UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

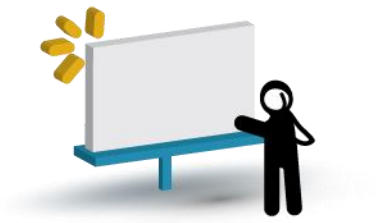
Desarrollo Web con Javascript

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

UTN.BA UNIVERSIDAD TECNOLÓGICA NACIONAL FACULTAD REGIONAL BUENOS AIRES | Centro de e-Learning

Módulo 1: Programando Javascript

Unidad 1: Introducción a Javascript



Presentación:

JavaScript, es un lenguaje de programación de páginas web de lado del cliente, esto significa, que cuando estamos viendo una página que utiliza JavaScript, hemos descargado el código a nuestro navegador y nuestro navegador lo está ejecutando de acuerdo con las acciones realizadas en la página.

Actualmente es una pieza fundamental en el desarrollo de aplicaciones web y es usado por las grandes compañías como Google, Yahoo, Microsoft, etc y se ha convertido en una herramienta tan poderosa, que su conocimiento es uno de los puntos más valorados en las búsquedas laborales de desarrolladores web.

Objetivos:

Que los participantes:

- Logren conocer los conceptos básicos del lenguaje

Bloques temáticos:

1. ¿Qué es Javascript?
2. Breve historia
3. Especificaciones oficiales
4. Javascript y HTML
5. Sintaxis
6. Cuadros de diálogo

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:

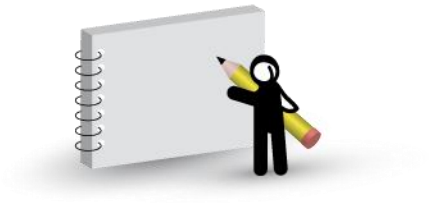
- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

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.

**** El MEC es el modelo de E-learning colaborativo de nuestro Centro.***



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.

1. ¿Qué es Javascript?

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas.

Una página web dinámica es aquella que incorpora distintos efectos (texto que aparece y desaparece), animaciones, acciones que se activan al pulsar o al pasar sobre botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. No requieren de un intérprete, como otros lenguajes web (php o mysql, por ejemplo).

JavaScript contiene una librería estándar de objetos, tales como Array, Date, y Math, y un conjunto central de elementos del lenguaje, tales como operadores, estructuras de control, y sentencias. El núcleo de JavaScript puede extenderse para varios propósitos, complementándolo con objetos adicionales, por ejemplo:

- **Client-side** JavaScript extiende el núcleo del lenguaje proporcionando objetos para controlar un navegador y su modelo de objetos (o DOM, por las iniciales de Document Object Model). Por ejemplo, las extensiones del lado del cliente permiten que una aplicación coloque elementos en un formulario HTML y responda a eventos del usuario, tales como clicks del *mouse*, ingreso de datos al formulario y navegación de páginas.
- **Server-side** JavaScript extiende el núcleo del lenguaje proporcionando objetos relevantes a la ejecución de JavaScript en un servidor. Por ejemplo, las extensiones del lado del servidor permiten que una aplicación se comunique con una base de datos, proporcionar continuidad de la información de una invocación de la aplicación a otra, o efectuar manipulación de archivos en un servidor.

A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java.

Legalmente, JavaScript es una marca registrada de la empresa Sun Microsystems, como se puede ver en http://www.sun.com/suntrademarks/.

JavaScript y Java

JavaScript y Java son similares en algunos aspectos, pero fundamentalmente diferentes en otros.

JavaScript usa la mayoría de la sintaxis de expresiones de Java, convenciones de nombrado, y las construcciones básicas de control de flujo, razón por la cual se le cambió el nombre del original LiveScript a JavaScript.

En contraste con el sistema de clases construidas por declaraciones que se usa en tiempo de compilación de Java, JavaScript soporta un sistema de tiempo de ejecución basado en un pequeño número de tipos de datos que representan valores numéricos, lógicos, y de cadena de caracteres (string). JavaScript tiene un modelo de objetos basado en prototipos en lugar del modelo de objetos basado en clases, que es más común. El modelo basado en prototipo proporciona herencia dinámica; esto es, que lo que se hereda puede variar entre objetos individuales. JavaScript también soporta funciones sin ningún requerimiento declarativo especial. Las funciones pueden ser propiedades de los objetos, ejecutándose como métodos levemente tipados.

Comparado con Java, JavaScript es un lenguaje muy libre de forma. No hay que declarar todas las variables, clases, y métodos. No hay que preocuparse de si los métodos son públicos, privados, o protegidos, y no hay que implementar interfaces. Las variables, parámetros, y retornos de funciones no tienen que declararse explícitamente de un tipo dado.

Java es un lenguaje de programación basado en clases que implica que los programas consisten exclusivamente en clases y sus métodos. Estos requerimientos hacen que la programación en Java sea más compleja que la programación en JavaScript.

En contraposición, JavaScript proviene en espíritu de una línea de lenguajes de programación más pequeños, con tipado dinámico, como HyperTalk, y dBASE. Estos lenguajes de scripting hacen accesibles las herramientas de programación a audiencias mucho más amplias. Esto se debe a su sintaxis más indulgente, funcionalidad especializada ya incluida, y mínimos requisitos para la creación de objetos.

2. Breve historia

A principios de los años 90, la mayoría de usuarios que se conectaban a Internet lo hacían con módems a una velocidad máxima de 28.8 kbps. En esa época, empezaban a desarrollarse las primeras aplicaciones web y por tanto, las páginas web comenzaban a incluir formularios complejos.

Con unas aplicaciones web cada vez más complejas y una velocidad de navegación tan lenta, surgió la necesidad de un lenguaje de programación que se ejecutara en el navegador del usuario. De esta forma, si el usuario no rellenaba correctamente un formulario, no se le hacía esperar mucho tiempo hasta que el servidor volviera a mostrar el formulario indicando los errores existentes.

Brendan Eich, un programador que trabajaba en Netscape, pensó que podría solucionar este problema adaptando otras tecnologías existentes (como ScriptEase) al navegador Netscape Navigator 2.0, que iba a lanzarse en 1995. Inicialmente, Eich denominó a su lenguaje LiveScript.

Posteriormente, Netscape firmó una alianza con Sun Microsystems para el desarrollo del nuevo lenguaje de programación. Además, justo antes del lanzamiento Netscape decidió cambiar el nombre por el de JavaScript. La razón del cambio de nombre fue exclusivamente por marketing, ya que Java era la palabra de moda en el mundo informático y de Internet de la época.

La primera versión de JavaScript fue un completo éxito y Netscape Navigator 3.0 ya incorporaba la siguiente versión del lenguaje, la versión 1.1. Al mismo tiempo, Microsoft lanzó JScript con su navegador Internet Explorer 3. JScript era una copia de JavaScript al que le cambiaron el nombre para evitar problemas legales.

Para evitar una guerra de tecnologías, Netscape decidió que lo mejor sería estandarizar el lenguaje JavaScript. De esta forma, en 1997 se envió la especificación JavaScript 1.1 al organismo ECMA European Computer Manufacturers Association).

ECMA creó el comité TC39 con el objetivo de "estandarizar de un lenguaje de script multiplataforma e independiente de cualquier empresa". El primer estándar que creó el comité TC39 se denominó ECMA-262, en el que se definió por primera vez el lenguaje ECMAScript.

Por este motivo, algunos programadores prefieren la denominación ECMAScript para referirse al lenguaje JavaScript. De hecho, JavaScript no es más que la implementación que realizó la empresa Netscape del estándar ECMAScript.

La organización internacional para la estandarización (ISO) adoptó el estándar ECMA-262 a través de su comisión IEC, dando lugar al estándar ISO/IEC-16262.

3. Especificaciones oficiales

JavaScript está estandarizado en ***Ecma International*** — la asociación europea para la creación de estándares para la comunicación y la información (ECMA originalmente era un acrónimo: European Computer Manufacturers Association) con el fin de ofrecer un lenguaje de programación estandarizado e internacional, basado en Javascript.

Esta versión de JavaScript, llamada ECMAScript, se comporta de la misma manera en todas las aplicaciones que implementan el estándar. Cualquier compañía puede usar el lenguaje basado en estándares abiertos, para desarrollar su implementación de JavaScript.

El estándar ECMAScript está documentado en la especificación ECMA-262.

El estándar ECMA-262 también está aprobado por ***ISO*** (Organización Internacional de Normalización), como ISO-16262.

También se puede encontrar la especificación en ***el sitio de Ecma International***.

La especificación ECMAScript no describe el Document Object Model (DOM), que está estandarizado por el ***World Wide Web Consortium (W3C)***. El DOM define la forma como los objetos de los documentos HTML se exponen a sus scripts (trabajaremos con el DOM en las próximas unidades).

4. Javascript y HTML

La integración de JavaScript y XHTML es muy flexible, ya que existen al menos tres formas para incluir código JavaScript en las páginas web.

Incluir JavaScript en el mismo documento XHTML

El código JavaScript se encierra entre etiquetas <script> y se incluye en cualquier parte del documento. Aunque es correcto incluir cualquier bloque de código en cualquier zona de la página, se recomienda definir el código JavaScript dentro de la cabecera del documento (dentro de la etiqueta <head>):

```
<!doctype html>
<html>
<head>
<meta charset="utf-8"/>
<title>Ejemplo de código JavaScript en el propio documento</title>
<script type="text/javascript">
  alert("Un mensaje de prueba");
</script>
</head>

<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

Este método se emplea cuando se define un bloque pequeño de código o cuando se quieren incluir instrucciones específicas en un determinado documento HTML que completen las instrucciones y funciones que se incluyen por defecto en todos los documentos del sitio web.

El principal inconveniente es que si se quiere hacer una modificación en el bloque de código, es necesario modificar todas las páginas que incluyen ese mismo bloque de código JavaScript.

Definir JavaScript en un archivo externo

Las instrucciones JavaScript se pueden incluir en un archivo externo de tipo JavaScript que los documentos XHTML enlazan mediante la etiqueta <script>. Se pueden crear todos los archivos JavaScript que sean necesarios y cada documento XHTML puede enlazar tantos archivos JavaScript como necesite.

Ejemplo:

Archivo codigo.js

```
alert("Un mensaje de prueba");
```

Documento XHTML

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>Ejemplo de código JavaScript en el propio documento</title>
<script type="text/javascript" src="/js/codigo.js"></script>
</head>

<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

Este método requiere definir el atributo src, que es el que indica la URL correspondiente al archivo JavaScript que se quiere enlazar. Cada etiqueta <script> solamente puede enlazar un único archivo, pero en una misma página se pueden incluir tantas etiquetas <script> como sean necesarias.

Los archivos de tipo JavaScript son documentos normales de texto con la extensión .js, que se pueden crear con cualquier editor de texto como Notepad, Sublime Text, EmEditor, UltraEdit, etc.

La principal ventaja de enlazar un archivo JavaScript externo es que se simplifica el código HTML de la página, que se puede reutilizar el mismo código JavaScript en todas las páginas

del sitio web y que cualquier modificación realizada en el archivo JavaScript se ve reflejada inmediatamente en todas las páginas HTML que lo enlazan.

Incluir JavaScript en los elementos HTML

Este último método es el menos utilizado, ya que consiste en incluir trozos de JavaScript dentro del código HTML de la página:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>Ejemplo de código JavaScript en el propio documento</title>
</head>

<body>
<p onclick="alert('Un mensaje de prueba')">Un párrafo de texto.</p>
</body>
</html>
```

El mayor inconveniente de este método es que ensucia innecesariamente el código HTML de la página y complica el mantenimiento del código JavaScript. En general, este método sólo se utiliza para definir algunos eventos y en algunos otros casos especiales.

5. Sintaxis

La sintaxis de un lenguaje de programación se define como el conjunto de reglas que deben seguirse al escribir el código fuente de los programas para considerarse como correctos para ese lenguaje de programación.

La sintaxis de JavaScript es muy similar a la de otros lenguajes de programación como Java y C. Las normas básicas que definen la sintaxis de JavaScript son las siguientes:

- ***No se tienen en cuenta los espacios en blanco y las nuevas líneas:*** como sucede con HTML, el intérprete de JavaScript ignora cualquier espacio en blanco sobrante, por lo que el código se puede ordenar de forma adecuada para entenderlo mejor (tabulando las líneas, añadiendo espacios, creando nuevas líneas, etc.)
- ***Se distinguen las mayúsculas y minúsculas:*** al igual que sucede con la sintaxis de las etiquetas y elementos HTML. Sin embargo, si en una página HTML se utilizan indistintamente mayúsculas y minúsculas, la página se visualiza correctamente, siendo el único problema la no validación de la página. En cambio, si en JavaScript se intercambian mayúsculas y minúsculas el script no funciona.
- ***No se define el tipo de las variables:*** al crear una variable, no es necesario indicar el tipo de dato que almacenará. De esta forma, una misma variable puede almacenar diferentes tipos de datos durante la ejecución del script.
- ***No es necesario terminar cada sentencia con el carácter de punto y coma (;):*** en la mayoría de lenguajes de programación, es obligatorio terminar cada sentencia con el carácter ;. Aunque JavaScript no obliga a hacerlo, es conveniente seguir la tradición de terminar cada sentencia con el carácter del punto y coma (;).
- ***Se pueden incluir comentarios:*** los comentarios se utilizan para añadir información en el código fuente del programa. Aunque el contenido de los comentarios no se visualiza por pantalla, sí que se envía al navegador del usuario junto con el resto del script, por lo que es necesario extremar las precauciones sobre la información incluida en los comentarios (teniendo en cuenta que al ser un lenguaje del lado del cliente, el código de Javascript puede visualizarse con el Inspector del navegador o accediendo al Código fuente).
    - **JavaScript define dos tipos de comentarios:** los de una sola línea y los que ocupan varias líneas.

Ejemplo de comentario de una sola línea:

```
// a continuación se muestra un mensaje
alert("mensaje de prueba");
```

Los comentarios de una sola línea se definen añadiendo dos barras oblicuas (//) al principio de la línea.

Ejemplo de comentario de varias líneas:

```
/* Los comentarios de varias líneas son muy útiles
cuando se necesita incluir bastante información
en los comentarios */
alert("mensaje de prueba");
```

Los comentarios multilínea se definen encerrando el texto del comentario entre los símbolos /* y */.

Etiqueta noscript

Algunos navegadores no disponen de soporte completo de JavaScript, otros navegadores permiten bloquearlo parcialmente e incluso algunos usuarios bloquean completamente el uso de JavaScript porque creen que así navegan de forma más segura.

En estos casos, es habitual que si la página web requiere JavaScript para su correcto funcionamiento, se incluya un mensaje de aviso al usuario indicándole que debería activar JavaScript para disfrutar completamente de la página. El siguiente ejemplo muestra una página web basada en JavaScript cuando se accede con JavaScript activado y cuando se accede con JavaScript completamente desactivado.

El lenguaje HTML define la etiqueta <noscript> para mostrar un mensaje al usuario cuando su navegador no puede ejecutar JavaScript. El siguiente código muestra un ejemplo del uso de la etiqueta <noscript>:

```
<body>
<noscript>
  <p>Bienvenido a Mi Sitio</p>
  <p>La página que estás viendo requiere para su funcionamiento el uso de JavaScript.
Si lo has deshabilitado intencionadamente, por favor vuelve a activarlo.</p>
```

```
</noscript>
</body>
```

La etiqueta <noscript> se debe incluir en el interior de la etiqueta <body> (normalmente se incluye al principio de <body>). El mensaje que muestra <noscript> puede incluir cualquier elemento o etiqueta HTML.

document.write()

El objeto **document** es el que tiene el contenido de toda la página que se está visualizando. Esto incluye el texto, imágenes, enlaces, formularios, etc.

Gracias a este objeto vamos a poder añadir dinámicamente contenido a la página, o hacer cambios, según nos convenga.

write() es uno de los métodos de este objeto.

Uno de los comandos básicos de JavaScript es **document.write**.

Esto imprime el texto especificado en la página. Para imprimir texto literalmente, debemos el texto en comillas simples y entre paréntesis como en el siguiente ejemplo:

```
document.write('texto');
```

El código anterior imprimirá el mensaje "texto" en la página.

También se puede utilizar document.write para imprimir variables. En ese caso debemos introducir el nombre de la variable sin comillas.

Ejemplo:

```
var nombre= 'Guillermo'
var edad=36;

document.write(nombre + " tiene " + edad + " años");
```

Cuadros de Dialogo

Para seguir ampliando conocimientos vamos a sumar tres funciones: alert(), prompt() y confirm()

alert ()

Es un cuadro de diálogo que aparece y desvía la atención lejos de la ventana actual.

```
<script type="text/javascript">
        alert("Texto a mostrar")
</script>
```

Este tipo de ventanas de diálogo muestra un botón de aceptar, normalmente con el nombre Aceptar, que el usuario deberá pulsar para continuar.

```
<script type="text/javascript">
        alert("¡Bienvenido! Estamos en la Unidad 7 del Módulo de JavaScript.");
</script>
```

Las alertas de JavaScript son ideales para las siguientes situaciones:

- Si queremos estar absolutamente seguros que aparezca un mensaje antes de hacer nada en el sitio web.
- Para advertir al usuario acerca de algo. Por ejemplo, "la siguiente página contiene humor no apto para menores de 18 años de edad."
- Se produjo un error y desea informar al usuario sobre el problema.
- Cuando se pregunta a los usuarios para la confirmación de algún tipo de acción.

confirm ()

Realiza exactamente lo mismo, pero además obliga al usuario a seleccionar entre dos opciones, una positiva y otra negativa, que se devolverá como parámetro (boolean).
Su función es solicitar al usuario confirmación en la realización de una acción.

```
<script type="text/javascript">
        confirm("¿Desea volver al inicio de la página?")
</script>
```

Este ejemplo pide confirmación para cargar el resto de la página. Si se elige la opción Cancelar, se cierra la ventana (el navegador mostrará un cuadro de diálogo de confirmación).
Si se pulsa Aceptar, se muestra el resto de la página.

prompt ()

Muestra una caja de texto en la que el usuario puede introducir contenidos. También muestra dos botones, Aceptar y Cancelar.

```
<script type="text/javascript">
        prompt("Su equipo favorito es: ","San Lorenzo");
</script>
```

El segundo parámetro es el valor predeterminado para la caja de texto, que el usuario podrá modificar. Este cuadro de diálogo permite capturar datos introducidos por el usuario y realizar una acción en base a ellos.
Por ejemplo, podemos pedirle al usuario que introduzca su nombre, y después mostrarlo:

```
<!doctype html>
<html
<head>
<meta charset="utf-8" />
<title>Prompt</title>
</head>
<body>
<script type="text/javascript">
var nombre = prompt("Introduzca su nombre:","");
document.write("<h2>Bienvenido, " + nombre + "</h2>");
</script>
<p>Acá va el resto de la página...</p>
</body>
</html>
```

Lo que vimos:

En esta unidad realizamos un primer acercamiento Javascript y a sus conceptos básicos.

Lo que viene:

Comenzaremos a trabajar con variables y operadores.