

# Professional backend developer



**UTN.BA**  
UNIVERSIDAD TECNOLÓGICA NACIONAL  
FACULTAD REGIONAL BUENOS AIRES

**Centro de  
e-Learning**

p. 2

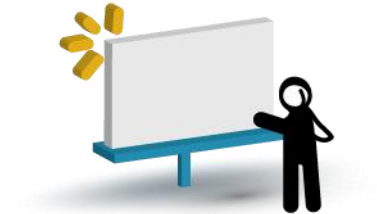
## **Modulo 2: Frameworks y jQuery**

### **Unidad 2: Nivelación Javascript**

**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](http://www.sceu.frba.utn.edu.ar/e-learning)**



## Presentación:

En la presente unidad los conceptos esenciales para trabajar con javascript, 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).

**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](http://www.sceu.frba.utn.edu.ar/e-learning)**



## Objetivos:

### Que los participantes\*:

- Conozcan el surgimiento y ventajas de javascript
- Aprendan conceptos básicos del mismo, como el DOM
- Conozcan su sintaxis básica y puedan aplicarla.

**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](http://www.sceu.frba.utn.edu.ar/e-learning)**



## Bloques temáticos\*:

- Javascript
- Javascript – Sintaxis
- Javascript – Variables
- Javascript – Operadores
- Javascript – Estructuras
- Javascript – Funciones – Arrays
- Javascript – Clases y objetos
- Javascript – DOM
- Javascript – Formularios
- Javascript – Campos de texto
- Javascript – Checkbox
- Javascript – Select
- Javascript – Eventos
- Javascript – Seleccionar elemento e imprimir en html

**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](http://www.sceu.frba.utn.edu.ar/e-learning)**

**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](http://www.sceu.frba.utn.edu.ar/e-learning)**



## Consignas para el aprendizaje colaborativo

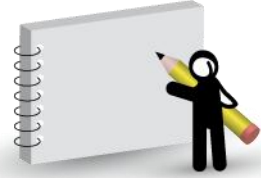
En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC\*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- 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

### Definición

Javascript es un lenguaje de programación que surgió con el objetivo inicial de programar ciertos comportamientos sobre las páginas web, respondiendo a la interacción del usuario y la realización de automatismos sencillos.

En los últimos años Javascript se está convirtiendo también en el lenguaje "integrador". **Lo encontramos en muchos ámbitos, ya no solo en Internet y la Web, también es nativo en sistemas operativos para ordenadores y dispositivos, del lado del servidor y del cliente.** Aquella visión de Javascript "utilizado para crear pequeños programitas encargados de realizar acciones dentro del ámbito de una página web" se ha quedado muy pequeña.

En el contexto de un sitio web, con Javascript puedes hacer todo tipo de acciones e interacción. Antes se utilizaba para validar formularios, mostrar cajas de diálogo y poco más. Hoy es el motor de las aplicaciones más conocidas en el ámbito de Internet: Google, Facebook, Twitter, Outlook... absolutamente todas las aplicaciones que disfrutas en tu día a día en la Web tienen su núcleo realizado en toneladas de Javascript.

La Web 2.0 se basa en el uso de Javascript para implementar aplicaciones enriquecidas que son capaces de realizar todo tipo de efectos, interfaces de usuario y comunicación asíncrona con el servidor por medio de Ajax.

Resumiendo...

- Lenguaje del lado del cliente
  - Se ejecuta del lado del navegador
- Podemos crear efectos especiales en las páginas y definir interactividades con el usuario
- **API Javascript del HTML5:** podemos acceder a todo tipo de recursos adicionales, como la cámara, espacio para almacenamiento de datos, creación de gráficos basados en vectores y mapas de bits
- **Javascript != Java**



## **Tipos de ejecución**

- El código javascript en muchos casos se embebe dentro del código HTML
- El código Javascript se coloca dentro de los tags **<script></script>**
- Hay dos formas de ejecutar código javascript:
  - Ejecución directa
  - Ejecución por eventos

### **Ejecución directa**

Es el método de ejecutar scripts más básico. En este caso se incluyen las instrucciones dentro de la etiqueta **<SCRIPT>**, cuando el navegador lee la página y encuentra un script va interpretando las líneas de código y las va ejecutando una después de otra.

### **Respuesta a un evento**

Los eventos son acciones que realiza el usuario. Los programas como Javascript están preparados para atrapar determinadas acciones realizadas, en este caso sobre la página, y realizar acciones como respuesta.

Por ejemplo la pulsación de un botón, el movimiento del ratón o la selección de texto de la página.

## **Incluir archivos externos**

Desde nuestro HTML podemos incluir código javascript externo.

Esto lo podemos hacer con el tag **<script>**, veamos el siguiente ejemplo:

```
<script src="/carrito_compra/assets/bootstrap/js/bootstrap.min.js"></script>
```



## Javascript – Sintaxis

### Comentarios

```
<script>
//Este es un comentario de una línea
/*Este comentario se puede extender
por varias líneas.
Las que quieras*/
</script>
```

En caso de utilizar archivos JS minificados, se deben utilizar comentarios extendidos y NO de una línea.

\* Archivo minificado -> un js minificado es “comprimir” todo el contenido de nuestro js en un archivo de una única línea.

### Separación de instrucciones

Javascript tiene dos maneras de separar instrucciones:

- La primera es a través del carácter punto y coma (;)

```
llamadoFuncion();
```

- La segunda es a través de un salto de línea.

```
llamadoFuncion()
```



## Javascript – Variables

### Variables

**Una variable es un espacio en memoria donde se almacena un dato**, un espacio donde podemos guardar cualquier tipo de información que necesitemos para realizar las acciones de nuestros programas.

Los nombres de las variables han de construirse con caracteres alfanuméricos y el carácter subrayado (\_).

Los nombres tienen que comenzar por un carácter alfabético o el subrayado. **No podemos utilizar caracteres raros como el signo +, un espacio.**

**Las variables no pueden utilizar nombres reservados, por ejemplo if, for, while, etc**



## Declaración de Variables

Declarar variables consiste en definir y de paso informar al sistema de qué vas a utilizar una variable.

Javascript cuenta con la palabra "**var**" que utilizaremos cuando queramos declarar una o varias variables. Como es lógico, se utiliza esa palabra para definir la variable antes de utilizarla.

Ejemplo:

```
var operando1;  
var operando2;  
  
var operando1 = 23;  
var operando2 = 33;  
  
var operando1,operando2;|
```

Ejemplo:

```
-----  
<script language="javascript">  
  
    //creo una variable  
    var x;  
    //x actualmente no tiene ningún valor  
    x = 25;  
    //ahora x tiene el valor numérico 25  
  
    //creo una segunda variable  
    var y = 230;  
    //he creado una variable y asignado un valor en un solo paso!!  
  
    //las variables guardan datos con los que puedo realizar operaciones  
    var suma;  
    suma = x + y;  
    //he creado la variable suma y he asignado la suma de x e y  
    alert(suma);  
    //he mostrado el valor de la variable suma  
  
</script>
```



## Ámbito de las Variables

**Se le llama ámbito de las variables al lugar donde estas están disponibles.** Por lo general, cuando declaramos una variable hacemos que esté disponible en el lugar donde se ha declarado.

- Variables globales: son las que están declaradas en el ámbito más amplio posible.

```
<script>  
var variableGlobal  
</script>
```

- Variables locales: sólo podremos acceder a ellas dentro del lugar donde se ha declarado

```
<script>  
function miFuncion () {  
    var variableLocal  
}  
</script>
```



## Javascript – Operadores

### Operadores de cadenas

Las cadenas de caracteres, o variables de texto, también tienen sus propios operadores para realizar acciones típicas sobre cadenas. Aunque javascript sólo tiene un operador para cadenas se pueden realizar otras acciones con una serie de funciones predefinidas en el lenguaje que veremos más adelante.

Para concatenar 2 cadenas de texto debemos usar el operador “+”

```
<script>
cadenaConcatenada = cadena1 + cadena2 /
</script>
```

### Operadores lógicos

Estos operadores sirven para realizar operaciones lógicas, que son aquellas que dan como resultado un verdadero o un falso, y se utilizan para tomar decisiones en nuestros scripts.

```
<script>

prueba && prueba2 //Operador and
prueba || prueba2 //Operador or

</script>
```

### Typeof

Para comprobar el tipo de un dato se puede utilizar otro operador que está disponible a partir de javascript 1.1, el **operador typeof**, que devuelve una cadena de texto que describe el tipo del operador que estamos comprobando.

```
<script>

document.write("<br>El tipo de booleano es: " + typeof booleano)

</script>
El tipo de booleano es: boolean
```

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](http://www.sceu.frba.utn.edu.ar/e-learning)



## Operadores de asignación

Un operador de asignación asigna un valor al operando de la izquierda en función del valor del operando de la derecha. El operador básico de asignación es el de igual (=), que asigna el valor del operando de la derecha al operando de la izquierda. Por ejemplo,  $x = y$ , está asignando el valor de  $y$  a  $x$ .

Nombre	Operador abreviado	Significado
Operadores de asignación	$x = y$	$x = y$
Asignación de adición	$x += y$	$x = x + y$
Asignación de sustracción	$x -= y$	$x = x - y$
Asignación de multiplicación	$x *= y$	$x = x * y$
Asignación de división	$x /= y$	$x = x / y$
Asignación de resto	$x \% = y$	$x = x \% y$
Asignación de exponenciación	$x ** = y$	$x = x ** y$
Asignación de desplazamiento a la izquierda	$x << = y$	$x = x << y$
Asignación de desplazamiento a la derecha	$x >> = y$	$x = x >> y$
Asignación de desplazamiento a la derecha sin signo	$x >>> = y$	$x = x >>> y$
Asignación AND binaria	$x \& = y$	$x = x \& y$
Asignación XOR binaria	$x \wedge = y$	$x = x \wedge y$
Asignación OR binaria	$x   = y$	$x = x   y$





## Operadores de comparación

Un operador de comparación compara sus operandos y devuelve un valor lógico en función de si la comparación es verdadera (true) o falsa (false). Los operadores pueden ser numéricos, de cadena de caracteres (Strings), lógicos o de objetos.


Operador	Descripción	Ejemplos devolviendo true
Igualdad (==)	Devuelve true si ambos operandos son iguales.	<code>3 == var1</code> <code>"3" == var1</code> <code>3 == "3"</code>
Desigualdad (!=)	Devuelve true si ambos operandos no son iguales.	<code>var1 != 4</code> <code>var2 != "3"</code>
Estrictamente iguales (===)	Devuelve true si los operandos son igual y tienen el mismo tipo. Mira también <a href="#">Object.is</a> y <a href="#">sameness in JS</a> .	<code>3 === var1</code>
Estrictamente desiguales (!==)	Devuelve true si los operandos no son iguales y/o no son del mismo tipo.	<code>var1 !== "3"</code> <code>3 !== "3"</code>
Mayor que (>)	Devuelve true si el operando de la izquierda es mayor que el operando de la derecha.	<code>var2 &gt; var1</code> <code>"12" &gt; 2</code>
Mayor o igual que (>=)	Devuelve true si el operando de la izquierda es mayor o igual que el operando de la derecha.	<code>var2 &gt;= var1</code> <code>var1 &gt;= 3</code>
Menor que (<)	Devuelve true si el operando de la izquierda es menor que el operando de la derecha.	<code>var1 &lt; var2</code> <code>"2" &lt; 12</code>
Menor o igual que (<=)	Devuelve true si el operando de la izquierda es menor o igual que el operando de la derecha.	<code>var1 &lt;= var2</code> <code>var2 &lt;= 5</code>



## Operadores aritméticos

Los operadores aritméticos toman los valores numéricos (tanto literales como variables) de sus operandos y devuelven un único resultado numérico. Los operadores aritméticos estándar son la suma (+), la resta (-), la multiplicación (\*) y la división (/).

Tabla 2.3 Operadores aritméticos

Operador	Descripción	Ejemplo
<a href="#">Resto (%)</a>	Operador binario correspondiente al módulo de una operación. Devuelve el resto de la división de dos operandos.	12 % 5 devuelve 2.
<a href="#">Incremento (++)</a>	Operador unario. Incrementa en una unidad al operando. Si es usado antes del operando (++x) devuelve el valor del operando después de añadirle 1 y si se usa después del operando (x++) devuelve el valor de este antes de añadirle 1.	Si x es 3, entonces ++x establece x a 4 y devuelve 4, mientras que x++ devuelve 3 y, solo después de devolver el valor, establece x a 4.
<a href="#">Decremento (--)</a>	Operador unario. Resta una unidad al operando. Dependiendo de la posición con respecto al operando tiene el mismo comportamiento que el operador de incremento.	Si x es 3, entonces --x establece x a 2 y devuelve 2, mientras que x-- devuelve 3 y, solo después de devolver el valor, establece x a 2.
<a href="#">Negación Unaria (-)</a>	Operación unaria. Intenta convertir a número al operando y devuelve su forma negativa.	- "3" devuelve -3. -true devuelve -1.
<a href="#">Unario positivo (+)</a>	Operación unaria. Intenta convertir a número al operando.	+ "3" devuelve 3. +true devuelve 1.
<a href="#">Exponenciación (**)</a> 	Calcula la potencia de la base al valor del exponente. Es equivalente a $base^{exponente}$	2 ** 3 devuelve 8. 10 ** -1 devuelve 0.1.



## Javascript – Estructuras

### IF

La estructura if nos permitirá mediante el uso de operadores de comparación tomar una decisión a partir de si dicha comparación o valor de una expresión es verdadera o falsa (en este caso entra por el else)

```
<script>

if (expresión) { |
    //acciones a realizar en caso positivo
    //...
} else {
    //acciones a realizar en caso negativo
    //...
}

</script>
```

### For

La estructura for nos permitirá recorrer un array en javascript.

Cada elemento recorrido se reconoce como una iteración, se “cortara” el for cuando la condición sea falsa.

```
<script>
for (inicialización; condición; actualización) {
    //sentencias a ejecutar en cada iteración
}
</script>
```

- Inicialización: Aquí inicializamos las variables. Por ejemplo: **i=0**
- Condición: Mientras la condición sea verdadera se seguirán produciendo iteraciones. Ejemplo **i<5**
- Actualización: se actualiza la variable de la condición. Ejemplo: **i++**

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](http://www.sceu.frba.utn.edu.ar/e-learning)



## While

La estructura while es similar a la for, nos permite realizar iteraciones sobre un array

```
<script>
while (condición){
    //sentencias a ejecutar
}
</script>
```

Como vemos no cuenta con el elemento de inicialización ni el de actualización.



## Javascript – Funciones - Arrays

### Definición

Primero se escribe la palabra **function**, reservada para este uso. Seguidamente se escribe el nombre de la función, que como los nombres de variables puede tener números, letras y algún carácter adicional como en guión bajo.

```
<script>
function nombrefuncion () {
    instrucciones de la función
    ... |
}
</script>
```

Entre paréntesis, luego del nombre de la función, se definirán los parámetros que recibirá la misma. En el caso de js no hace falta el tipo de dato en cada parámetro recibido.

### Arrays

El primer paso para utilizar un array es crearlo. Para ello utilizamos un objeto Javascript ya implementado en el navegador.

```
<script>
//Array vacio
var miArray = new Array()
//Creacion de array con numero de compartimentos
var miArray = new Array(10)
//Setear datos
miArray[0] = 290
miArray[1] = 97
miArray[2] = 127
//Declaracion e inicializacion
var arrayRapido = [12,45,"array inicializado en su declaración"]
</script>
```

A diferencia de PHP no podemos tener vectores asociativos, es decir arrays cuya clave sea un string.



## Longitud de un array

Para obtener la longitud de un array utilizaremos el método **length**

```
<script>  
document.write("Longitud del array: " + miArray.length)  
</script>
```

## Javascript – Clases y objetos

### Objetos

Para instanciar un objeto lo hacemos con el método **new**. Para acceder a un método de un objeto lo hacemos con “.”

```
<script>  
//Instancia un objeto  
var miObjeto = new miClase()  
//Acceder a un metodo del objeto  
miObjeto.miMetodo(parametro1,parametro2)  
</script>
```

### Clase string

Propiedades:

- **Length**: Propiedad de la clase string que guarda el numero de caracteres del string.

Métodos:

- **charAt(índice)**: Devuelve el carácter que hay en la posición indicada como índice.
- **indexOf(carácter, desde)**: Devuelve la primera vez que aparece el carácter indicado por el parámetro en un string. Si no lo encuentra devuelve -1. El segundo parámetro es opcional y sirve para indicar a partir de qué posición se desea que empiece la búsqueda.
- **lastIndexOf(carácter, desde)**: Igual que la función anterior pero busca desde atrás.



## Clase date

```
<script>
//Instanciar el objeto
miFecha = new Date(año,mes,día,hora,minutos,segundos)
</script>
```

Métodos:

- **getDate():** Devuelve el día del mes
- **getDay():** Devuelve el día de la semana
- **getHours():** retorna la hora
- **getMinutes():** Devuelve los minutos
- **getMonths():** Devuelve el mes
- **getSeconds():** Devuelve los segundos

## Creación de clases

Para crear nuestros propios objetos debemos crear una clase, que recordamos que es algo así como la definición de un objeto con sus propiedades y métodos. Para crear la clase en Javascript debemos escribir una función especial, que se encargará de construir el objeto en memoria e inicializarlo. Esta función se le llama **constructor**.

```
<script>
function MiClase (valor_inicializacion){
    //Inicializo las propiedades y métodos
    //this hace referencia a su propia clase
    this.miPropiedad = valor_inicializacion
    this.miMetodo = nombre_de_una_funcion_definida
}
</script>
```

El nombre del constructor comienza con mayúscula



Para colocar un método en una clase debemos asignar la función que queremos que sea

El método al objeto que se está creando.

```
<script>
function AlumnoUniversitario(nombre, edad){
    this.nombre = nombre
    this.edad = edad
    this.numMatricula = null
    //matriculate es una funcion declarada a parte
    this.matriculate = matriculate
    this.imprime = imprime
}
function matriculate(){
    //Declaracion del metodo
}
</script>
```





## 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.

Para ver más: <http://www.desarrolloweb.com/articulos/826.php>



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.

Para ver más: <http://www.desarrolloweb.com/articulos/827.php>

## 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.

Para ver más: <http://www.desarrolloweb.com/articulos/846.php>

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](http://www.sceu.frba.utn.edu.ar/e-learning)



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.

Para ver más: <http://www.desarrolloweb.com/articulos/861.php>

## 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 submita 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 submita, aunque no se haya pulsado el botón de submit.
- **reset():** Para reinicializar 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

```
2  
3 <script>  
4     document.getElementById('id_del_elemento');  
5 </script>
```

### InnerHTML

Retorna o setea el contenido HTML de un elemento

- Set

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

- Get

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



## **Ejercicio propuesto**

Desarrollador una calculadora que tenga:

- 2 campos inputs para los operandos
- 4 botones de operadores básicos (suma, resta, multiplicación, división)
- Al realizar la cuenta se deberá actualizar el campo resultado



## **Ejercicio de la unidad**

Aplicar una validación de formularios para el formulario de registro:

Validar los campos obligatorios

- Nombre
- Apellido
- Email
- Password
- Repetir Password

Validar los campos numéricos

- Teléfono

Validar el campo password y repetir password sean iguales.



## Bibliografía utilizada y sugerida

<http://www.desarrolloweb.com/manuales/20/>

<http://www.desarrolloweb.com/articulos/826.php>

<http://www.desarrolloweb.com/articulos/827.php>

<http://www.desarrolloweb.com/articulos/846.php>

<http://www.desarrolloweb.com/articulos/861.php>

**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](http://www.sceu.frba.utn.edu.ar/e-learning)**



## Lo que vimos:

En esta unidad hemos aprendido los conceptos teóricos más importantes de javascript y aplicado los mismos de manera práctica.





## Lo que viene:

En la próxima unidad aprenderemos a utilizar una biblioteca que facilitara el uso de javascript, jQuery.

