**~~5/9/2023-5/12/2023 Token github: ghp\_RHoOWifuu1CzmYBpaUOSlTIuDyn5p63LfOie~~**

**6/9/2023-6/10/2023 Token github: ghp\_OpGqbijYTFy1cjykDqhmo6lgAt6NVP2MN94w**

**6/9/2023-6/10/2023 Token github: ghp\_Ca9z3XfYlvLinlOpOYjwmwUKsiKkWr348A7Z**

-Curso-JavaScript-DOM- 10/08/2023

Que es el DOM? Ese es el modelo que vamos a usar durante el curso y está estrechamente relacionado con el desarrollo web, nos va a permitir crear páginas interactivas que necesitamos. Cuando se crea una página web normalmente se crea un archivo HTML para definir la estructura de la página. Esa estructura en el navegador está representada también por una jerarquía. Un elemento está dentro de otro elemento y esos elementos se van a poder representar como ciertos obj en nuestro programa. Toda esa jerarquía de distintos elementos que se van agrupar y estructurar para poder mostrarlos en el navegador es lo que conocemos como DOM pero se va creando en base a lo que definimos en el archivo HTML. El navegador crea una representación de c/u de esos elementos y su ubicación en una jerarquía especifica. El significado de DOM→Modelo de Objetos del Documento. Formal mente podemos definir el DOM como la representación de los obj, los distintos elementos que conforman la estructura de un documento en la web.

A partir de estos elementos podemos generar un diagrama similar a este que representa su jerarquía HTMT



En el grafico vemos al inicio la etiqueta <html> que es el elemento raíz a partir de ahí continúan todos los otros elementos que van hacer descendientes del elemento principal. Luego tenemos <head> y <body> que están en la misma línea, esto nos dice que son hermanos porque están ubicados en el mismo nivel en la jerarquía. Seguidamente dentro de <head> encontramos el elemento <title> y por debajo tenemos otro nivel que representa el texto en sí. El DOM llega hasta ese ínfimo detalle de representar el texto de forma individual. Con <body> ocurre lo mismo, pero en este caso <body> contiene dos elementos. Un elemento <h1> y un elemento <p> están al mismo nivel en la jerarquía pero están contenidos en el elemento <body> y así se puede ir leyendo este diagrama hasta que llegamos al texto que corresponde a cada uno si es que contienen textos. Por ejemplo si tenemos un elemento de imagen no tendríamos texto, esto es básicamente la idea del diagrama. ¿Porque es importante representar el documento a través del DOM? Representar el documento HTML a través del DOM nos permite *acceder* a sus elementos en nuestro programa usando JS y también *manipularlos* que significa crear elementos nuevos, eliminar elementos que ya están en la página, modificar sus atributos, agregar atributos nuevos, cambiar el texto. Todas esas operaciones mencionadas son ejemplos de cómo podemos manipular elementos una vez que ya están representadas en el DOM en el navegador. Es uno de los aspectos más importante que se tiene que saber. El DOM nos permite representar los elementos como *NODOS* y *OBJETOS* con los cuales podemos trabajar *JS* con sus propiedades, métodos y con distintos elementos. Por ej. cuando se hace click sobre un botón todos esos eventos vamos a poder manejarlos y detectarlos en nuestro código gracias al DOM porque en si nuestro archivo HTML es solo eso un archivo pero el DOM es el que le da esa capacidad poderosa para poder trabajar con esos elementos en JS. hay que recordar estos dos conceptos *NODO y OBJETOS* son muy importantes para el DOM los Nodos son muy utilizados y prácticamente fundamentales en el área de estructuras de datos. También en el DOM vamos a tener Nodos pero esos Nodos en lugar de representar información como tal van a representar elementos con todos sus datos y toda esa información y los objetos van a estar presente cuando trabajemos con esos elementos en JS, van a tener propiedades y métodos.

Un Nodo es un punto específico del diagrama o del árbol de Nodos del DOM por ejemplo en el siguiente grafico tenemos la estructura del archivo HTML y el mismo diagrama anterior pero ves que todos tienen un borde amarillo es para indicarles que todos estos elementos son Nodos incluyendo el texto. Individualmente el texto en el DOM se representa como un Nodo y eso va ser importante cuando trabajemos con algunos métodos que nos puede permitir acceder a un elemento o a un Nodo como tal. Los Nodos en el DOM incluyen: El documento en sí, los elementos HTML que definamos en nuestro archivo, también cualquier texto que incluyas entre etiquetas se representa como un Nodo y también algo importante es que los comentarios que escribas en el archivo html también se representan como Nodos. Todo el proceso de transformar un archivo html a un DOM interactivo que podemos usar en nuestro archivo JS lo realiza el navegador o browser en inglés.



**Conceptos Importantes:**

Al referirte al DOM y a los elementos representados en el DOM dentro de su jerarquía, al igual que un árbol genealógico vamos a tener esos elementos representados como una jerarquía. Cual de esos elementos contiene a otros y cuales están contenidos dentro de otros elementos todo eso lo vamos a poder representar y cada tipo de Nodo va a tener un concepto diferente para referirnos a esa ubicación.



Por ejemplo como vemos en el diagrama lo primero que conseguimos en nuestra jerarquía en html es el Nodo raíz es el que contiene a todos los otros elementos. Luego comenzamos hablar sobre las distintas relaciones que puede haber. Primero tenemos parent node o (nodo padre) este nodo es el que contiene a otro nodo en la jerarquía del DOM.



En el gráfico de arriba tenemos señalado el elemento *head* ese nodo es el nodo padre de *title* porque esta contenido por *head* lo mismo ocurre con *body* es el nodo padre de *h1* y de p. Eso también nos lleva a que tenemos otra relación reciproca si un Nodo es el nodo padre el otro nodo contenido por este es el nodo hijo. Digamos entonces que el nodo hijo es el que esté conectado directamente un nivel por debajo el nodo padre. Y en este ejemplo podemos ver que en la misma jerarquía esta h1 y p que son contenidos por su nodo padre body



Ahora que ocurre si la jerarquía es más elaborada como indica el siguiente gráfico. Digamos que como muestra el grafico anterior body contiene a h1 y p y en lugar de eso como vemos en el siguiente grafico en lugar de h1 tenemos un div y dentro de ese div tenemos un h1 y un a, en ese caso ya sería relevante el concepto de Nodo descendiente. Al igual que una familia tendrías a tu tátara abuelo tendría muchos descendientes en la jerarquía que no son hijos directos pero si son sus descendientes porque son sus bisnietos y nietos. Ese mismo concepto del árbol genealógico también aplica aquí al DOM. En resumen: html es el nodo padre de toda la estructura en la cual body es hijo de html a su vez body es padre de div y p y div es padre de h1 y a



Además otro concepto importante que ya hemos mencionado pero aquí tenemos el concepto formal. Esos nodos que están al mismo nivel como podemos observar en el siguiente diagrama se denominan Nodos hermanos y comparten el mismo Nodo padre.



**NODOS Vs. ELEMENTOS**

El concepto de Nodo es más amplio que el concepto de Elementos en el DOM, es importante saber cuál es la diferencia entre ambos. Estos conceptos son fáciles de intercambiar y es fácil confundirse entre un Nodo y un Elemento cuando ya estamos trabajando en el contexto del DOM. Un Nodo puede contener más tipo de información que los Elementos. Usualmente nos referimos a los Elementos del documento HTML como “elementos” es decir toda esa estructura que escribimos en el archivo html una vez que ya se traduce al DOM nos vamos a referir a ellos como elementos. Pero un Nodo puede ser un documento HTML puede representar o ser texto o comentarios que hayas escrito en el documento. Asique el Nodo como tal no solamente representa esas etiquetas html o los elementos como tal, también incluye el texto que es un Nodo aparte para cada uno de esos elementos que contengan texto o también pueden ser comentarios que hayas escrito en el documento html que aunque no tengan una función como tal en la estructura de la página web si son tomados en cuenta al momento de crear el DOM y crean nuevos nodos en el DOM por eso es la diferencia. El Nodo es un concepto más amplio que el Elemento

**Herramientas de Desarrollo de Chrome**

Veamos cómo usar las herramientas de google Chrome con la cual vamos a trabajar en el curso, en estas herramientas vamos a poder visualizar el DOM la estructura en el archivo html con todos su nodos. Primero que nada en VSC instalamos la extensión live server y puedes ver que se nos abre el archivo HTML en nuestro browser, en la barra superior aparece un elemento, un número (<http://127.0.0.1:5500/index.html>) Estos números 127.0.0.1 representan una dirección IP local a continuación de los dos puntos 5500 representa el puerto que es como un código para identificar un proceso específico dentro del servidor local, es como una ubicación y es único para ese proceso y luego vemos donde estamos ubicados→ /index.html ¡si en el proyecto! En el archivo que está abierto en VSC que también nos avisa que el servidor se abrió en el puerto 5500. Y si necesitas volver a entrar a esa página puedes escribir la URL que se describe más arriba en azul. Esa es la parte fundamental de live server y luego si quieres detener el servidor haces click en la barra inferior de VSC en port 5500 y eso lo detiene. Ahora vamos a ver cómo acceder al DOM con las herramientas de desarrollo de Chrome. Hacemos click derecho en cualquier lugar de la página web se abre un menú y hacemos click en inspeccionar en la parte derecha por defecto se abre una ventana de herramientas. En el menú aparece por defecto una pestaña ELEMENTOS esa pestaña es la que se necesita para poder visualizar el DOM allí vamos a ver toda la estructura del archivo html como el DOM y su jerarquía o sea que ya estamos dentro de las herramientas de desarrollo de Chrome. Esta es la representación en el browser del DOM que es nos que interesa. Se puede ver toda la estructura html pero con algo interactivo agregado. Si posamos el cursor en html veremos que se resalta todo el documento. Ese es el NODO padre, dentro de html vemos head pero este no es visible como tal en la estructura asique si lo seleccionamos no lo podemos encontrar pero está allí y existe y luego aquí vemos en body lo que si podemos observar en la página y a mediada que nos internamos podemos ver cómo se va construyendo esa estructura del DOM y también dentro de estos elementos tenemos textos. Además de esto como estamos usando la extensión live server para actualizar nuestros cambios en vivo vemos los elementos de comentarios que ha sido agregado por esa extensión esto nos dice que si son parte del DOM si se reflejan en las herramientas de Chrome.

**CAMBIAR EL DOM:**

En las herramientas de Chrome podemos hacer cambios, si haces doble click en el texto de algún a los elemento puedes modificarlo y esto en realidad no es html es una versión interactiva del DOM puedes confirmarlo si haces click derecho se despliega un menú de opciones que nos permite editar el elemento como si fuera html. Veamos la de diferencia de editar texto o como html se recuerda que el texto se representa de forma individual como un Nodo separado del elemento como tal. Si editamos el texto automáticamente nos selecciona el texto y podemos modificarlo pero si seleccionamos editar como html nos selecciona todo el elemento, todo el nodo del Dom incluyendo el texto que contiene. <h1>texto</h1> y te permite modificar no solo el texto si no también la etiqueta en vez de h1 puedes llevarlo al valor que desees h2, h3, etc. Todo eso va a ser temporal, los cambios que hagas en las herramientas de Chrome son temporales hasta que vuelvas a cargar la página. Es muy útil para tener una vista previa de modificaciones que planeamos hacer en la página web. También podemos seleccionar o inspeccionar algún elemento en especial.

En el vértice superior izquierdo veremos una flechita tipo puntero que si hacemos click al seleccionarla se pondrá de color azul. El cursor se va a tornar como un cursor especial que nos va a permitir seleccionar un elemento en la página para inspeccionarlos y se podrá apreciar como el elemento o el nodo correspondiente en el DOM se va resaltando, eso es muy útil cuando ya tienes una página más elaborada y quieres inspeccionar algún elemento especifico. También tenemos otras herramientas interactivas como por ejemplo una consola donde se puede ejecutar código de JS para ver distintos resultados o para interactuar con los distintos elementos.

**Nodos que no son elementos:**

Veamos un poco más sobre los elementos que son Nodos pero no son como tal elemento html, mencionamos que el texto era uno de ellos y que los comentarios era uno de ellos. Si agregamos un comentario dentro del body

</--Comentarios--> Como tenemos la extensión live server activada se va actualizar automáticamente y en las herramientas de Chrome lo podremos ver el comentario como parte del DOM, no es un elemento html como el texto de h1 pero si es parte del DOM y se mantiene allí como un nodo, eso va a ser importante al momento de seleccionar los métodos que vamos a usar. Porque hay algunos que seleccionan todos los Nodos incluyendo comentarios y textos y hay otros que seleccionan comentarios html y es importante que se comprenda esa diferencia y también como estamos trabajando con live server puedes hacer cualquier cambio, lo guardas con ctrl + s y veras ese cambio reflejado en vivo en el DOM, también en las herramientas de Chrome tenemos la pestaña de styles en este caso aún no hemos agregado styles pero más adelante lo haremos.

**Proyecto: Toppings de Pizza**

**HTML**

**CSS**

/\* Estilos generales Css\*/

/\* Con el selector universal vamos asignar por defecto un margin: 0; y padding: 0; esto se recomienda y lo vas a ver comúnmente para que ningún elemento tenga un margin o padding asignado por defecto en el Browser pero cuando iniciamos esto de la forma mencionada estamos reseteando todo eso. También vas a encontrar esta propiedad box-sizing: border-box; esto le dice al Browser como va a calcular el tamaño en base a su margin y a su padding, border-box es una de las propiedades específicas de los valores específicos. Y luego el tipo de fuente font-family: "lato", sans-serif;

\*/

\* {

} \*/

/\*Selector de tipo \*/