar valen undefined
- Variables y miembros sin declarar valen undefined
- Las funciones siempre devuelven un valor, si no tienen valor de retorno devuelven undefined

Funciones anónimas

- Se usan para no crear tantas funciones que se usan en un solo lugar
- Es una funcion sin nombre que se escribe directamente donde la quería pasar de parámetro
- En ese caso encapsula a la funcion que si pasa parámetros

Arreglos

- Almacenan elementos en una colección de datos ordenados
- Los elementos se acceden por índice (comienzan en 0)

Javascript #3

JSON

- Encapsulan datos y comportamientos
- Los objetos se caracterizan por ser organizados y fáciles de leer

```
let documento = {
  "titulo" : "Practico JavaScript",
  "autor" : {
    "nombre" : "Web",
    "email" : "web@gmail.com"
  }
}
```

Ventajas

- Super liviano para transferir
- Datos auto-descriptos
- Legible por el humano
- Fácil de adoptar por los lenguajes orientados a objetos

Funciones

Una funcion es un objeto

Parametros

- Tipos primitivos: se pasan por copia-valor
- Tipo objetos: se pasan por referencia

Arrow Functions

Es una forma abreviada para escribir funciones:

Obtener nodos del DOM

- Se puede obtener nodos del DOM consultando por un ID, nombre clase o un selector.
- Podemos obtener como resultado de uno o múltiples elementos del DOM

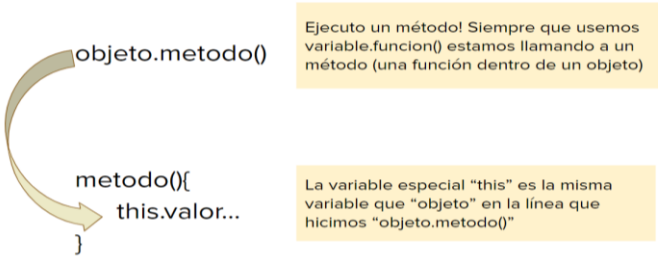
Recorrer el DOM

Los elementos del DOM se pueden recorrer como un árbol y ser localizados:

- element.children, encuentra los elementos hijos
- element.parentElement , encuentra el elemento padre
- element.nextElementSibling , encuentra el siguiente hermano
- element.previousElementSibling , encuentra el hermano anterior
- element.firstElementChild , encuentra el primer hijo
- element.lastElementChild , el último hijo

This

- This hace referencia al objeto que ejecuto el método



Localizar todos los elementos de una clase

```
// Búsqueda de todos los botones con una clase
let btns = document.querySelectorAll('.btn');

// asignación de evento a todos los elementos
for(let i = 0; i < btns.length; i++) {
    btns[i].addEventListener('click', miFuncion);
}
```

Eliminar elementos del DOM

- Metodo remove()

En cada elemento se puede hacer el .remove() para eliminarlo del DOM.

```
document.querySelector("#id").remove()
```

SPA Y AJAX

Single Page Application (SPA)

- Es un tipo de aplicación web
- Nunca se recarga completamente el navegador
- Los recursos se cargan parcial y dinamicamente cuando lo requiera la pagina

Asynchronus Javascript And XML (AJAX)

- No es un lenguaje o tecnologia
- Hace mejorar la experiencia de usuario en las paginas web (mas amigables y rapidas)
- Permite **cambios dinamicos** al lado del cliente
- Mejora la **experiencia del usuario**

Partial Render AJAX

- Carga un fragmento HTML y lo inserta en nuestro html

Un ejemplo seria apretar un boton y mostrar un pedazo de HTML dentro de un DIV

ES7 incorpora la interfaz **fetch()** para llamados **ajax**

```
let promise = fetch(url);

fetch(url).then(response => …do something )
```

Promesas

Js es de un solo hilo (mono thread), es decir, dos porciones de secuencias de comandos no se pueden ejecutar al mismo tiempo. Solamente puede ejecutar uno despues del otro

- ES6 introduce **promises**
- Son un objeto que representa la terminacion o el fracaso de una operación asincronica

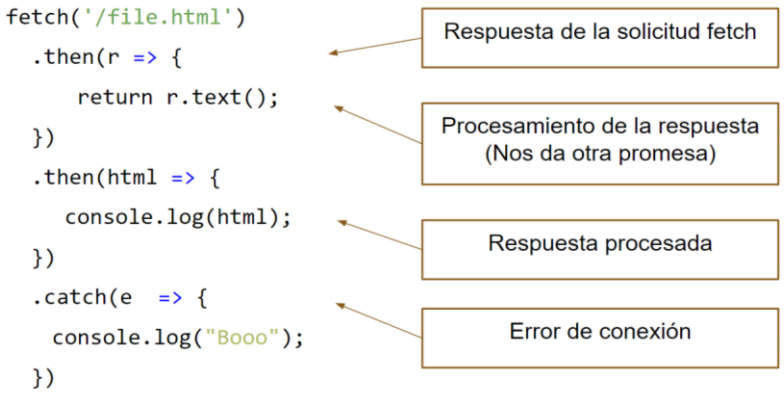
La carga parcial consiste en ir a buscar texto HTML a un archivo externo y ponerlo en un div con ID “use-ajax”

response

El objeto “response” tiene informacion de la respuesta obtenida del servidor.

Con response.ok nos dice si la descarga pudo hacerse correctamente

¿Qué está ocurriendo en cada llamado a la función **then()**?



AJAX asincronico

Orden de ejecucion:

- Primero se hace la promesa
- Luego se sigue ejecutando
- Finalmente se ejecuta la resolucion de la promesa

AWAIT ASYNC

Son palabras reservadas de ES8

ASYNC:

- Hace una funcion que devuelve una promesa
- El return se encapsularan en la promesa automaticamente.

AWAIT

- Desencapsula el contenido de una promesa
- Se reescribe como THEN de la promesa
- Solo puede usarse dentro de funciones ASYNC

```
async function load2(event) {
  event.preventDefault();
  let container = document.querySelector("#use-ajax");
  container.innerHTML = "<h1>Loading...</h1>";
 try {
    let response = await fetch(url);
    if (response.ok) {
      let t = await response.text()
      container.innerHTML = t;
    }
    else
      container.innerHTML = "<h1>Error - Failed URL!</h1>";
  }
 catch (error) {
    container.innerHTML = "<h1>Connection error</h1>";
  };
}
```