UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

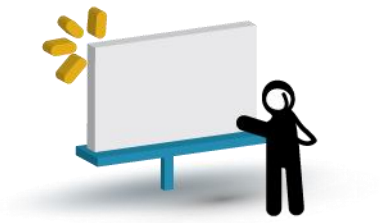
Centro de e-Learning

Proffessional backend developer

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Modulo 2: Javascript

Unidad 2: Manejo del DOM con javascript



Presentación:

En la presente unidad los conceptos esenciales para trabajar con javasript, su historia y el potencial presente y futuro que tiene utilizado a través de distintos frameworks y bibliotecas (jquery, node js y angular por ejemplo).

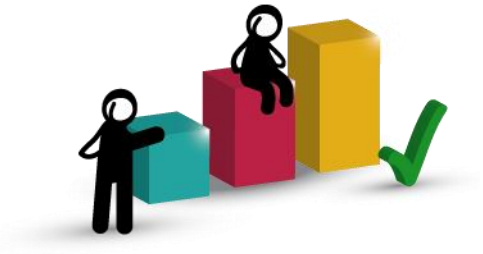
Objetivos:

Que los participantes*:

- Aprendan a maquetar en HTML y la sintaxis básica de javascript
- Conozcan cómo dar diseño a sus páginas utilizando CSS
- Comprendan qué es Javascript, el DOM y la importancia de la estructura de una página web

Bloques temáticos*:

- Javascript – DOM
- Javascript – Formularios
- Javascript – Eventos
- Javascript – Seleccionar elemento e imprimir en html
- Ejemplos en javascript



Consignas para el aprendizaje colaborativo

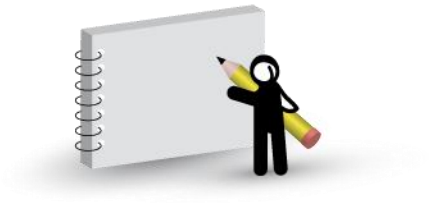
En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.



Tomen nota*

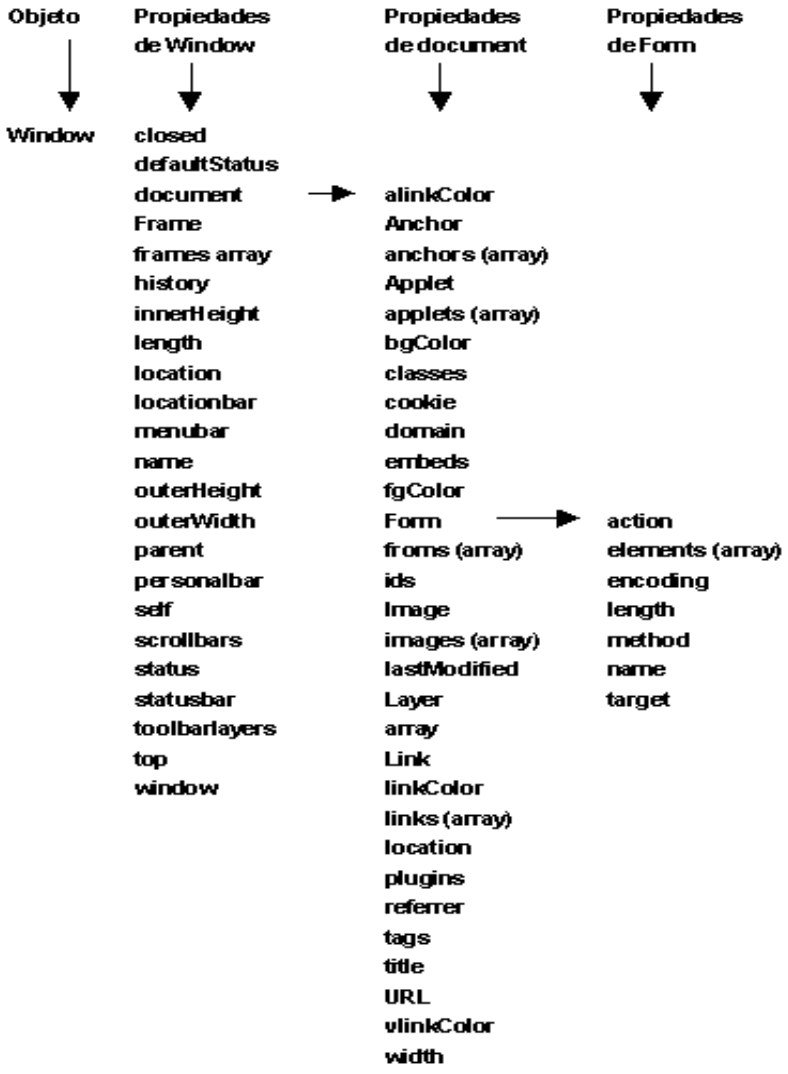
Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

Javascript – DOM

Definición

DOM (Document Object Model o modelo de objetos del navegador) que nos sirve para acceder a cualquiera de los componentes que hay dentro de una página. Por medio del DOM podremos controlar al navegador en general y a los distintos elementos que se encuentran en la página. Jerarquía del DOM:



Window

Todos los objetos comienzan en un objeto que se llama window. Este objeto ofrece una serie de métodos y propiedades para controlar la ventana del navegador. Con ellos podemos controlar el aspecto de la ventana, la barra de estado, abrir ventanas secundarias y otras cosas que veremos más adelante cuando expliquemos con detalle el objeto.

Además de ofrecer control, el objeto window da acceso a otros objetos como el documento (La página web que se está visualizando), el historial de páginas visitadas o los distintos frames de la ventana. De modo que para acceder a cualquier otro objeto de la jerarquía deberíamos empezar por el objeto window. Tanto es así que javascript entiende perfectamente que la jerarquía empieza en window aunque no lo señalemos.

```
<script>
window.document.bgColor = "red"
window.document.write("El texto a escribir")
</script>
```

Propiedades:

- **document**: Objeto que contiene el la página web que se está mostrando.
- **Frame**: Un objeto frame de una página web. Se accede por su nombre.
- **history**: Objeto historial de páginas visitadas.
- **location**: La URL del documento que se está visualizando. Podemos cambiar el valor de esta propiedad para movernos a otra página.
- **name**: Nombre de la ventana. Lo asignamos cuando abrimos una nueva ventana.

Métodos:

- **alert(texto):** Presenta una ventana de alerta donde se puede leer el texto que recibe por parámetro
- **blur():** Quitar el foco de la ventana actual
- **close():** Cierra la ventana.
- **forward():** Ir una página adelante en el historial de páginas visitadas.
- **open():** Abre una ventana secundaria del navegador.

Document

Se trata de las propiedades que son arrays, por ejemplo la propiedad images es un array con todas las imágenes de la página web. También encontramos arrays para guardar los enlaces de la página, los applets, los formularios y las anclas.

Cuando una página se carga, el navegador construye en memoria la jerarquía de objetos. De manera adicional, construye también estos arrays de objetos.

```
<script>
for (i=0;i<document.images.length;i++){
    document.write(document.images[i].src)
    document.write("<br>")
}
</script>
```

Propiedades:

- **bgColor**: El color de fondo del documento.
- **cookie**: Accede a una cookie del navegador
- **domain**: Nombre del dominio del servidor de la página.
- **fgColor**: El color del texto. Para ver los cambios hay que recargar la página.
- **ids**: Para acceder a estilos CSS.
- **title**: El título de la página.

Métodos:

- **close():** Cierra el flujo del documento.
- **open():** Abre el flujo del documento.
- **write():** Escribe dentro de la página web. Podemos escribir etiquetas HTML y texto normal.
- **writeln():** Escribe igual que el método write(), aunque coloca un salto de línea al final.

Javascript – Formularios

Acceder a formularios con document

```
> document.form_registrar
<- ▶<form method="POST" name="form_registrar" class="regForm">…</form>
```

Acceder al formulario a través del objeto document y el nombre del formulario.

```
> document.forms[0]
<- ▶<form method="POST" name="form_registrar" class="regForm">…</form>
```

Acceder al formulario a través del objeto document y el índice del formulario.

Acceder a campos dentro del formulario

```
document.form_registrar.nombre
  <input name="nombre" type="text" class="form-control" aria-required="true">
```

Acceder al valor del campo

```
document.form_registrar.nombre.value
""
```

Evento onclick

Con este evento definimos una acción (función) a ejecutar cuando se hace click en un elemento (un botón)

```
<input type="Button" name="enviar" value=" enviar " onclick="enviar()">
```

Propiedades

- **Action**: Es la acción que queremos realizar cuando se submite un formulario.
- **Encoding**: El tipo de codificación del formulario
- **Length**: El número de campos del formulario.
- **Method**: El método por el que mandamos la información. Corresponde con el atributo METHOD del formulario.
- **Name**: El nombre del formulario, que corresponde con el atributo NAME del formulario.
- **Target**: La ventana o frame en la que está dirigido el formulario. Cuando se submite se actualizará la ventana o frame indicado. Corresponde con el atributo target del formulario.

Ejemplo de propiedad method:

```
document.form_registrar.method
"post"
```

Métodos

- **submit():** Para hacer que el formulario se submite, aunque no se haya pulsado el botón de submit.
- **reset():** Para reiniciar todos los campos del formulario, como si se hubiese pulsado el botón de reset.

Javascript – Campos de texto

Propiedades

- **defaultValue**: Es el valor por defecto que tiene un campo. Lo que contiene el atributo VALUE de la etiqueta <INPUT>.
- **Form**: Hace referencia al formulario.
- **Name**: Contiene el nombre de este campo de formulario
- **Type**: Contiene el tipo de campo de formulario que es.
- **Value**: El texto que hay escrito en el campo.

Métodos

- **blur():** Retira el foco de la aplicación del campo de texto.
- **focus():** Pone el foco de la aplicación en el campo de texto.
- **select():** Selecciona el texto del campo.

Javascript – Checkbox

Propiedades

- **checked**: Informa sobre el estado del checkbox. Puede ser true o false.
- **defaultChecked**: Si está chequeada por defecto o no.
- **Value**: El valor actual del checkbox.

Métodos

- **click():** Es como si hiciésemos un click sobre el checkbox, es decir, cambia el estado del checkbox.
- **blur():** Retira el foco de la aplicación del checkbox.
- **focus():** Coloca el foco de la aplicación en el checkbox.

Javascript – Select

Propiedades

- **length**: Guarda la cantidad de opciones del campo select. Cantidad de etiquetas <OPTION>
- **Option**: Hace referencia a cada una de sus opciones. Son por sí mismas objetos.
- **Options**: Un array con cada una de las opciones del select.
- **selectedIndex**: Es el índice de la opción que se encuentra seleccionada.

Métodos

- **blur():** Para retirar el foco de la aplicación de ese elemento de formulario.
- **focus():** Para poner el foco de la aplicación.
- **Objeto option:**
  - **defaultSelected**
    Indica con un true o un false si esa opción es la opción por defecto.
  - **index**
    El índice de esa opción dentro del select.
  - **selected**
    Indica si esa opción se encuentra seleccionada o no.
  - **text**
    Es el texto de la opción. Lo que puede ver el usuario en el select, que se escribe después de la etiqueta <OPTION>.
  - **value**
    Indica el valor de la opción, que se introduce con el atributo VALUE de la etiqueta <OPTION>.

Javascript – Eventos

Manejadores

- **onblur**
  Se desata un evento onblur cuando un elemento pierde el foco de la aplicación. El foco de la aplicación es el lugar donde está situado el cursor, por ejemplo puede estar situado sobre un campo de texto, una página, un botón o cualquier otro elemento.
- **onchange**
  Se desata este evento cuando cambia el estado de un elemento de formulario, en ocasiones no se produce hasta que el usuario retira el foco de la aplicación del elemento.
- **onclick**
  Se produce cuando se da una pulsación o clic al botón del ratón sobre un elemento de la página, generalmente un botón o un enlace.
- **onload**
  Este evento se desata cuando la página, o en Javascript 1.1 las imágenes, ha terminado de cargarse.
- **onsubmit**
  Ocurre cuando el visitante apreta sobre el botón de enviar el formulario. Se ejecuta antes del envío propiamente dicho.

Para ver más manejadores: ***http://www.desarrolloweb.com/articulos/1236.php***

Ejemplo de on blur:

```
</head>
<body>
<form name=f1>
Escriba un número entero: <input type=text name=numero size=8 value="" onblur="compruebaValidoEntero()">
</form>

</body>
</html>
```

Javascript – Seleccionar elemento e imprimir en html

getElementById

Permite, como su nombre indica, seleccionar un elemento del documento por medio del valor del atributo id que se le haya asignado

```
<script>
    document.getElementById('id_del_elemento');
</script>
```

InnerHTML

Retorna o setea el contenido HTML de un elemento

- Set (Modifica el HTML para mostrar el valor en pantalla)

```
document.getElementById("alertas").innerHTML += "El campo nombre es obligatorio<br/>";
```

- Get (Obtiene el valor que tiene el elemento en pantalla)

```
document.getElementById("alertas").innerHTML;
```

Cuando el elemento es de formulario (input, select, textarea, etc) obtenemos o modificamos el valor del mismo con la propiedad value.

Cuando el elemento es HTML (div, span, label, etc) obtenemos o modificar el valor del mismo con la propiedad innerHTML

Ejemplos en javascript

Validar un entero en javascript

Creamos un archivo llamado “script.js” el cual contiene la siguiente función:

```
function validarEnteroEnCampo(identificadorDelCampo) {
   var field = document.getElementById(identificadorDelCampo);
   var valueInt = parseInt(field.value);

   if (!Number.isInteger(valueInt)) {
     alert("No es un entero")
   } else {
     field.value = valueInt;

   }
}
```

parseInt: Parsea el valor que pasamos como parámetro a un entero. Es decir, js por ejemplo tomara los valores que introducimos en un input como string, aplicando parseInt transforma los mismos a un tipo de dato Number.

Number.isInteger: Retorna true si el valor que pasamos como parámetro es un entero y false si no lo es.

HTML:

```
<html>
<head>
<title>Validar entero</title>
<script src="script.js"></script>
</head>

<body >

<form name=formul>
<input type=text name=texto id="texto">
<input type=button value=validar onclick="validarEnteroEnCampo('texto')">
</form>

</body>
</html>
```

Comprobar si dos claves son iguales

Creamos un archivo llamado “script.js” el cual contiene la siguiente función:

```
function comprobarClave(){
  clave1 = document.f1.clave1.value
  clave2 = document.f1.clave2.value

  if (clave1 == clave2)
    alert("Las dos claves son iguales...\nRealizaríamos las acciones del caso positivo")
  else
    alert("Las dos claves son distintas...\nRealizaríamos las acciones del caso negativo")
}
```

Como vemos con la utilización del DOM podemos acceder al elemento y su propiedad value con lo cual obtenemos lo que el usuario escribió en ambos campos.

HTML:

```
<html>
<head>
<title>Comprobar Claves</title>
<script src="script.js"></script>
</head>

<body >

<form action="" name="f1">
    Contraseña: <input type="password" name="clave1" size="20">
    <br>
    Repite contraseña: <input type="password" name="clave2" size="20">
    <br>
    <input type="button" value="Comprobar si son iguales" onClick="comprobarClave()">
</form>

</body>
</html>
```

Del lado del HTML se resume a implementar un formulario y la captura del evento onclick

Inhibir un campo de texto

El concepto focus, está relacionado con ganar foco de la aplicación. El método focus (), que tienen los campos de texto y otros elementos de formulario, sirve otorgar el foco de la aplicación a ese elemento. El manejador de evento onfocus salta cuando un elemento gana el foco de la aplicación.

El concepto blur, está asociado a perder el foco de la aplicación. El método blur() sirve para que los elementos de formulario pierdan el foco y el manejador de eventos onblur se activa cuando el elemento al que lo apliquemos pierda el foco de la aplicación.

Nosotros utilizaremos el evento onfocus para detectar el instante en el que el elemento gana el foco y en ese momento haremos uso del método blur() para retirar el foco.

```
<html>
<head>
<title>Inhibir campo de texto</title>
<script src="script.js"></script>
</head>

<body >

    <form>
        <input type="text" value="122" onfocus="this.blur()">
    </form>

</body>
</html>
```

El único detalle que merece la pena señalar es el uso de la palabra this, que hace referencia al elemento donde se está utilizando, en ese caso el campo de texto. this.blur() sería una simple llamada al método blur() en el elemento de formulario donde está colocada.

Contar caracteres en un textarea

Creamos un archivo llamado “script.js” el cual contiene la siguiente función:

```
function cuenta(){
   document.forms[0].caracteres.value=document.forms[0].texto.value.length
}
```

En este caso accedemos al formulario de índice 0 en la colección de formularios de document (en este caso solo tenemos un formulario, pero si tuviésemos mas en el HTML accederíamos al primero)

Dentro del formulario actualizamos el value del elemento “caracteres” (input) con la cantidad de caracteres que presente el elemento “texto” (textarea).

Length: Retorna la cantidad de elementos de un array o la cantidad de caracteres de un campo de texto

HTML:

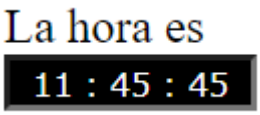
```
<html>
<head>
<title>Contar Caracteres</title>
<script src="script.js"></script>
</head>

<body >

    <form action="#" method="post">
        <table>
            <tr>
                <td>Texto:</td>
                <td>
                    <textarea cols="40" rows="5" name="texto" onKeyDown="cuenta()" onKeyUp="cuenta()">
                    </textarea>
                </td>
            </tr>
            <tr>
                <td>Caracteres:</td>
                <td><input type="text" name=caracteres size=4></td>
            </tr>
        </table>
    </form>

</body>
</html>
```

onKeyDown: Captura la escritura de cualquier carácter mediante el teclado al momento de ser presionada la tecla

onKeyUp: Captura la escritura de cualquier carácter mediante el teclado al momento de ser soltada la tecla

Reloj

Desarrollemos un pequeño “reloj” que muestre la hora actual por pantalla utilizando javascript



Creamos un archivo llamado “script.js” el cual contiene la siguiente función:

```
function mueveReloj(){
    var momentoActual = new Date()
    var hora = momentoActual.getHours()
    var minuto = momentoActual.getMinutes()
    var segundo = momentoActual.getSeconds()

    var str_segundo = new String (segundo)
    if (str_segundo.length == 1)
       segundo = "0" + segundo

    var str_minuto = new String (minuto)
    if (str_minuto.length == 1)
       minuto = "0" + minuto

    str_hora = new String (hora)
    if (str_hora.length == 1)
       hora = "0" + hora

    var horaImprimible = hora + " : " + minuto + " : " + segundo

    document.form_reloj.reloj.value = horaImprimible

    setTimeout("mueveReloj()",1000)
}
```

Asignamos

var momentoActual = new Date()

new Date() al no recibir parámetro en su constructor nos retorna la fecha actual

var hora = momentoActual.getHours()

var minuto = momentoActual.getMinutes()

var segundo = momentoActual.getSeconds()

Con el objeto resultante de Date() obtenemos la hora, minutos y segundos

```
var str_segundo = new String (segundo)
   if (str_segundo.length == 1)
      segundo = "0" + segundo


   var str_minuto = new String (minuto)
   if (str_minuto.length == 1)
      minuto = "0" + minuto


   str_hora = new String (hora)
   if (str_hora.length == 1)
      hora = "0" + hora
```

Verificamos el formato y agregamos un 0 a la izquierda en caso de que la hora, minutos o segundos tenga una longitud de 1

var horaImprimible = hora + " : " + minuto + " : " + segundo

document.form_reloj.reloj.value = horaImprimible

setTimeout("mueveReloj()",1000)

Armamos el string con el formato de la hora, incluyendo los dos puntos. Seteamos el value del input correspondiente y actualizamos cada 1 segundo utilizando la función setTimeout

En el HTML tenemos lo siguiente:

```
<html>
<head>
<title>Reloj con Javascript</title>
<script src="script.js"></script>
</head>

<body onload="mueveReloj()">

La hora es

<form name="form_reloj">
<input type="text" name="reloj" size="10" >
</form>

</body>
</html>
```

onLoad: Esto quiere decir que cuando se termine de cargar la página se llame a la función mueveReloj(), que se encargará de mover el reloj y llamarse a sí misma para hacer el proceso de manera continuada.

Bibliografía y Webgrafía utilizada y sugerida

https://developer.mozilla.org/es/docs/Tools/Page_Inspector

http://www.w3schools.com/Js/

Lo que vimos:

En esta unidad aprendimos lo esencial para poder comenzar a trabajar con el DOM de javascript

Lo que viene:

En la próxima unidad aprendemos más sobre Javscript, su sintaxis básica, módulos y componentes con javascript ECMA script 6