# Clase 1 – Introducción al FrontEnd

Si bien se ve CSS y HTML la materia esta enfocada en JavaScript. Tambien veremos modelo Cliente-Servidor, AJAX, JSON, Bootstrap, yarn(npm de facebook: Node Package Manager) y si se llega algo de TypeScript.
Vamos a ver el concepto SAP (Single Application Page) de paginas webs en lugar de hacer MPA (Multiple Page Application, muchos archivos HTML). En vez de tener que recargar cada vez que hacemos una peticion a una pagina vamos a utilizar AJAX para que no se refresque.
AJAX es Asynchronous JavaScript and XML.

JS es un lenguaje asincrono y concurrente. No tiene un pool de hilos como C# o C++ (Que recurren al paralelismo).
El único lenguaje que ejecuta un navegador es JavaScript. Es importante aprender JavaScript ya que los frameworks como Vue, React o Angular lo implementan en su base.
JavaScript implementa la asincronia con callbacks, promesas y funciones asincronas.
JavaScript es de debil tipado y dinamico. Puede contener cualquier en su declaración y la variable puede mutar a cualquier tipo conocido durante la ejecución.

Transpilar significa traducir de un lenguaje de programación a otro, por ejemplo transpilar de TypeScript a JavaScript para un navegador web.

¿Que es la arquitectura cliente servidor?. El browser (cliente) envia peticiones al servidor que tiene los recursos que le van a pedir los clientes. El servidor tiene una programación backend que se encarga de manejar la petición con la base de dato del backend.
Tanto el cliente como el servidor tienen un IP (Internet Protocol) para ser identificadas unequivocamente. Los servidores tienen una IP fija ya que quieren ser encontrados, el dueño del servidor paga por esa IP para que no cambie. Los clientes tienen una IP variable ya que no le interesa que lo encuentren, le interesa encontrar el servidor, la IP del cliente se la da el provedor de internet.
El router del provedor genera una IP que se le asigna al cliente, el router generará una subred para cada dispositivo conectado.

Para solicitar el HTML de una pagina web debemos conocer la IP del servidor. Los seres humanos no son buenos para recordar los numeros de las IP's, por lo tanto recurrimos a una direccion URL. http es el protocolo de transferencia de informacion, www es el tipo de web (world wide web), y el resto es el nombre de fantasia que enmascara que esta asociada con la IP.
Los DNS (Domain Name Server) son los encargados de descrifrar en donde se encuentra la IP de ese nombre URL que ingresamos. Nuestro navegador va a consultar el DNS para conectarse al servidor.
El recurso que envia ese servidor ante una petición por defecto es el index.html. Si bien se puede cambiar y configurar el servidor para que cambie, por convención se utiliza index.html.
Luego en ese index HTML habrá links para pedirle al servidor otro tipo de recursos según nuestra interacción.

JavaScript es un lenguaje interpretado. En cambio C# o C son compilados, nuestro SO va a ejecutar un .exe con las intrucciones ya programadas dentro de sí empezando por main.
¿Que significa que es interpretado? Que hay alguien que lo lee sentencia a sentencia y lo va ejecutando en tiempo real (Lo lee por ejemplo el motor V8 de google en navegadores). Un motor es el encargado de leer el Script.

JavaScript realmente es ECMA Script (Script = libreto). El Script es texto plano que se puede escribir practicamente en cualquier lado, el motor va a leerlo en tiempo real linea a linea y lo va a ejecutar.
Esto va a provocar que los errores van a aparecer/romper en el programa cuando el motor ejecute esa parte del codigo.
En los lenguajes compilados el compilador revisa el 100% del codigo fuente y si no encuentra errores va a compliar y va a permitir ejecutar un .exe.

¿Que es NodeJS? Cada empresa que hace un navegador web implementa un motor de JavaScript. Chrome por ejemplo utiliza V8. Ese motor V8 se separo de los navegadores y se permitio que se utilice en contextos no-web, por ejemplo ejecutar un archivo JS a través de una terminal.
Esto permitio la aparición de librerias, funciones, paquetes, etc.
El NPM (Node Package Manager) es un administrador de paquetes de Node. Es una gran base de datos de paquetes/codigos JavaScript.

HTML es un lenguaje marcado de hipertexto, no es un lenguaje de programación. Tiene etiquetas que estructuran la pagina y le dice al navegador cuando renderiza la interfaz como funciona.

En las etiquetas se estructuran por una etiqueta de apertura y cierre, estas tienen contenido dentro. Otras etiquetas son de self-closing. Ambos tipos de etiquetas tienen propiedades. Estos son valores adicionales que configuran los elementos o ajustan su comportamiento de diversas formas para cumplir los criterios de los usuarios.

Hay muchas etiquetas meta que tienen que ver con el SEO (Search Engine Optmization). Son para tratar que nuestras paginas aparezcan en los primeros resultados del motor de busqueda. Vamos a incrporar las etiquetas semanticas para que los motores de busqueda aparezcan para ciertos resultados.

<!DOCTYPE html> -> Le dice al navegador en la "precompilacion" que este archivo es codigo HTML.
<html lang="es"> -> La etiqueta html indica el inicio y cierre del codigo HTML. Con la propiedad lang le decimos al navegador el lenguaje de nuestra Web. Por ejemplo si esta en ingles pero escribimos en español el navegador nos va a dar una opción para traducir la pagina.
<head> -> No es visible para el usuario salvo la etiqueta title. Las etiqueta metas son de metadata, son etiquetas para el navegador y motores de busqueda.
    <meta charset="UTF-8"> -> Indica el juego de caracteres que debe interpretar el navegador.
    <meta http-equiv="X-UA-Compatible" content="IE=edge"> -> Es una etiqueta para productos Microsoft.
    <meta name="viewport" content="width=device-width, initial-scale=1.0"> -> El viewport es la parte visible de nuestra pagina web. Le dice que el viewport sea igual al maximo que permite el dispositivo y que la escala (zoom) sea 1. Tiene que ver con el concepto de responsive.
    <title>Document</title> -> El titulo que se despliega en la pestaña.
    <link rel="stylesheet" href="./estilos.css">  -> Linkeo mi CSS y le indico en el rel que tipo de archivo/relacion tiene con el HTML.
</head>
<body></body>  ->  Vamos a escribir el codigo que se va a ver/renderizar.

En la programacion hay tiempo de desarrollo y a hay tiempo de producción. Cuando estamos desarrollando nuestra computadora tiene una personalidad multiple, vamos a tener 3 programas, VSC, cliente (navegador web) y un servidor. Con el plugin Live Server nuestro navegador va a ser peticiones al servidor.
Cuando nuestro producto esta terminado lo enviamos a producción. Esto significa que estos archivos van a ir a un Hosting donde vmaos a alojar nuestros archivos.
¿Como hace la maquina para hablar con ella misma (navegador-servidor)? Cuando hablamos de nuestro Hardware, para hablar con ella misma utiliza su propia direccion IP. Su nombre de fantasia es localhost, y los programas que se conectan por red a una computadora utilizan puertos ($2^{32}$ puertos disponibles). 127.0.0.1:PUERTO es el "this" de nuestra IP.
Algo asi como si nuestra PC fuese un edificio con una direccion (IP) y los puertos son los departamentos que tiene esa calle.

Las etiquetas de bloque ocupán todo el anocho disponibles del viewport a pesar de que el contenido no ocupe todo el ancho.
Las etiquetas de linea tan solo ocupan el ancho que ocupe el contenido.

Hay etiquetas que se llaman etiquetas semanticas tienen un significado para el motor de busqueda.
La etiqueta h1 deberia estar sola en la pagina como titulo de pagina, h2 como subtitulo, h3 como subittulo del subtitulo y así. Es una etiqueta de bloque.
Las etiquetas de parrafo genera un parrafo. Es texto que deja una linea por arriba y una linea por debajo. Es una etiqueta de bloque.
Si queremos formatear el contenido dentro de una etiqueta HTML podemos utilizar HTML Entities. https://dev.w3.org/html5/html-author/charref https://www.w3schools.com/html/html_entities.asp
La etiqueta br singifica breaking rule, es una especie de \n.
La etiqueta strong es muy parecida a la b, la b agrega de manera visual negita, strong visualmente hace lo mismo pero en el contexto donde se encuentra le indica al navegador que el texto es importante. De la misma manera funcionan la i de italica y la em de emphasis. Estas son conocidas como etiquetas semanticas.
La etiqueta contenedora span es un elemento en linea que me sirve para contener una seccion del texto sin contaminarla y me permite a esa seccion agregarle una clase.
La etiqueta div es una etiqueta contenedora en bloque.
Las etiquetas contenedoras semanticas son la section (seccion), nav (barra de navegacion), footer(pie de pagina), header(cabecera), main(etiqueta principal, deberia haber una sola por pagina), article(articulo, encierra una seccion independiente al resto) y aside (ponemos contenido que no tiene que ver con el tema de la pagina, publicidad x ejemplo. Los article suelen estar incluidos dentro de un section. Funcionan como div pero son semanticas.
No se utilizan etiquetas semanticas por utilizar, debemos usarlas siempre y cuando sean necesarias. Se pueden ver en HTML viejos etiquetas div con clases footer, nav, etc. No se recomienda ya que las nuevas etiquetas llegaron para mejorar los motores de busqueda y dar significado.

En la vida real una diseñador utiliza una herramienta de diseño para crear una pagina que genera un archivo (jpg, png, etc), ese archivo pasa al programador el programador es el encargado de llevar el maqueteado al HTML.

Existen 3 tipos de listas
Listas ul: listas desordenadas, dentro tienen etiquetas li: list item.
Listas ol: listas ordenadas, dentro tienen etiquetas li: list item.
Listas dl: listas de defincion, tiene una relacion de dos etiqueta key:value. dt y dd.

Tablas: Todas las tablas tienen filas pero no columnas (table row). Sin embargo podemos darle estilo y podemos crear un cabecero con table header. Las table data dan informacion a las table row. A su vez la table row que tiene lo th es una buena practica encerrarla en la etiqueta thead. Y el resto en una etiqueta tbody. En el footer podemos utilizar un tfooter.

```
<table>
  <thead>
  <tr>
    <th>Id</th>
    <th>Nombre</th>
    <th>Apellido</th>
    <th>Edad</th>
  </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>Juan</td>
      <td>Perez</td>
      <td>30</td>
    </tr>
  </tbody>
</table>
```

Es importante que HTML/CSS/JS tengan su archivo correspondiente y no mezclar. Es una buena práctica y es mucho más fácil dar mantenimiento.
Se le puede aplicar a una etiqueta estilos en linea. No es una buena practica ya que es muy dificil de mantener y podria generar conflictos con nuestro archivo CSS por un tema de jerarquias. Son en linea porque afectan a una sola etiqueta.
Tampoco se recomienda el estilo a nivel de pagina (style en el head) porque mezclo en un archivo HTML estilos CSS y tambien puedo generar conflictos con un archivo externo CSS.
La mejor practica es tener un archivo CSS aparte y linkearlo al HTML. Para linkearlo utilizamos la etiqueta <link href="PATH_RELATIVO_CSS" rel="stylesheet">
Existen dos tipos de path, el asboluto (ej: C://Documentos/Carpeta/achivos.css) el relativo es a partir de donde estoy hasta donde quiero llegar (ej: Carpeta/archivo.css). Sin embargo por un tema de servidores vamos a agregar un ./ antes del archivo (ej: ./archivo.css), el ./ indica que estamos en el directorio actual.

La jerarquia CSS-HTML se conoce specificity/especificidad. Lo veremos mas adelante en CSS. Es una relación de cascada que va de arriba (general) al mas especifico (sobreescribe al general si tiene relacion).

Las etiquetas se manejan en modelos de caja, se puede ver en Chrome en la consola de desarrollo en la parte de Elements->Computed

Cuando un servidor devuelve la peticion al navegador solamente le manda el index HTML. El navegador cuando lo lee puede reconoce que el header del HTML que tiene linkeado un CSS, entonces vuelve a hacer una petición para traer el CSS y continua leyendo el resto del HTML renderizando con el CSS linkeado la pagina en el navegador.
El navegador espera al CSS ya que para renderizar necesita esos estilos que aplique a las tags, sin embargo no espera al JS.
Por eso al no esperar el JS ponemos el script al final del HTML, ya que debe esperar que se renderice todo antes de aplicar codigo JS.

El DOM es Document Object Model, esto quiere decir que los documentos HTML se organizan en objetos. document.getElementById(""); nos devuelve un puntero al objeto del DOM. Al ser un objeto también nos permite acceder a las propiedades con el operador '.'.