Desarrollador de Aplicaciones Web

Programación Web I

2do Cuatrimestre (2013)

Departamento de Ingeniería e Investigaciones Tecnológicas

Javascript

Comisión Miércoles Noche
Ing. Gerardo Barbosa
Ing. Lucas Videla
Rubén Goria
Noelia Galvano

Comisión Viernes Mañana
Ing. Karina Avila DiÁrsico
Ing. Mariano D'Ortona

https://piazza.com/class/hko9dxh4r7rlm

Introducción

¿Qué es Javascript?

Es un lenguaje de programación que se utiliza principalmente para crear páginas con efectos dinámicos:

- Secciones / Textos aparecen y desaparecen
- Acciones que se activan al pulsar botones
- Ventanas con mensajes de aviso al usuario.
- etc.

Es un lenguaje de programación interpretado

- No es necesario compilar los programas para ejecutarlos.
- Se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

No tiene ninguna relación directa con el lenguaje de programación Java.
JScript (Microsoft) es una copia de JavaScript (pero con otro nombre por cuestiones legales).

Los archivos de tipo JavaScript son documentos normales de texto con la extensión **.js**, que se pueden crear con cualquier editor de texto.

Introducción

Especificación ECMA

El desarrollo empezó en 1996 y estuvo basado en JavaScript.

Actualmente aceptado como el estándar ISO 16262.

ECMA (European Computer Manufacturers Association) ha publicado varios estándares ECMAScript (ECMA-262)

1era Edición (Junio 1997)

2da Edición (Junio 1998)

3ra Edición (Diciembre 1999)

4ta Edición (Abandonada)

5ta Edición (Diciembre 2009)

http://www.ecma-international.org/publications/standards/Ecma-262.htm

Introducción

Cada navegador tiene extensiones propias al estándar ECMAScript, pero cualquier código que se adecue al estándar debería funcionar en todos ellos.

Empresa	Lenguaje
Sun / Oracle	Javascript
Microsoft	Jscript
Adobe	ActionScript

http://www.ecma-international.org/publications/standards/Ecma-262.htm

Conceptos

Script: Cada uno de los programas, aplicaciones o trozos de código creados con el lenguaje de programación JavaScript.

Sentencia: Cada una de las instrucciones que forman un script.

Palabras reservadas: Son las palabras (en inglés) que se utilizan para construir las sentencias de JavaScript y que por tanto no pueden ser utilizadas libremente.

Algunas de las palabras reservadas por JavaScript son:

break, case, catch, continue, default, delete, do, else, finally, for, function, if, in, instanceof, new, return, switch,this, throw, try, typeof, var, void, while, with.

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: El intérprete de JavaScript ignora cualquier espacio en blanco sobrante.

Se distinguen las mayúsculas y minúsculas: Si se intercambian mayúsculas y minúsculas el script no funciona.

No se define el tipo de las variables: 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 (;): 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 Una sola línea “//” o de varias líneas (entre /* y */)

JavaScript en xHTML

Incluirlo dentro del Archivo 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, no se recomienda

Para que la página XHTML resultante sea válida, es necesario añadir el atributo **type** a la etiqueta **<script>**

Los valores que se incluyen en el atributo type están estandarizados y para el caso de JavaScript, el valor correcto es **text/javascript**

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.

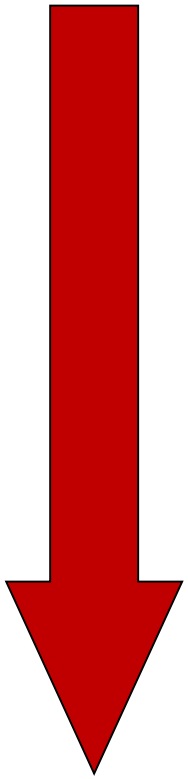
JavaScript en xHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <title>Ejemplo de código JavaScript en el propio documento</title>

    <script type="text/javascript">
      alert("Un mensaje de prueba");
    </script>

</head>

<body>

    <div>Hola Mundo.</div>

</body>

</html>
```

JavaScript en xHTML

Secuencia de Ejecución

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
      <title>Ejemplo de código JavaScript en el propio documento</title>

      <script>
                  document.write(“hola mundo”);

                  function escribirHolaMundo2() {
                              document.write(“hola mundo2”);
                  }
      </script>
</head>
<body>
      Hola mundo 3
      <script>
                   escribirHolaMundo2();
      </script>
</body>

</html>
```



JavaScript en xHTML

Incluirlo como archivo externo

Las instrucciones JavaScript se pueden incluir en un archivo externo de tipo JavaScript que los documentos XHTML enlazan mediante la etiqueta <script>.

Además del atributo **type**, 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.

La principal ventaja de enlazar un archivo JavaScript externo es que se simplifica el código XHTML 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 XHTML que lo enlazan.

JavaScript en xHTML

Archivo xHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /> <title>Ejemplo de
    código JavaScript en el propio documento</title>

    <script type="text/javascript" src="/js/codigo.js"></script>

</head>

<body>

    <div>Hola Mundo.</div>

</body>

</html>
```

Archivo /js/Codigo.js

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

JavaScript en xHTML

En los elementos xHTML

Consiste en incluir trozos de JavaScript dentro del código XHTML de la página:

El mayor inconveniente de este método es que *ensucia* innecesariamente el código XHTML 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.

JavaScript en xHTML

Archivo xHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /> <title>Ejemplo de
    código JavaScript en el propio documento</title>

    <script type="text/javascript" src="/js/codigo.js"></script>

</head>
<body>
    <input type=“button” Value=“Presionar” onclick=“javascript:Hola()”/>
</body>

</html>
```

Archivo /js/Codigo.js

```
function Hola () {
    alert("Un mensaje de prueba");
}
```

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 lenguaje HTML define la etiqueta **<noscript>** para mostrar un mensaje al usuario cuando su navegador no puede ejecutar JavaScript.

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

Variables

Una variable es un elemento que se emplea para almacenar y hacer referencia a otro valor.

Se crean mediante la palabra reservada **var**, pero no es necesario (aunque se recomienda hacerlo).

Si cuando se declara una variable se le asigna también un valor, se dice que la variable ha sido **inicializada**

El nombre de una variable también se conoce como identificador y debe cumplir las siguientes normas:

- Sólo puede estar formado por letras, números y los símbolos $ (dólar) y _ (guión bajo).
- El primer carácter no puede ser un número

```
var $numero1;
var _$letra = “L”;
```

Operadores

Los operadores permiten a los programas realizar cálculos complejos y tomar decisiones lógicas en función de comparaciones y otros tipos de condiciones.

- Asignación
  - =
  - +=, -=, *=, /=

```
a = 8;
b = 3;
a += 3;
document.write(a);          // a -> 11
a -= 2;
document.write(a);          // a -> 9
b *= 2;
document.write(b);          // b -> 6
```

Operadores

- Aritméticos
  - +
  - -
  - *
  - /
  - %
  - ++ (Incremento)
  - -- (decremento)

```
a = 8;
b = 3;
document.write(a++);   // Muestra 8 a→9
document.write(b--);   // Muestra 3 b→2
a = b++;               // a = 2, b = 3
a = ++b;               // a = 4, b = 4
a = --b;               // a = 3, b = 3
a = b--;               // a = 3, b = 2
```

Operadores

- Lógicos
  - Negación (!)
  - AND (&&)
  - OR (||)

```
a = 8;
b = 3;
c = 3;
document.write( (a == b) && (c > b) );      // false
document.write( (a == b) && (b == c) );     // false
document.write( (a == b) || (b == c) );     // true
document.write( (b <= c) );                 // true
document.write( !(b <= c) );                // false
```

Operadores

- Relacionales
  - >, >=
  - <, <=
  - ==
  - !=
  - === (Estrictamente igual)
  - !== (No estrictamente igual)

```
var numero1 = 3;
var numero2 = 5;
resultado = numero1 > numero2;      // resultado = false
resultado = numero1 < numero2;      // resultado = true
numero1 = 5; numero2 = 5;
resultado = numero1 >= numero2;    // resultado = true
resultado = numero1 <= numero2;    // resultado = true
resultado = numero1 == numero2;    // resultado = true
resultado = numero1 != numero2;     //  resultado = false
numero1 = 5; numero2 = “5”;
resultado = numero1 == numero2  // resultado = true
resultado = numero1 === numero2  // resultado = false
```

Estructura de Control de Flujo

Si se utilizan estructuras de control de flujo, los programas dejan de ser una sucesión lineal de instrucciones para convertirse en programas inteligentes que pueden tomar decisiones en función del valor de las variables.

- if
- If..else

```
var edad = 18;

if(edad >= 18) {
        alert("Eres mayor de edad");
}
else {
        alert("Todavía eres menor de edad");
}
```

Estructura de Control de Flujo

for

```
var dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"];

for(var i=0; i<7; i++) {
        alert(dias[i]);
}
```

- for .. In

```
var dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"];

for(i in dias) {
        alert(dias[i]);
}
```

Estructura de Control de Flujo

- While (*mientras se cumpla la condición indicada, repite indefinidamente las instrucciones incluidas dentro del bucle)*

```
var resultado = 0;
var numero = 100;
var i = 0;
while(i <= numero) {
        resultado += i;
        i++;
}
alert(resultado);
```

- do...while

```
do {
        var nombre = prompt("Ingrese su nombre (no puede ser vacio)");
} while (nombre=="");

alert("Bienvenido "+nombre);
```

Funciones y Propiedades básicas

¿Qué es una función?

Una función es un conjunto de instrucciones que se agrupan para realizar una tarea concreta y que se pueden reutilizar fácilmente.

```
function nombre_funcion(argumento1, argumento2, …., argumentoN) {

        ...

        return valor_a_retornar;
}
```

A continuación se presentarán las funciones y propiedades básicas de Javascript

Funciones y Propiedades básicas

Funciones útiles para cadenas de texto

- **length** (Calcula la longitud de una cadena de texto)

```
var mensaje = "Hola Mundo";
var numeroLetras = mensaje.length;    // Devuelve 10
```

- **+** o **concat()** (Se emplean para concatenar varias cadenas de texto)

```
var mensaje1 = "Hola";
var mensaje2 = " Mundo";
var mensaje = mensaje1 + mensaje2;              // mensaje = "Hola Mundo“
var mensaje3 = mensaje1.concat(mensaje2);    // mensaje3 = "Hola Mundo"
```

- **toUpperCase()** (Transforma todos los caracteres en mayúsculas)
- **toLowerCase()** (Transforma todos los caracteres de la cadena a sus correspondientes caracteres en minúsculas)

```
var mensaje1 = "Hola";
var mensaje2 = mensaje1.toUpperCase();          // mensaje2 = "HOLA“
var mensaje3 = mensaje1.toLowerCase();          // mensaje3 = "hola”
```

Funciones y Propiedades básicas

Funciones útiles para cadenas de texto

- **charAt(posicion)** (Obtiene el carácter que se encuentra en la posición indicada)

```
var mensaje = "Hola";
var letra = mensaje.charAt(0);          // letra = H
letra = mensaje.charAt(2);              // letra = l
```

- **indexOf(caracter)** (Calcula la posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si el carácter se incluye varias veces dentro de la cadena de texto, se devuelve su primera posición empezando a buscar desde la izquierda. Si la cadena no contiene el carácter, la función devuelve el valor -1)

```
var mensaje = "Hola Carola";
var posicion = mensaje.indexOf('a');    // posicion = 3
posicion = mensaje.indexOf('b');        // posicion = -1
```

JavaScript: Funciones y Propiedades básicas

Funciones útiles para cadenas de texto

- **lastIndexOf(caracter)** (calcula la última posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si la cadena no contiene el carácter, la función devuelve el valor -1. La posición devuelta se calcula empezando a contar desde el principio de la palabra)

```
var mensaje = "Hola Carola";
var posicion = mensaje.lastIndexOf('a');          // posicion = 10
posicion = mensaje.lastIndexOf('b');              // posicion = -1
```

- **substring(inicio, final)** (Extrae una porción de una cadena de texto. El segundo parámetro es opcional e indica la posición del carácter de corte en la cadena original. Si sólo se indica el parámetro inicio, la función devuelve la parte de la cadena original correspondiente desde esa posición hasta el final. Si se indica un inicio negativo, se devuelve la misma cadena original. Si se indica un final más pequeño que el inicio, JavaScript los considera de forma inversa, ya que automáticamente asigna el valor más pequeño al inicio y el más grande al final)

```
var mensaje = "Hola Mundo";
var porcion = mensaje.substring(2);       // porcion = "la Mundo"
porcion = mensaje.substring(5);           // porcion = "Mundo"
porcion = mensaje.substring(7,8);         // porcion = "n"
```

Funciones y Propiedades básicas

Funciones útiles para cadenas de texto

- **split(separador)** (Convierte una cadena de texto en un array de cadenas de texto. La función parte la cadena de texto determinando sus trozos a partir del carácter separador indicado)

```
var palabra = "Hola";
var letras = palabra.split("");                    // letras = ["H", "o", "l", "a"]
```

Funciones y Propiedades básicas

Funciones útiles para arrays

- **length** (Calcula el número de elementos de un array)

```
var vocales = ["a", "e", "i", "o", "u"];
var numeroVocales = vocales.length;                    // numeroVocales = 5
```

- **concat()** (Se emplea para concatenar los elementos de varios array)

```
var array1 = [1, 2, 3];
array2 = array1.concat(4, 5, 6);          // array2 = [1, 2, 3, 4, 5, 6]
array3 = array1.concat([4, 5, 6]);        // array3 = [1, 2, 3, 4, 5, 6]
```

- **join(separador)** (Es la función contraria a split(). Une todos los elementos de un array para formar una cadena de texto. Para unir los elementos se utiliza el carácter separador indicado)

```
var array = ["hola", "mundo"];
var mensaje = array.join("");             // mensaje = "holamundo"
mensaje = array.join(" ");                // mensaje = "hola mundo"
```

Funciones y Propiedades básicas

Funciones útiles para arrays

- **pop() (**Elimina el último elemento del array y lo devuelve. El array original se modifica y su longitud disminuye en 1 elemento)

```
var array = [1, 2, 3];
var ultimo = array.pop();          // ahora array = [1, 2], ultimo = 3
```

- **push()** (Añade un elemento al final del array. El array original se modifica y aumenta su longitud en 1 elemento. También es posible añadir más de un elemento a la vez)

```
var array = [1, 2, 3];
array.push(4);                     // ahora array = [1, 2, 3, 4]
```

- **shift()** (Elimina el primer elemento del array y lo devuelve. El array original se ve modificado y su longitud disminuida en 1 elemento)

```
var array = [1, 2, 3];
var primero = array.shift();                  // ahora array = [2, 3], primero = 1
```

Funciones y Propiedades básicas

Funciones útiles para arrays

- **unshift()** (Añade un elemento al principio del array. El array original se modifica y aumenta su longitud en 1 elemento. También es posible añadir más de un elemento a la vez)

```
var array = [1, 2, 3];
array.unshift(0);                    // ahora array = [0, 1, 2, 3]
```

- **reverse()** (Modifica un array colocando sus elementos en el orden inverso a su posición original)

```
var array = [1, 2, 3];
array.reverse();                     // ahora array = [3, 2, 1]
```

Funciones y Propiedades básicas

Funciones útiles para números

- **NaN** (del inglés, *"Not a Number"*. Indica un valor numérico no definido. Por ejemplo, la división 0/0)

```
var numero1 = 0;
var numero2 = 0;
alert(numero1/numero2);                    // se muestra el valor NaN
```

- **isNaN()** (Permite proteger a la aplicación de posibles valores numéricos no definidos)

```
var numero1 = 0;
var numero2 = 0;
if(isNaN(numero1/numero2)) {
        alert("La división no está definida para los números indicados");
}
else {
        alert("La división es igual a => " + numero1/numero2);
}
```

Funciones y Propiedades básicas

Funciones útiles para números

- **Infinity** (Hace referencia a un valor numérico infinito y positivo. También existe el valor –Infinity para los infinitos negativos)

```
var numero1 = 10;
var numero2 = 0;
alert(numero1/numero2);                    // se muestra el valor Infinity
```

- **toFixed(digitos)** (Devuelve el número original con tantos decimales como los indicados por el parámetro dígitos y realiza los redondeos necesarios. Se trata de una función muy útil por ejemplo para mostrar precios)

```
var numero1 = 4564.34567;
numero1.toFixed(2);          // 4564.35
numero1.toFixed(6);          // 4564.345670
numero1.toFixed();           // 4564
```

Ámbito de las variables

Variables locales

El ámbito de una variable (llamado "scope" en inglés) es la zona del programa en la que se define la variable. JavaScript define dos ámbitos para las variables: global y local.

```
function creaMensaje() {
        var mensaje = “Mensaje de prueba”;
}
creaMensaje();
alert(mensaje);
```

Al ejecutar el código anterior no se muestra ningún mensaje por pantalla. La razón es que la variable ”mensaje” se ha definido dentro de la función creaMensaje() y por tanto, es una variable local que solamente está definida dentro de la función.

Cualquier instrucción que se encuentre dentro de la función puede hacer uso de esa variable, pero todas las instrucciones que se encuentren en otras funciones o fuera de cualquier función no tendrán definida la variable “mensaje”.

Ámbito de las variables

Variables globales

Está definida en cualquier punto del programa (incluso dentro de cualquier función).

```
var mensaje = “Mensaje de prueba”;

function muestraMensaje() {
        alert(mensaje);
}
```

La variable ”mensaje” se ha definido fuera de cualquier función.

Este tipo de variables automáticamente se transforman en variables globales y están disponibles en cualquier punto del programa (incluso dentro de cualquier función)

Si en el interior de una función, las variables se declaran mediante var se consideran locales y las variables que no se han declarado mediante var, se transforman automáticamente en variables globales.

¿Qué sucede si una función define una variable local con el mismo nombre que una variable global que ya existe? En este caso, las variables locales prevalecen sobre las globales, pero sólo dentro de la función:

Ámbito de las variables

Variables globales

¿Qué sucede si dentro de una función se define una variable global con el mismo nombre que otra variable global que ya existe?
La variable global definida dentro de la función simplemente modifica el valor de la variable global definida anteriormente:

```
var mensaje = "gana la de fuera";

function muestraMensaje() {
        mensaje = "gana la de dentro";
        alert(mensaje);
}

alert(mensaje);        // gana de la fuera
muestraMensaje();  // gana la de adentro
alert(mensaje);        // gana la de adentro
```

Document Object Model (DOM)

DOM permite a los programadores web acceder y manipular las páginas XHTML como si fueran documentos XML (se diseñó originalmente para manipularlos de forma sencilla)

Se ha convertido en una utilidad disponible para la mayoría de lenguajes de programación (Java, PHP, JavaScript). Se diferencian en la forma de implementarlo.

Los navegadores web transforman automáticamente todas las páginas web en una estructura más eficiente de manipular.

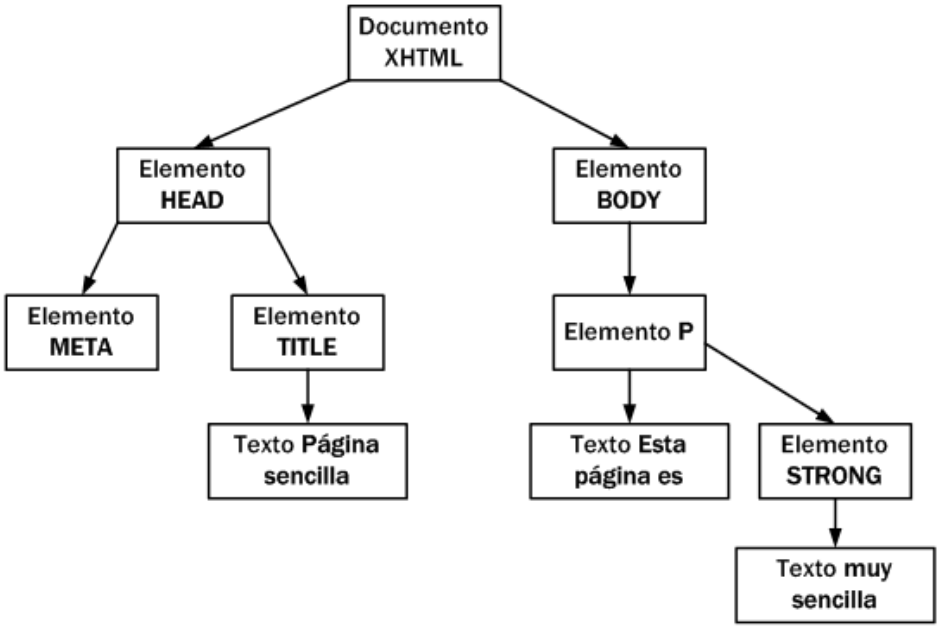
Esta transformación la realizan todos los navegadores de forma automática y nos permite utilizar las herramientas de DOM de forma muy sencilla. El motivo por el que se muestra el funcionamiento de esta transformación interna es que condiciona el comportamiento de DOM y por tanto, la forma en la que se manipulan las páginas.

DOM transforma todos los documentos XHTML en un conjunto de elementos llamados *nodos*, que están interconectados y que representan los contenidos de las páginas web y las relaciones entre ellos. Por su aspecto, la unión de todos los nodos se llama *"árbol de nodos"*.

Las páginas XHTML habituales producen árboles con miles de nodos. Aun así, el proceso de transformación es rápido y automático, siendo las funciones proporcionadas por DOM las únicas que permiten acceder a cualquier nodo de la página de forma sencilla e inmediata.

Document Object Model (DOM)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
        <title>Página sencilla</title>
</head>
<body>
        <p>Esta página es <strong>muy sencilla</strong></p>
</body>
</html>
```



Document Object Model (DOM)

Tipos de Nodos

La especificación completa de DOM define 12 tipos de nodos, aunque las páginas XHTML habituales se pueden manipular manejando solamente cuatro o cinco tipos de nodos:

1. Document, nodo raíz del que derivan todos los demás nodos del árbol.
2. Element, representa cada una de las etiquetas XHTML. Se trata del único nodo que puede contener atributos y el único del que pueden derivar otros nodos.
3. Attr, se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas XHTML, es decir, uno por cada par atributo=valor.
4. Text, nodo que contiene el texto encerrado por una etiqueta XHTML.
5. Comment, representa los comentarios incluidos en la página XHTML.

Los otros tipos de nodos existentes que no se van a considerar son DocumentType, CDataSection, DocumentFragment,Entity, EntityReference, ProcessingInstruction y Notation.

Document Object Model (DOM)

Tipo de Acceso a los nodos

Una vez construido automáticamente el árbol **completo** de nodos DOM, ya es posible utilizar las funciones DOM para acceder de forma directa a cualquier nodo del árbol. Es decir, su consulta, modificación y su eliminación solamente es posible después de que la página XHTML se cargue por completo.

DOM proporciona dos métodos alternativos para acceder a un nodo específico: acceso a través de sus nodos padre y acceso directo.

Las funciones que proporciona DOM para acceder a un nodo a través de sus nodos padre consisten en acceder al nodo raíz de la página y después a sus nodos hijos y a los nodos hijos de esos hijos y así sucesivamente hasta el último nodo de la rama terminada por el nodo buscado. Sin embargo, cuando se quiere acceder a un nodo específico, es mucho más rápido acceder directamente a ese nodo y no llegar hasta él descendiendo a través de todos sus nodos padre.

Document Object Model (DOM)

Acceso directo a los nodos

1) getElementsByTagName(nombreEtiqueta)

La función obtiene todos los elementos de la página XHTML cuya etiqueta sea igual que el parámetro que se le pasa a la función.

```
var parrafos = document.getElementsByTagName("p");
```

El valor devuelto es un array de nodos DOM, no un array de cadenas de texto o un array de objetos normales

2) getElementsByName()

Se buscan los elementos cuyo atributo “name” sea igual al parámetro proporcionado.

Internet Explorer 6.0 no implementa de forma correcta esta función, ya que sólo la tiene en cuenta para los elementos de tipo <input> y <img>. Además, también tiene en consideración los elementos cuyo atributo id sea igual al parámetro de la función.

Document Object Model (DOM)

Acceso directo a los nodos

3) getElementById()

Devuelve el elemento XHTML cuyo atributo id coincide con el parámetro indicado en la función. Como el atributo id debe ser único para cada elemento de una misma página, la función devuelve únicamente el nodo deseado.

```
var cabecera = document.getElementById("cabecera");
```

Es tan importante y tan utilizada en todas las aplicaciones web, que casi todos los ejemplos y ejercicios que siguen la utilizan constantemente.

Internet Explorer 6.0 también interpreta incorrectamente esta función, ya que devuelve también aquellos elementos cuyo atributo ”name” coincida con el parámetro proporcionado a la función.

var cabecera = document.getElementById("cabecera");

Document Object Model (DOM)

Creación de nodos

Creación de elementos XHTML simples

Por este motivo, crear y añadir a la página un nuevo elemento XHTML sencillo consta de cuatro pasos diferentes:

1. Creación de un nodo de tipo Element que represente al elemento.
2. Creación de un nodo de tipo Text que represente el contenido del elemento.
3. Añadir el nodo Text como nodo hijo del nodo Element.
4. Añadir el nodo Element a la página, en forma de nodo hijo del nodo correspondiente al lugar en el que se quiere insertar el elemento.

El proceso de creación de nuevos nodos puede llegar a ser tedioso, ya que implica la utilización de tres funciones DOM:

1. createElement(etiqueta): crea un nodo de tipo Element que representa al elemento XHTML cuya etiqueta se pasa como parámetro.
2. createTextNode(contenido): crea un nodo de tipo Text que almacena el contenido textual de los elementosXHTML.
3. nodoPadre.appendChild(nodoHijo): añade un nodo como hijo de otro nodo. Se debe utilizar al menos dos veces con los nodos habituales: en primer lugar se añade el nodo Text como hijo del nodo Element y a continuación se añade el nodo Element como hijo de algún nodo de la página.

Document Object Model (DOM)

Creación y Eliminación de nodos

```
// Crear nodo de tipo Element
var parrafo = document.createElement("p");

// Crear nodo de tipo Text
var contenido = document.createTextNode("Hola Mundo!");

// Añadir el nodo Text como hijo del nodo Element
parrafo.appendChild(contenido);

// Añadir el nodo Element como hijo de la pagina
document.body.appendChild(parrafo);
```

var cabecera = document.getElementById("cabecera");

Document Object Model (DOM)

Creación y Eliminación de nodos

Eliminación

Solamente es necesario utilizar la función removeChild().

```
var parrafo = document.getElementById("provisional");
parrafo.parentNode.removeChild(parrafo);


<p id="provisional">...</p>
```

La función removeChild() requiere como parámetro el nodo que se va a eliminar. Además, esta función debe ser invocada desde el elemento padre de ese nodo que se quiere eliminar.

Así, para eliminar un nodo de una página XHTML se invoca a la función removeChild() desde el valor parentNode del nodo que se quiere eliminar. Cuando se elimina un nodo, también se eliminan automáticamente todos los nodos hijos que tenga, por lo que no es necesario borrar manualmente cada nodo hijo.

var cabecera = document.getElementById("cabecera");

Document Object Model (DOM)

Acceso directo a los atributos XHTML

Mediante DOM, es posible acceder de forma sencilla a todos los atributos XHTML y todas las propiedades CSS de cualquier elemento de la página.

Los atributos XHTML de los elementos de la página se transforman automáticamente en propiedades de los nodos. Para acceder a su valor, simplemente se indica el nombre del atributo XHTML detrás del nombre del nodo.

```
var enlace = document.getElementById("enlace");

alert(enlace.href); // muestra http://www...com

<a id="enlace" href="http://www...com">Enlace</a>
```

Las propiedades CSS no son tan fáciles de obtener como los atributos XHTML. Para obtener el valor de cualquier propiedad CSS del nodo, se debe utilizar el atributo style.

```
var imagen = document.getElementById("imagen");

alert(imagen.style.margin);

<img id="imagen" style="margin:0; border:0;" src="logo.png" />
```

var cabecera = document.getElementById("cabecera");

Document Object Model (DOM)

Acceso directo a los atributos XHTML

La transformación del nombre de las propiedades CSS compuestas consiste en eliminar todos los guiones medios (-) y escribir en mayúscula la letra siguiente a cada guión medio.

- font-weight se transforma en fontWeight
- line-height se transforma en lineHeight
- border-top-style se transforma en borderTopStyle
- list-style-image se transforma en listStyleImage

El único atributo XHTML que no tiene el mismo nombre en XHTML y en las propiedades DOM es el atributo “class”. Como la palabra class está reservada por JavaScript, no es posible utilizarla para acceder al atributo class del elemento XHTML. En su lugar, DOM utiliza el nombre className para acceder al atributo class de XHTML

Desarrollador de Aplicaciones Web

Programación Web I

2do Cuatrimestre (2013)

Departamento de Ingeniería e Investigaciones Tecnológicas

Muchas gracias

Comisión Miércoles Noche
Ing. Gerardo Barbosa
Ing. Lucas Videla
Rubén Goria
Noelia Galvano

Comisión Viernes Mañana
Ing. Karina Avila DiÁrsico
Ing. Mariano D'Ortona

https://piazza.com/class/hko9dxh4r7rlm