3. ESTANDARES WEB (CLIENTE)

Materia: Programación para Internet
Alumno: Luis Miguel Navarro Jr
Ciclo: 2021A
Profesor: Dr. José Sandoval Chávez

3.1 HTML, CSS y Javascript

3.1.1 Resumen, introducción a CSS

El lenguaje HTML está limitado a la hora de aplicarle forma a un documento. Esto es así porque fue concebido para otros usos (científicos sobre todo), distinto a los actuales, mucho más amplios. Para solucionar estos problemas los diseñadores han utilizado técnicas tales como la utilización de tablas, imágenes transparentes para ajustarlas, utilización de etiquetas que no son estándares del HTML y otras. Estas "trampas" han causado a menudo problemas en las páginas a la hora de su visualización en distintas plataformas.

3.1.2 Ventajas y desventajas de CSS

3.1.2.1 Ventajas

1. Un documento HTML o página, se puede definir la forma, en un pequeño trozo de código en la cabecera, a toda la página.

2. Una etiqueta en concreto, llegando incluso a poder definir varios estilos diferentes para una sola etiqueta. Esto es muy importante ya que ofrece potencia en la Programación.

3.1.2.2 Desventajas

1. No todos los navegadores la soportan, si hay una actualización podría haber errores de compatibilidad y cambiar el diseño general de la pagina web
2. Hay muchas cosas en CSS qu3 se pueden hacer en JavaScript y viceversa, lo cual puede llegar a diferencias creativas entre diseñadores.

3.1.3 Navegadores y que versionas soportan de CSS

La mayoría de los navegadores principales de mercado tienen soporte para el 100% de las integraciones de CSS en sus versiones mas recientes, incluidas Mozilla Firefox, Google Chrome y Brave algunos de los ejemplos mas reconocidos, en cambio, a continuación una lista de navegadores ampliamente utilizados que no tienen una compatibilidad completa o incluso carecen de soporte, y aun así, son navegadores frecuentes en algunos segmentos de la población.

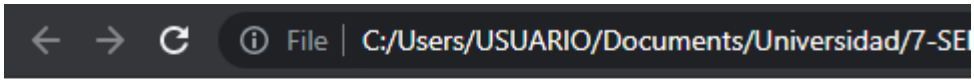
Technology	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
CSS 3 changes					
CSS 3 Units	75%	75%	80%	82%	79%
CSS 3 At-rules	N	N	N	N	26%
CSS 3 Basic selectors	N	50%	63%	63%	63%
CSS 3 Pseudo-classes	N	N	25%	30%	20%
CSS 3 Pseudo-elements	N	N	N	N	N
CSS 3 Basic properties	14%	14%	18%	19%	5%
CSS 3 Print properties	N	N	N	N	N

3.1.4 Dar estilos CSS

3.1.4.1 Estilo para pequeñas partes de una pagina

Esta manera de dar estilo a secciones HTML es buena si se desea que sea algo extremadamente especifico, pero debería de evitarse si es un cambio muy general para toda la pagina.

```
<p>
Esto es un párrafo en varias palabras <SPAN style="color:red">
en color rojo </SPAN> resulta muy fácil.
</p>
```

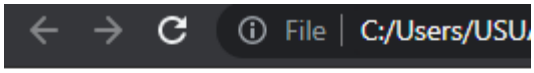


Esto es un párrafo en varias palabras en color rojo resulta muy fácil.

3.1.4.2 Estilo por etiquetas

Esta manera sigue siendo individual para ofrecer estilos, comúnmente se usa con la etiqueta <DIV> para dar estilo a grandes fragmentos de código, pero en esta ocasión solo es a párrafos de texto.

```
<p style="color:yellow">
Esto es un párrafo de color amarillo.
</p>
<p style="color:green">
Esto es un párrafo de color verde.
</p>
```



Esto es un párrafo de color amarillo.

Esto es un párrafo de color verde.

3.1.4.3 Estilo definido a una parte de la página.

Con la etiqueta <DIV> podremos dar estilo a una parte de la página, pudiendo introducir muchas etiquetas <p> y que todas tengan el mismo estilo mientras estén dentro de la fracción correspondiendo, afectando también otras etiquetas como las <H1> <H2> …

```
<div style="color:red; font-weight:bold">
<h3> Estas etiquetas van en <i>rojo y negrita</i></h3>
<p>
Seguimos dentro del DIV, luego permanecen los etilos
</p>
</div>
```

Estas etiquetas van en *rojo y negrita*

Seguimos dentro del DIV, luego permanecen los etilos

3.1.5 Usos más avanzados de CSS

3.1.5.1 Estilo definido para una página completa

De esta manera podemos definir una generalidad de estilo para toda la página, a pesar de que no está puesta en un fichero, y seria mas complicada su exportación a otros directorios de la página web, es ideal para definir los estilos de una pagina.

```
<STYLE type="text/css">
<!--
H1 {text-decoration: underline; text-align:justify}
P {font-Family:arial,verdana; color: white; background-color: green}
BODY {color:black;background-color: grey; text-indent:2cm}
// -->
</STYLE>
```

3.1.5.2 Estilo definido para todo un sitio web

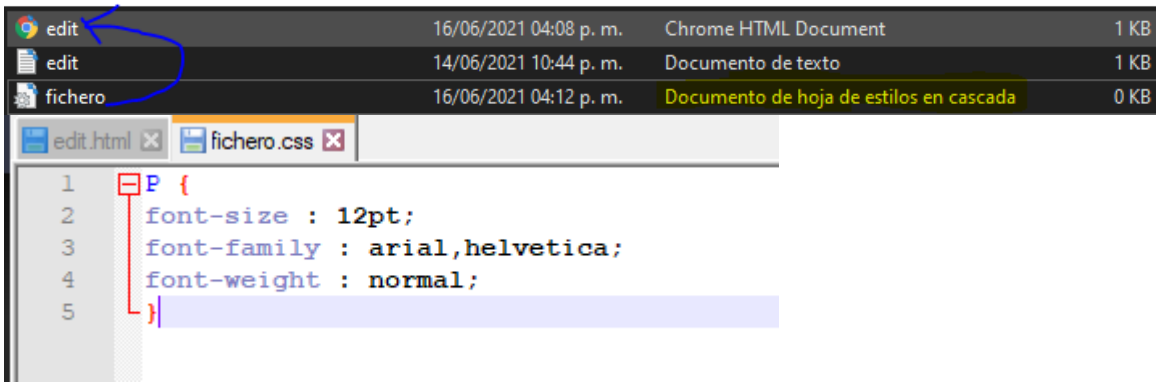
De esta forma podemos exportar un solo estilo para todo el sitio en general, con el uso de ficheros, es la manera perfecta de administrar todo nuestro proyecto sin definir muchísimas líneas de código muchas veces.

-> Creamos un fichero de extencion ".CSS" y le damos un estilo como (Escrito esto dentro del documento):

```
P {
font-size : 12pt;
font-family : arial,helvetica;
font-weight : normal;
}
```

Y lo enlazamos al documento principal con:

```
<link rel="STYLESHEET" type="text/css" href="estilos.css">
```



3.1.5.3 Reglas de importancia en los estilos

Los estilos se heredan de una etiqueta a otra, como se indicó anteriormente. Por ejemplo, si se tiene declarado en el <BODY> unos estilos, por lo general, estas declaraciones también afectarán a etiquetas que estén dentro de esta etiqueta, o lo que es lo mismo, dentro de todo el cuerpo.

 Declaración de estilos con fichero externo. (Para todo un sitio web)

 Declaración de estilos para toda la página. (Con la etiqueta <STYLE> en la cabecera de la página)

 Estilos definidos en una parte de la página. (Con la etiqueta <DIV>)

 Definidos en una etiqueta en concreto. (Utilizando el atributo style en la etiqueta en cuestión)

 Declaración de estilo para una porción pequeña del documento. (Con la etiqueta <SPAN>)

3.1.5.4 Otra manera de definir estilos en una archivo externo

```
<style type="text/css">
@import url ("estilo.css");
body{
background-color:#ffffcc;
}
</style>
```

Si declaramos la etiqueta BODY, es porque le damos un estilo para la página, pero todos los demás estilos serán heredados.

3.1.6 Sintaxis de CSS

Para ficheros externos:

<etiqueta> { //ESTILOS }

Para una pagina en especifico:

<STYLE> //Etiqueta{//atributos definidos} </STYLE>

En una etiqueta:

<etiqueta style = “atributos” > </etiqueta>

Fragmento especifico:

<etiqueta> //texto //TEXTO AFECTADO POR ESTILOS //texto no afectado </etiqueta>

3.1.7 Javascript

3.1.7.1 ¿Qué es Javascript?

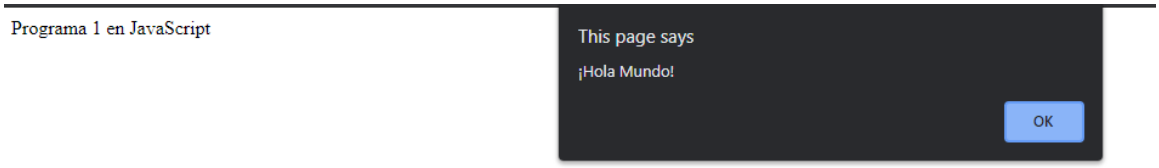
JavaScript es un lenguaje de programación creado por la empresa Netscape (creadora de uno de los navegadores más conocido)
Es el lenguaje de programación más utilizado en Internet para añadir interactividad a las páginas Web

3.1.7.2 ¿De qué manera se escapa Javascript?

```
<SCRIPT LANGUAGE="JavaScript">
alert("¡Hola Mundo!");
</SCRIPT>
```

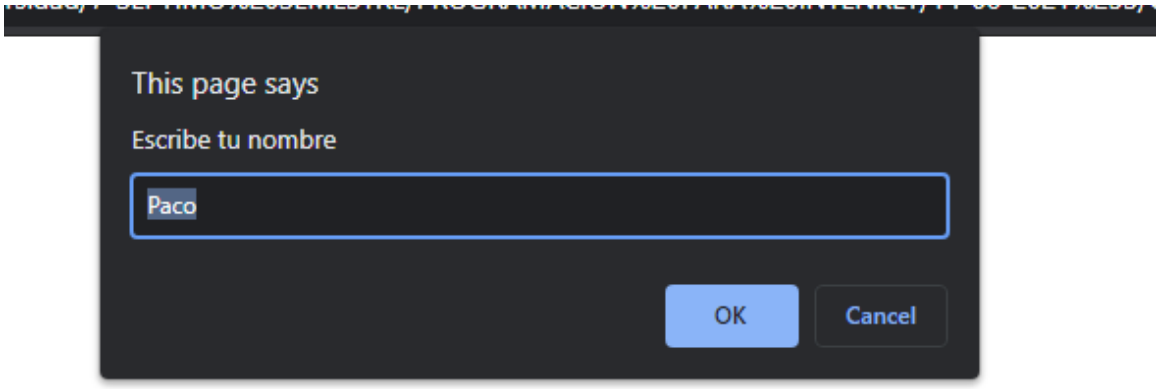
3.1.7.3 Ejemplo de uso de alert, prompt y confirm:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
alert("¡Hola Mundo!");
</SCRIPT>
</HEAD>
<BODY>
<P>
Programa 1 en JavaScript
</P>
</BODY>
</HTML>
```



<HTML>

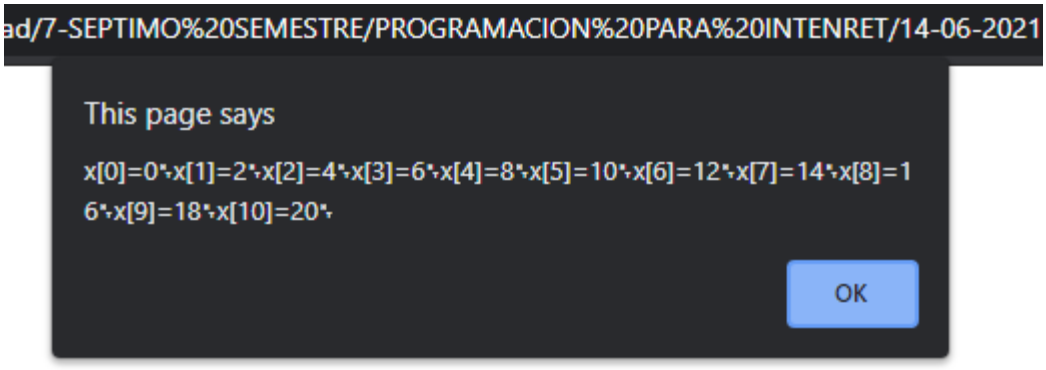
```
<SCRIPT LANGUAGE="JavaScript">
var nom;
nom=prompt("Escribe tu nombre ","Paco");
alert("Mucho gusto "+ nom);
</SCRIPT>
</HTML>
```



3.1.7.4 Uso de arreglos

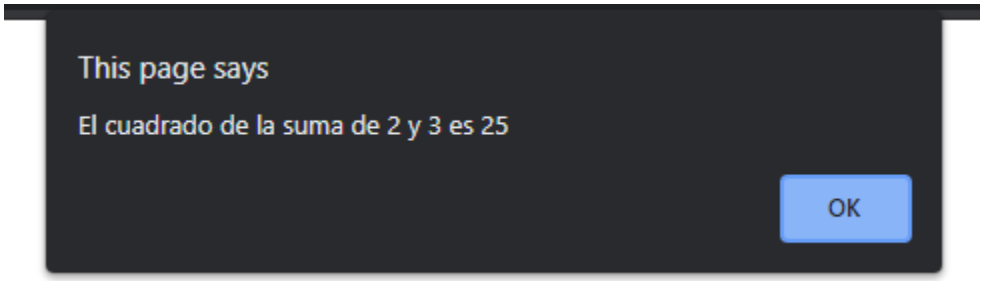
```
<HTML>
<SCRIPT LANGUAGE="JavaScript">
var x=new Array();
var salida="";
for(i=0;i<=10;i++)
{
x[i]=2*i;
salida=salida+"x["+i+"]="+x[i]+"\t";
}

alert(salida);
</SCRIPT>
</HTML>
```



3.1.7.5 Funcion con parámetros y retorno

```
<HTML>
<SCRIPT LANGUAGE="JavaScript">
/* Programa Principal */
var x,y,result;
x=prompt("Escribe el primer sumando","");
x=parseFloat(x);
y=prompt("Escribe el segundo sumando","");
y=parseFloat(y);
/* A continuación llamo a la función para calcular
el cuadrado de la suma de x e y */
result=SumaCuadrado(x,y);
alert("El cuadrado de la suma de "+ x +" y "+ y +" es "+result);
// Fin del programa principal
/* A continuación tengo la definición de la función */
function SumaCuadrado(a,b)
{
return (a*a+b*b+2*a*b);
}
</SCRIPT>
</HTML>
```



3.2 AJAX con JQuery

jQuery (http://jquery.com/) es la librería JavaScript que ha irrumpido con más fuerza como alternativa a Prototype.
Como sucede con Prototype, las funciones y utilidades relacionadas con AJAX son parte fundamental de jQuery. El método principal para realizar peticiones AJAX es $.ajax() (importante no olvidar el punto entre $ y ajax). A partir de esta función básica, se han definido
otras funciones relacionadas, de más alto nivel y especializadas en tareas concretas: $.get(),
$.post(), $.load(), etc.
La sintaxis de $.ajax() es muy sencilla:
$.ajax(opciones);

Opción	Descripción
`async`	Indica si la petición es asíncrona. Su valor por defecto es `true`, el habitual para las peticiones AJAX
`beforeSend`	Permite indicar una función que modifique el objeto `XMLHttpRequest` antes de realizar la petición. El propio objeto `XMLHttpRequest` se pasa como único argumento de la función
`complete`	Permite establecer la función que se ejecuta cuando una petición se ha completado (y después de ejecutar, si se han establecido, las funciones de `success` o `error`). La función recibe el objeto `XMLHttpRequest` como primer parámetro y el resultado de la petición como segundo argumento
`contentType`	Indica el valor de la cabecera `Content-Type` utilizada para realizar la petición. Su valor por defecto es `application/x-www-form-urlencoded`
`data`	Información que se incluye en la petición. Se utiliza para enviar parámetros al servidor. Si es una cadena de texto, se envía tal cual, por lo que su formato debería ser `parametro1=valor1&parametro2=valor2`. También se puede indicar un array asociativo de pares clave/valor que se convierten automáticamente en una cadena tipo *query string*
`dataType`	El tipo de dato que se espera como respuesta. Si no se indica ningún valor, jQuery lo deduce a partir de las cabeceras de la respuesta. Los posibles valores son: `xml` (se devuelve un documento XML correspondiente al valor `responseXML`), `html` (devuelve directamente la respuesta del servidor mediante el valor `responseText`), `script` (se evalúa la respuesta como si fuera JavaScript y se devuelve el resultado) y `json` (se evalúa la respuesta como si fuera JSON y se devuelve el objeto JavaScript generado)

error	Indica la función que se ejecuta cuando se produce un error durante la petición. Esta función recibe el objeto XMLHttpRequest como primer parámetro, una cadena de texto indicando el error como segundo parámetro y un objeto con la excepción producida como tercer parámetro
ifModified	Permite considerar como correcta la petición solamente si la respuesta recibida es diferente de la anterior respuesta. Por defecto su valor es false
processData	Indica si se transforman los datos de la opción data para convertirlos en una cadena de texto. Si se indica un valor de false, no se realiza esta transformación automática
success	Permite establecer la función que se ejecuta cuando una petición se ha completado de forma correcta. La función recibe como primer parámetro los datos recibidos del servidor, previamente formateados según se especifique en la opción dataType
timeout	Indica el tiempo máximo, en milisegundos, que la petición espera la respuesta del servidor antes de anular la petición
type	El tipo de petición que se realiza. Su valor por defecto es GET, aunque también se puede utilizar el método POST
url	La URL del servidor a la que se realiza la petición

```
// Petición GET simple
$.get('/ruta/hasta/pagina.php');

// Petición GET con envío de parámetros y función que
// procesa la respuesta
$.get('/ruta/hasta/pagina.php',
  { articulo: '34' },
  function(datos) {
    alert('Respuesta = '+datos);
  });
```