Tutorial
HTML - Javascript - DOM - 2021-2

Ph.D. Carlos Cares
carlos.cares@ceisufro.cl

Este es un tutorial del curso de Programación Avanzada ICC343 de la Universidad de La Frontera. Es un ejercicio práctico donde se entregan y practican elementos teóricos. En este caso revisaremos cómo manipular elementos HTML desde código incrustado en Javascript en una página web. Si usted sigue el tutorial fuera del ámbito del curso ICC343 y tiene comentarios o preguntas hágalas directamente al email indicado, estaré encantado de contestar.

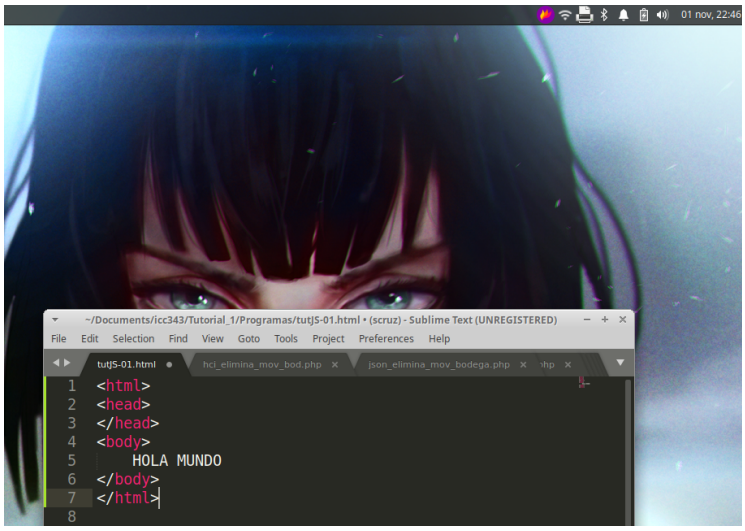
Requerimientos:

- Un editor de texto, idealmente un editor de desarrollo de software. En las imágenes de este tutorial se usa Sublime, pero Visual Code, BlueJ, o cualquier otro indicado aquí no debería presentar ningún problema.
- Un navegador de Internet. En este tutorial las imágenes son de Chrome, pero "casi" cualquier navegador contemporáneo debería andar bien.
- Este tutorial se puede seguir en cualquier plataforma, Linux, Windows, o Mac.

La evaluación del tutorial. El tutorial cuenta con 10 partes, cada una evaluada en un 10%. Cada entrega debe hacerse en el canal privado de Slack del curso **PEGANDO (Crtl-V) las imágenes pedidas como producto** (prefiera pegar a subir un archivo) cuando se pidan *screenshots*. Las preguntas y respuestas se hacen en el canal público. De manera **PREVIA** a pegar su foto o subir archivo de código, **DEBE** escribir la etiqueta que identifica ese resultado. Cada foto de pantalla (screenshot) debe hacerse con las siguientes características:

Características de los screenshots:

- Deben ser legibles
- Deben incluir una porción del fondo del escritorio
- Deben incluir la parte del escritorio que muestra fecha y hora
  Ejemplo:



Parte 1. (10%). Estructura y Títulos en HTML

Elementos HTML. HTML es un lenguaje de marcas especificado en texto. Lo que hace un navegador es interpretar estas marcas y producir una salida gráfica, es decir, el navegador actúa como un intérprete, no compila ni produce código ejecutable. La interpretación del navegador ocurre en la máquina que tiene instalado el navegador, NO en el servidor web, de este modo, todo lo que es HTML, y la manipulación javascript de los elementos HTML ocurre en el navegador de la máquina cliente, es decir, para este tutorial, no necesitamos un servidor web activo, ya que la mayoría de los navegadores pueden interpretar archivos HTML almacenados en su propio disco.

Los elementos HTML se marcan con etiquetas y las etiquetas pueden tener atributos. Un excelente resumen de los elementos HTML los puede encontrar aquí, en la escuela del W3C.

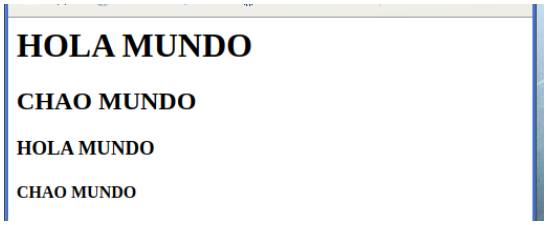
Paso 1-1.
Un “programa” HTML, es básicamente una declaración de lo que queremos visible en una página web, se compone de un bloque principal indicado por la etiqueta <HTML> y su fin </HTML>, el cual puede tener un encabezado, indicado por la etiqueta <HEAD> y su fin </HEAD>, y tiene un cuerpo indicado por la etiqueta <BODY> y su fin </BODY>.

Copie el siguiente ejemplo para revisar esta estructura, que utiliza además las etiquetas <H1> a <H4> para títulos de diferente tamaño dentro de la misma página. Además hemos usado la etiqueta <TITLE> en el encabezado, que es la que muestra el navegador en sus pestañas.

tutJS-01.html

```
<html>
<head>
    <title>Tutorial HTML-JavaScript</title>
</head>
<body>
    <h1>HOLA MUNDO</h1>
    <h2>CHAO MUNDO</h2>
    <h3>HOLA MUNDO</h3>
    <h4>CHAO MUNDO</h4>
</body>
</html>
```

Paso 1-2
En un ambiente de ventanas como Mac, Windows o Linux lo más sencillo es arrastrar el archivo creado sobre el navegador. Hágalo y debería ver lo siguiente.



Paso 1-3 Reporte su resultado con los dos screenshots anteriores pero siguiendo los requerimientos de screenshots indicados.

IDENTIFIQUE CON LAS SIGUIENTES ETIQUETAS los productos pedidos en este paso:

Tu1-P1-A:	Código HTML en su editor de texto
Tu1-P1-B:	Foto de la salida en el navegador
Tu1-P1-C:	Comentario en canal privado sobre el significado de las etiquetas H1, H2, H3, y H4.

Parte 2. (10%). Tablas y Estilos en HTML

Una tabla en HTML, indicada por la etiqueta <TABLE> es una estructura que básicamente tiene filas, indicadas por la etiqueta <TR>, y celdas dentro de las filas, indicadas por etiquetas <TD> anidadas dentro de las etiquetas <TR>. He dicho "básicamente" porque puede indicarse un encabezado de tabla y un cuerpo de tabla con otras etiquetas que ahora no usaremos.

La mayoría de las etiquetas HTML aceptan el atributo style, donde se indica, como un listado diferente de atributos, el "estilo" de una etiqueta específica. Los nombres de atributos y valores de estilos conforman **OTRO** lenguaje de especificación conocido como CSS (Cascade Style Sheest) y una buen referencia rápida se encuentra en https://www.w3c.es/Divulgacion/GuiasReferencia/CSS21/. No es necesario que visite esta página ahora, el ejemplo proveerá de su uso básico.

Paso 2-1 Copie el siguiente programa

```
1  <html>
2  <head>
3      <title>Tutorial HTML-JavaScript</title>
4  </head>
5  <body>
6      <h4>Mi Tabla</h4>
7      <table
8          border="2px"
9          style="background-color: #90CC90;
10                 font-family: courier;
11                 font-size: 9">
12         <tr>
13             <td>Hola</td>
14             <td>Mundo</td>
15         </tr>
16         <tr>
17             <td colspan="2">
18                 Chao Mundo
19             </td>
20         </tr>
21     </table>
22 </body>
23 </html>
```

La línea 7 podría continuar con todo el contenido. Aquí se ha separado con el fin que sus elementos se diferencien. Note cómo el contenido del atributo style es complejo, pues son varios atributos que hemos cambiado, el color de fondo y el tipo y tamaño de letra. Lo otro nuevo que se ha agregado es el uso de atributos propios en la etiqueta <TABLE> hemos usado un borde de 2 pixels (atributo border) y en el caso de la segunda fila (segundo <tr>) hemos agregado sólo una celda pero que ocupa dos lugares, lo que es indicado por el atributo “colspan”. El resultado debería ser como se indica en la siguiente imagen

Paso 2-2 Reporte su resultado con los dos screenshots anteriores pero siguiendo los requerimientos de etiquetas que se indican.

Tu1-P2-A: Código HTML en su editor de texto
Tu1-P2-B: Foto de la salida en el navegador

Parte 3 (10%) La etiqueta <style>

La etiqueta <style> permite nombrar un conjunto de estilos con un solo nombre y nominalizar (dar un nombre) a este conjunto. El conjunto se puede usar posteriormente en más de un elemento usando el atributo class, admitido por la mayoría de las etiquetas HTML.

Paso 3-1. Refactorizamos el ejemplo anterior separando el estilo pero manteniéndolo incrustado en la misma página. Copie el programa en su nueva versión tal como se indica (se recomienda ejecutar un “grabar como” para que mantenga las dos versiones).

```
1  <html>
2  <head>
3      <title>Tutorial HTML-JavaScript</title>
4      <style>
5      .mi_tverde {
6          background-color: #90CC90;
7          font-family: courier;
8          font-size: 9
9      }
10     .mi_tverde_claro {
11         background-color: #90F590;
12     }
13     </style>
14 </head>
15 <body>
16     <h4>Mi Tabla</h4>
17     <table border="2px" class="mi_tverde">
18     <tr>
19         <td>Hola</td>
20         <td class="mi_tverde_claro">Mundo</td>
21     </tr>
22     <tr>
23         <td colspan="2">
24             Chao Mundo
25         </td>
26     </tr>
27     </table>
28 </body>
29 </html>
```

Note que en el lenguaje de estilos un nombre nuevo se comienza con “.” (línea 5 y 10), los nombres de los estilos se usan en el atributo class (línea 17 y 20). Por último, note que los atributos de estilo se heredan, pues la celda verde claro, mantiene el estilo de la tabla en cuanto a tipo y tamaño de letra.

Tu1-P3-A: Código HTML extendido en su editor de texto
Tu1-P3-B: Foto de la salida en el navegador de la nueva tabla.

Parte 4 (10%) Hojas de estilo separadas vía etiqueta <LINK>

Para una mayor legibilidad del código se recomienda separar las hojas de estilo (archivos de extensión css) del código HTML. De este modo se pide que cree un archivo con SU nombre como nombre de estilo, yo usaré “carlos.css” y usted usará su propio nombre. Así nos quedan dos archivos, el css y el html.

Paso 4-1 Copie las modificaciones del programa.

```
1  <html>
2  <head>
3      <title>Tutorial HTML-JavaScript</title>
4      <link rel="stylesheet" type="text/css" href="./carlos.css">
5  </head>
6  <body>
7      <h4>Mi Tabla</h4>
8      <table border="2px" class="mi_tverde">
9      <tr>
10         <td>Hola</td>
11         <td class="mi_tverde_claro">Mundo</td>
12     </tr>
13     <tr>
14         <td colspan="2">
15             Chao Mundo
16         </td>
17     </tr>
18     </table>
19 </body>
20 </html>
```

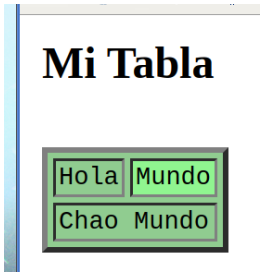
Note que el atributo href en la línea 4 tiene un “.” delante, esto es porque estamos usando la interpretación de archivos en lugar de sitios web. Si usted está usando su propio servidor web entonces no necesita este punto ni el slash. El archivo de estilos no tiene cambios y quedó de la siguiente forma

Paso 4-2 Copie usando su propio nombre como archivo de estilo lo siguiente:

tutJS-02.html × carlos.css × hci_elimina_mov_bod.php

```
1
2  .mi_tverde {
3      background-color: #90CC90;
4      font-family: courier;
5      font-size: 9
6  }
7  .mi_tverde_claro {
8      background-color: #90F590;
9  }
```

El producto debería ser lo mismo que en el caso anterior. Pero usted notará que los contenidos quedan ordenados. Esto es una práctica recomendada.



Tu1-P4-A: Código CSS en su editor de texto
Tu1-P4-B: Código HTML con el link al CSS
Tu1-P4-C: Salida en el navegador actualizada

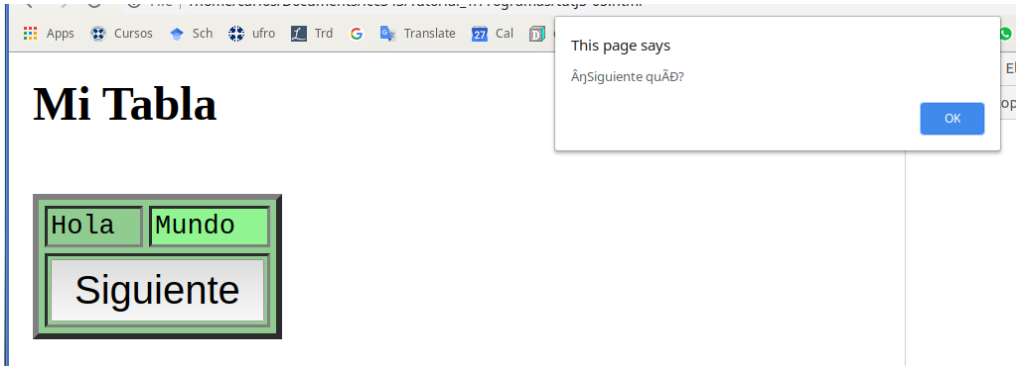
Parte 5 (10%). Incrustando Javascript con la etiqueta <SCRIPT>

Javascript es un lenguaje de script que se puede incrustar en una página web. Se interpreta en el cliente, es decir, en el navegador. En el siguiente ejemplo creamos un botón con HTML y gatillamos (trigger) una función javascript. Hay dos formas de crear botones en HTML, por medio de la etiqueta <INPUT> con atributo TYPE=BUTTON, o usamos una etiqueta introducida posteriormente llamada <BUTTON>. He visto que es más flexible button, en Macintosh los navegadores no respetan los estilos impuestos en las etiquetas de <INPUT>, lo que afecta el botón del input, porque priorizan el estilo del sistema operativo. Sin embargo, el estilo de Button es respetado.

En este ejemplo creamos una función javascript que emite una ventana de alerta.

Paso 5-1. Agregando el código Javascript. Copie y pruebe el siguiente código.

```
<html>
<head>
    <title>Tutorial HTML-JavaScript</title>
    <link rel="stylesheet" type="text/css" href="./carlos.css">
    <script>
        function quepasa() {
            alert("¿Siguiente qué?");
        }
    </script>
</head>
<body>
    <h4>Mi Tabla</h4>
    <table border="2px" class="mi_tverde">
    <tr>
        <td>Hola</td>
        <td class="mi_tverde_claro">Mundo</td>
    </tr>
    <tr>
        <td colspan="2">
            <button type="button" onclick="quepasa()">Siguiente
            </button>
        </td>
    </tr>
    </table>
</body>
</html>
```



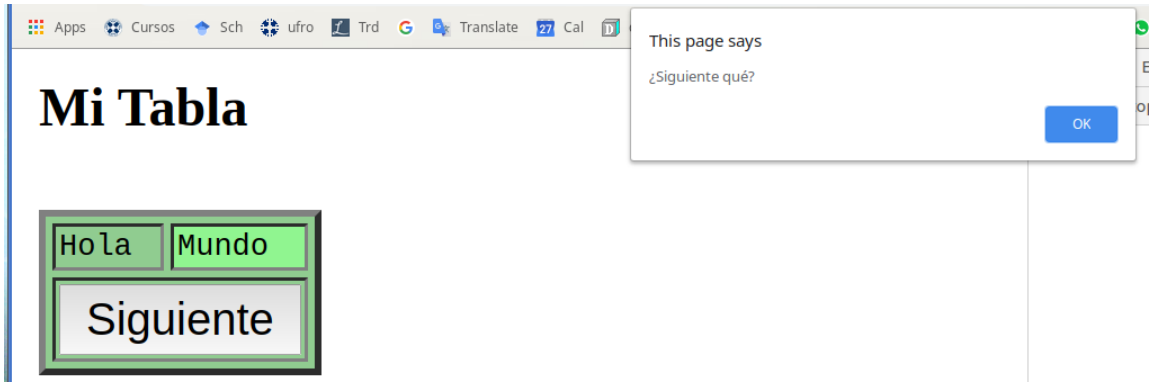
Primero note que el botón no asume el estilo de la tabla. Esto es porque, estructuralmente el botón sólo está como contenido de la tabla, una fila y una celda, sin embargo, son parte estructural de la tabla. Segundo, note que al hacer clic en el botón siguiente aparece la ventana emergente. Note que el mensaje de la ventana no ha respetado los símbolos de interrogación españoles, esto es porque no hemos especificado en qué set de caracteres trabajaremos. Es posible que su sistema operativo sí arroje el mensaje correcto, pero esto es sólo temporal y presentará problemas dependiendo de quién sea el visitante del sitio web.

Paso 5-2. Especificando el conjunto de caracteres

El conjunto de caracteres se especifica con la etiqueta <META> en el encabezado. El encabezado ha quedado de esta forma:

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Tutorial HTML-JavaScript</title>
    <link rel="stylesheet" type="text/css" href="./carlos.css">
    <script>
        function quepasa() {
            alert("¿Siguiente qué?");
        }
    </script>
</head>
```

Y la salida ha quedado ahora como se indica



Tu1-P5-A: Código HTML incluyendo el código Javascript
Tu1-P5-B: Salida en el navegador de la tabla y el mensaje de texto
Tu1-P5-C: Agregue un comentario respecto si le sucedió o no que el primer mensaje de alerta se realizó sin respetar los códigos y si efectivamente la etiqueta <META> soluciona el problema.

Parte 6 (10%). Accediendo a valores HTML

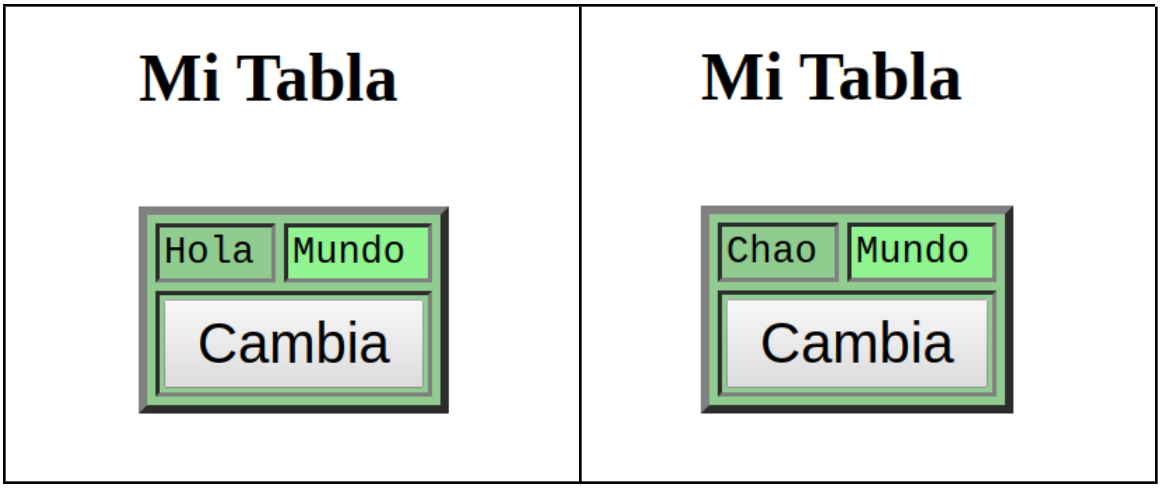
Javascript permite acceder y controlar los elementos HTML en la página web. Siempre existe el objeto `document`, el cual contiene métodos y atributos usables de todo el documento. La manera más segura de obtenerlos es por medio del uso de identificadores únicos expuestos en el atributo ID. En esta parte del tutorial cambiamos el contenido de la celda de la tabla que dice Hola, por lo tanto, en términos de HTML, le agregamos un identificador a esta celda para poder manipularla.

Paso 6-1. Accediendo al contenido HTML de una celda.

En este paso usamos el método de document getElementById, el cual nos devuelve el objeto que tiene dicho identificador (id) Por ese motivo la celda que dice “Hola” le hemos puesto un id cualquiera (“ele100” en la línea 20). Una vez capturado el objeto accedemos a su contenido mediante la propiedad innerHTML. En el método usamos una estructura condicional (if-else) para alternar entre los valores “Hola” y “Chao”. Cuidado que el nombre de la función fue cambiado a “holachao”. Tenga cuidado al copiar el programa porque javascript hace diferencia entre mayúsculas y minúsculas. Fíjese que las instrucciones son similares a java, pero no es un lenguaje fuertemente tipeado como Java (strongly typed) porque las variables pueden ser de cualquier tipo.

```html
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Tutorial HTML-JavaScript</title>
    <link rel="stylesheet" type="text/css" href="./carlos.css">
    <script>
        function holachao() {
            celda = document.getElementById('ele100');
            if(celda.innerHTML=='Hola')
                celda.innerHTML='Chao';
            else
                celda.innerHTML='Hola';
        }
    </script>
</head>
<body>
    <h4>Mi Tabla</h4>
    <table border="2px" class="mi_tverde">
    <tr>
        <td id="ele100">Hola</td>
        <td class="mi_tverde_claro">Mundo</td>
    </tr>
    <tr>
        <td colspan="2">
            <button type="button" onclick="holachao()">Cambia</button>
        </td>
    </tr>
    </table>
</body>
</html>
```

Haciendo clic en el botón “Cambia”, la salida alterna entre las dos imágenes siguientes



Tu1-P6-A: Código HTML incluyendo el código Javascript
Tu1-P6-B: Salida en el navegador de la tabla ANTES Y DESPUÉS del clic

Parte 7 (20%). Creando objetos HTML desde Javascript y revisando la Consola.

Usted se ha fijado que en el ejercicio anterior hemos dejado la función javascript en el encabezado HTML. En general se usa dejar el javascript separado, pero si el javascript va a estar en el mismo documento, una mejor práctica es dejar las funciones en el encabezado. El script que se agrega puede contener no sólo funciones sino también invocaciones e instrucciones de control que se ejecutarán a medida que se despliega la página web. En este ejercicio lo que se hace es que se creará una tabla desde el código javascript.

Paso 7-1. Copie y estudie el siguiente código

```
1  <html>
2  <head>
3      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4      <title>Tutorial HTML-JavaScript</title>
5      <link rel="stylesheet" type="text/css" href="./carlos.css">
6      <script>
7          function creaTablaHolaMundo() {
8              var tabla = document.createElement("TABLE");
9              tabla.setAttribute("class","mi_tverde");
10             tabla.border = "2px";
11             var fila = document.createElement("TR");
12             var celd = document.createElement("TD");
13             celd.innerHTML="Hola";
14             celd.id ="ele100";
15             fila.appendChild(celd);
16             celd = document.createElement("TD");
17             celd.innerHTML="Mundo";
18             fila.appendChild(celd);
19             tabla.appendChild(fila);
20             var b = document.getElementsByTagName("BODY");
21             b[0].appendChild(tabla);
22         }
23     </script>
24 </head>
25 <body>
26     <h4>Mi Tabla</h4>
27     <button type="button" onclick="holachao()">Cambia</button>
28     <script>
29         creaTablaHolaMundo();
30     </script>
31 </body>
32 </html>
```

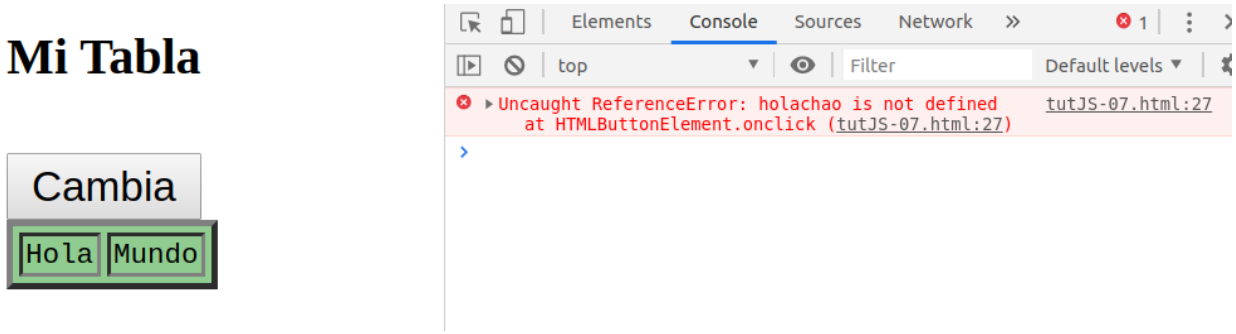
Primero, mantenemos en el encabezado una función javascript. Usamos el método `createElement`, para crear, EN MEMORIA, un objeto HTML, es decir este objeto no es parte del documento hasta que se agrega a otro elemento que esté en el documento. Para agregar un elemento tenemos que tener el objeto contenedor, el cual puede agregar su elemento contenido usando el método `appendChild`. Para crear el objeto se usa su nombre de etiqueta (TAG). Revise cómo se crea una tabla (línea 8), se crea una fila (línea 11) y se crean sus celdas (líneas 12 y 16). No es necesario usar variables diferentes para las celdas porque, básicamente la variable es un puntero al objeto, entonces la fila agrega su primera celda (línea 15) y luego entonces se puede re-usar la

variable `celd` porque la estructura (la fila) ya sabe su primera celda. Finalmente, tenemos toda la tabla armada, pero se necesita ligar al documento HTML, entonces, usamos el método `getElementsByTagName` (línea 20). Este método SIEMPRE, devuelve una lista de los objetos que tienen el nombre “BODY” (arreglo en realidad), sabemos que no puede haber más de uno en el caso de “BODY”, pero no importando aquello, el método devuelve una lista. La lista podría estar vacía, o podría tener muchos elementos. Como estamos seguro que sí tiene un elemento, entonces accedemos a su elemento 0 (el primero) y a ese elemento, el único body, agregamos la tabla creada.

También merece comentario la línea 9. En esta línea usamos el método `setAttribute`, y sin embargo en la línea 10 y en la línea 14 accedemos directamente a los atributos `border` e `id` respectivamente. Lo que sucede es que el atributo `class`, no tiene una “existencia previa” o no es un atributo estándar (para el navegador específico).

Paso 7-2 Revisando la Consola

La salida será como el lado izquierdo de la siguiente figura.



Presione el botón y verá que no cambiará el contenido como antes. Sobre el navegador use el botón derecho y elija la opción inspeccionar, se abrirá una ventana lateral si está en Chrome o en la parte inferior si está en Firefox. Seleccione la pestaña “Console”. Debería ver el error de la figura. Lo que ha sucedido es que la función `holachao` no la hemos copiado en esta versión .Cópiela de la versión anterior y compruebe que el botón funciona como antes.

Tu1-P7-A:	Primer código HTML incluyendo el código Javascript erróneo
Tu1-P7-B:	Salida en el navegador de la tabla y el mensaje de error en la consola
Tu1-P7-C:	Código HTML/Javascript corregido (función vacía con ventana de alert)
Tu1-P7-D:	Salida en el navegador de la tabla, ventana de inspección abierta sin el error e incluyendo mensaje emergente.

Parte 8 (20%). Creando objetos HTML a partir de arreglos

En este ejemplo tenemos una función que agrega una tabla de códigos de colores y colores de acuerdo a un arreglo. Al arreglo le podemos agregar más datos vía un campo de entrada de texto, esto es <INPUT> con atributo TYPE=TEXT. También hacemos algo diferente al caso anterior, pues al agregar elementos adicionales al BODY siempre se agregará al final, y a veces podríamos querer agregar en una parte diferente del documento. Entonces usamos una sección, identificada por el elemento <DIV>. Entonces volveremos al despliegue inicial con el botón al final.

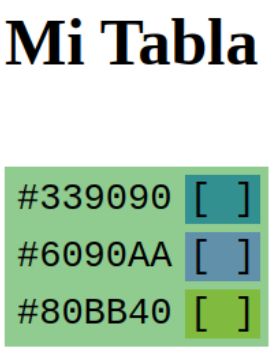
Paso 8-1. Generación de la Tabla de Colores

En este paso sólo generamos la tabla de colores, el código es el siguiente

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Tutorial HTML-JavaScript</title>
    <link rel="stylesheet" type="text/css" href="./carlos.css">
    <script>
        var col = ["#339090","#6090AA","#80BB40"]
        function generaTablaColor(idDiv,listaCol) {
            var tabla,fila,celd,i;
            tabla = document.createElement("TABLE");
            tabla.setAttribute("class","mi_tverde");
            for(i=0; i<listaCol.length ; i++) {
                fila = document.createElement("TR");
                celd = document.createElement("TD");
                celd.innerHTML=listaCol[i];
                fila.appendChild(celd);
                celd = document.createElement("TD");
                celd.innerHTML='[   ]';
                celd.style='background-color:'+listaCol[i];
                fila.appendChild(celd);
                tabla.appendChild(fila);
            }
            document.getElementById(idDiv).innerHTML="";
            document.getElementById(idDiv).appendChild(tabla);
        }
    </script>
</head>
<body>
    <h4>Mi Tabla</h4>
    <div id="aqui"></div>
    <script>
        generaTablaColor("aqui",col);
    </script>
</body>
</html>
```

Primero que nada fíjese que la función es genérica. Recibe dos parámetros, el primero es el elemento que contendrá la tabla y el segundo es el arreglo de colores. Es importante declarar las variables así su “scope” queda reducido al ámbito local, es decir la función. La creación es como antes con la única diferencia que tenemos un ciclo `for`

para iterar sobre cada color en el arreglo. Lo que además hace la función, es que borra el contenido HTML de dicho contenedor para dejar sólo a la tabla (línea 23). Finalmente en la línea 30 se agrega la sección, por ahora vacía que contendrá la tabla de colores. La invocación de la línea 32 usa el arreglo de colores declarado al principio (es global). El resultado hasta aquí debería ser el siguiente:



Paso 8-2. Usando un campo de entrada de datos.

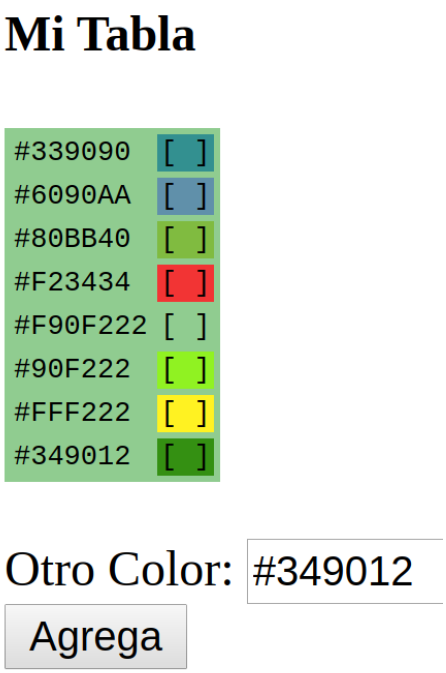
Lo primero que agregamos en esta versión extendida es la función que agrega el color. En realidad, no es mucho lo que hay que hacer pues la tabla se genera en base al listado de colores, entonces simplemente, agregamos a la lista el color ingresado. Algunos supuestos de base es que el campo de entrada tiene identificador “otromas”. Además, fíjese que en javascript los arreglos tienen métodos y propiedades asociadas, antes ya usamos la propiedad `length`, para saber la cantidad de elementos del arreglo, y ahora usamos el método `push`, para agregar un elemento en el arreglo.

```
26          function agregaColor() {
27              var valor = document.getElementById("otromas").value;
28              col.push(valor);
29              generaTablaColor("aqui",col);
30          }
31      </script>
```

En la parte de HTML agregamos tanto el campo de entrada como el botón para invocar esta nueva función. Esto queda así

```
33  <body>
34      <h4>Mi Tabla</h4>
35      <div id="aqui"></div>
36      <script>
37          generaTablaColor("aqui",col);
38      </script>
39      <br>
40      Otro Color: <input type="text" id="otromas">
41      <button onclick="agregaColor()">Agrega</button>
42  </body>
```

De este modo la salida esperada es como la que se indica, por supuesto después de haber agregado algunos colores adicionales. Fíjese que ante el error, el color asignado fue el del estilo previo.



Tu1-P8-A:	Código HTML incluyendo el código Javascript
Tu1-P8-B:	Salida en el navegador de la tabla antes de ingresar color adicional
Tu1-P8-C:	Salida en el navegador después de agregar 3 colores adicionales.

Ejercicio Sugerido. Note que es ineficiente cada vez generar toda la tabla desde el comienzo. Intente una versión donde sólo se agrega una fila a la tabla cada vez.