



¿Qué es JavaScript?

- Lenguaje interpretado
- Sintaxis básica similar a Java y C++
- No tipado
- Puede funcionar como lenguaje procedimental y como lenguaje orientado a objetos

¿Qué es JavaScript? (continuación...)

Capacidades dinámicas:

- Construcción de objetos en tiempo de ejecución
- Listas con parámetros variables
- Variables pueden contener funciones

Javascript estandarizado: **ECMAScript**.

Distintas versiones:

- ECMAScript5.1 (2011) <http://caniuse.com/#feat=es5>
- ECMAScript6 (2015) <http://caniuse.com/#search=es6>
- ECMAScript 2016 / ES7 <http://kangax.github.io/compat-table/es2016plus/>

A partir de 2016, ECMA International promete actualizaciones anuales, identificadas por su año de publicación.


JavaScript - Ejemplo

Para indicarle al navegador que estamos usando Javascript, debemos usar los tags `<script>` y `</script>`

Pueden incluirse directamente en la página JSP (ó HTML) o puede importarse desde un archivo externo.

```
<html>
<body>
<script>
    //código Javascript
</script>
</body>
</html>
```

```
<html>
<head>
<script src="misScripts.js"/>
</head>
<body>.....</body>
</html>
```



El nombre del archivo externo donde está el código fuente javascript. Usualmente finaliza con .js

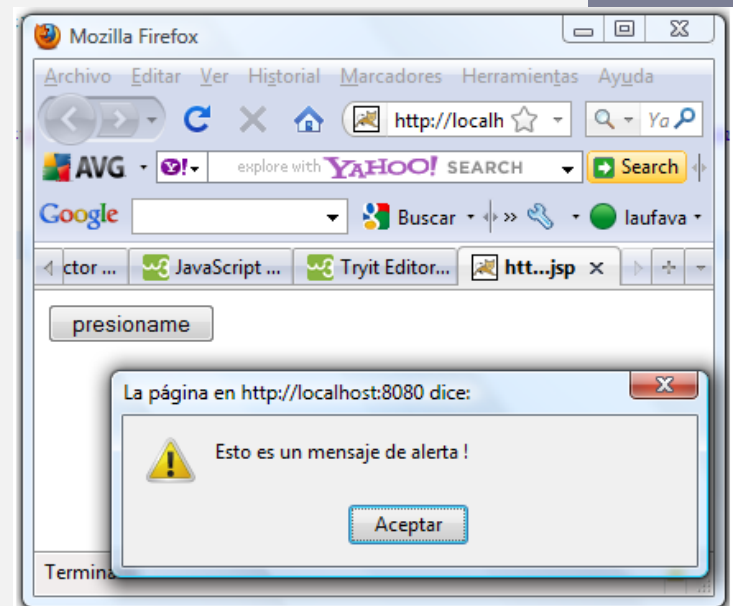
JavaScript - Ejemplo

Un primer ejemplo simple es mostrar una ventana de alerta al presionar un botón en una página JSP. Con javascript podemos implementar funciones que se invoquen ante determinados eventos.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
```

```
<html>
<head>
<script>
function mostrar_alerta() {
    alert("Esto es un mensaje de alerta !");
}
</script>
</head>
```

```
<body>
<input type="button" onclick="mostrar_alerta()" value = "presioname" />
</body>
</html>
```



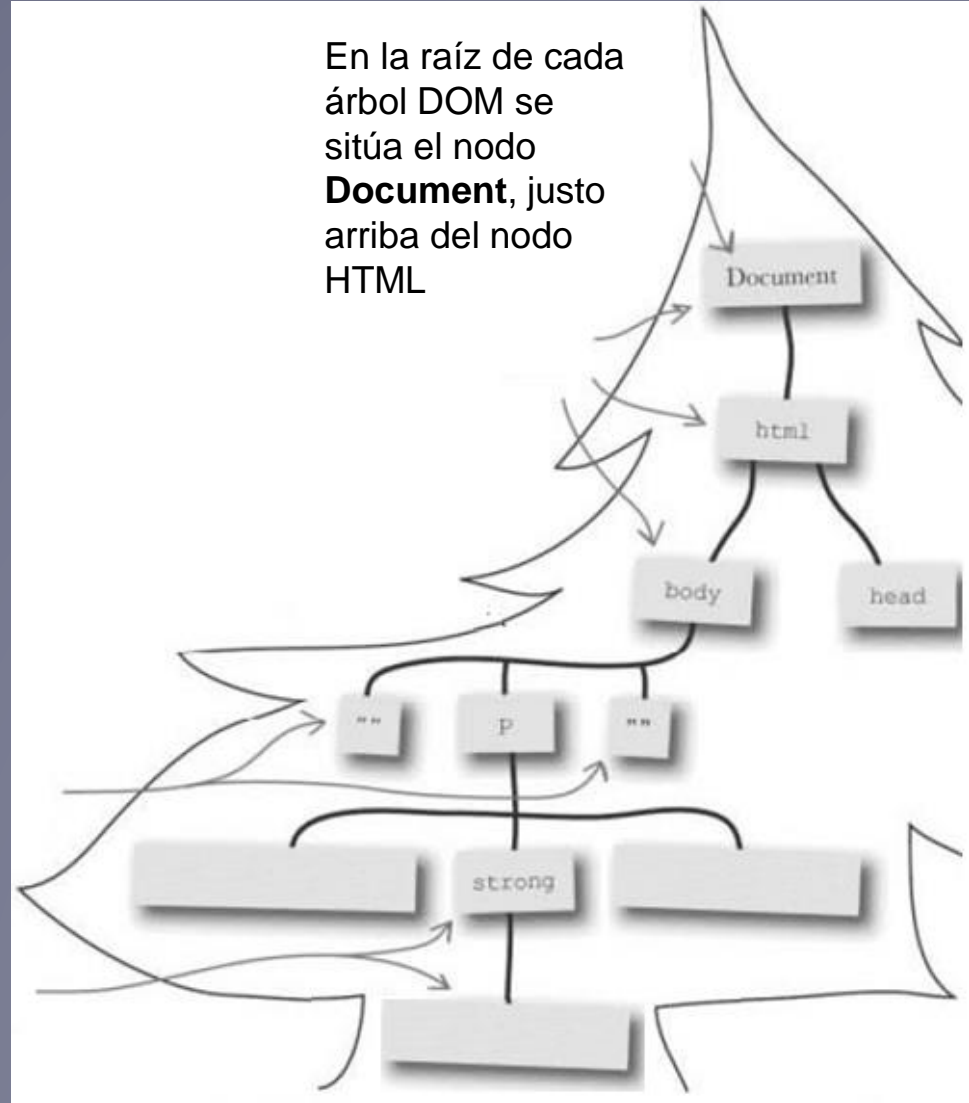
Ante un click del mouse sobre el botón, se invoca a la función **mostrar_alerta()**

JavaScript - DOM

DOM expone la página web al motor de Javascript. Las páginas web se exponen a través de una variable global document, que sirve como punto de partida para todas las manipulaciones DOM.

```
<html>
<head></head>
<body>
  <p id="censo">
    El <strong>censo
    2010</strong> sigue en marcha
    ...
  </p>
</body>
</html>
```

En la raíz de cada árbol DOM se sitúa el nodo **Document**, justo arriba del nodo HTML



JavaScript - DOM

La mayoría de las interacciones con el DOM comienzan con el objeto **document**, el cual es el nodo raíz.

El objeto **document** ofrece el método **getElementById()** que permite acceder desde JavaScript a los elementos de la página web. Para hacerlo esta función necesita el **ID** del elemento. Esta función no toma el dato del elemento directamente, sino que proporciona el campo HTML como un objeto de JavaScript. Para acceder a los datos se lo hace a través de la propiedad **value** del campo.

```
<input type="text" id="cantidad" name="cantidad" />
```

El atributo **id** es usado para acceder al campo del formulario desde el código JavaScript.

```
document.getElementById("cantidad")
```

```
document.getElementById("cantidad").value
```

El **value** nos permite acceder al dato

El **id** es la clave para acceder al elemento

cantidad: 18

JavaScript - DOM

También se puede tomar el contenido completo de un elemento HTML de una página web usando la propiedad `innerHTML`.

El **censo 2010**
sigue en marcha...

El `innerHTML` obtiene el
contenido del elemento,
incluyendo los tags HTML

```
<p id="censo">  
  El <strong>censo 2010</strong>  
  sigue en marcha...  
</p>
```

```
document.getElementById("censo").innerHTML
```

La propiedad `innerHTML` también puede usarse para setear contenido en una página. El nuevo contenido asignado reemplaza completamente al anterior.

```
document.getElementById("censo").innerHTML = "El <strong>censo</strong>finalizó";
```

El **censo** finalizó

Tipos de datos

```
var length = 16;           // Number
var bool = true;           // Boolean
var lastName = "Johnson"; // String
var cars = ["Saab", "Volvo", "BMW"]; // Array
var x = {firstName:"John", lastName:"Doe"}; // Object
var miFuncion = function(){ //hacer algo } // Function
```

- Tipado dinámico
- Strings pueden definirse con comillas simples o comillas dobles
- Los números son siempre de formato punto flotante (64 bits)
- Uso de *typeof*, *undefined* y *null*

```
var x = 16 + "Volvo";
x = "Volvo" + 16;
x = 16 + 4 + "La marca es 'volvo'";
x = 2 + 0.155;
```

Estructuras de control

```
if (time < 10) {  
    greeting = "Buen día";  
} else if (nombre == "José") {  
    greeting = "Buenas tardes";  
} else {  
    greeting = "Buenas noches";  
}
```

```
while (i < 10) {  
    text += "The number is " + i;  
    i++;  
}
```

```
var text = "";  
for (var i = 0; i < 5; i++) {  
    text += "El número es " + i + "\n";  
} // for
```

```
var text = "";  
var arreglo = [5, 9, 15, 47, 12];  
for (var i in arreglo) {  
    text += "El número es " + arreglo[i] + "\n";  
} // foreach
```

Funciones

```
function miFuncion() {  
    alert("Hola!");  
}
```

```
//otra forma equivalente  
var miFuncion = function() {  
    alert("Hola!");  
}
```

```
function miFuncionP(p1, p2) {  
    return p1 * p2;           // Función con parámetros  
}  
  
var otraFuncion = function(p1, p2, p3) {  
    p3 = p3 || 10;  
    return p1 * p2 * p3;  
}
```

- Las variables declaradas dentro de una función, se consideran locales a esa función. Cuando la función se completa, las mismas se eliminan de memoria.
- Los parámetros funcionan como variables locales a la función.

Arreglos

```
var cars = ["Saab", "Volvo", "BMW"];

var cars = [ //los espacios y saltos de línea no son tenidos en cuenta
  "Saab",
  "Volvo",
  "BMW"
];

var x = cars.length; //devuelve el tamaño del arreglo
var ultimo = cars.pop(); //elimina y retorna el último elemento
cars.push("Fiat"); //agrega un elemento al final del arreglo
var primero = cars.shift(); //elimina y retorna el primer elemento
cars.unshift("VW"); //agrega un elemento al principio del arreglo
var segundo = cars[1];
cars[1] = "Otro";
var pos = cars.indexOf("Fiat"); //retorna en qué posición se encuentra
```

Objetos

```
var msje = {  
  mensaje: "Texto mensaje",  
  autorNombre: "Juan",  
  autorApellido: "Gomez",  
  fecha: new Date(),  
  getNombreCompleto: function(){  
    return this.autorNombre+" "+this.autorApellido  
  }  
}
```

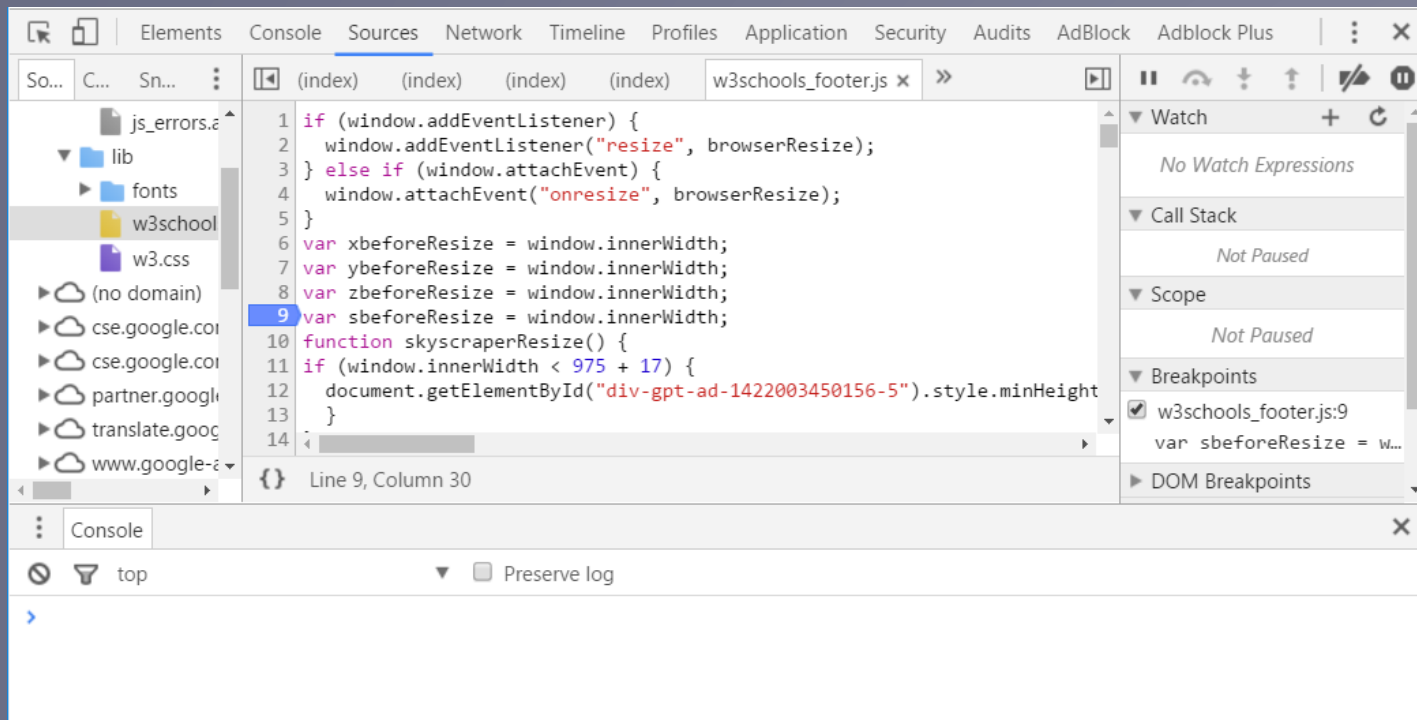
```
function Mensaje() {           //especificación clase Mensaje  
  this.mensaje;  
  this.autorNombre;  
  this.autorApellido;  
  this.fecha;  
  this.getNombreCompleto = function(){  
    return this.autorNombre+" "+this.autorApellido  
  }  
}  
  
var unMensaje = new Mensaje(); //se instancia un nuevo mensaje  
var otroMensaje = new Mensaje();  
  
unMensaje.nombre = "Juan"; // unMensaje["nombre"] = "Juan";  
...
```

Ref: http://www.w3schools.com/js/js_objects.asp /

[https://developer.mozilla.org/es/docs/Web/JavaScript/Introducci%C3%B3n a JavaScript orientado a objetos](https://developer.mozilla.org/es/docs/Web/JavaScript/Introducci%C3%B3n_a_JavaScript_orientado_a_objetos)

Debugging

- `console.log()`
- *debugger;*





Bootstrap 3

- Framework para el desarrollo del front-end de aplicaciones web de manera rápida y sencilla
- Incluye templates de base diseñados con HTML y CSS para el manejo de tipografía, formularios, tablas, navegación, cuadros de diálogo, diagramación, etc.
- Respeta estándares web
- Gratuito (Licencia MIT)



Ventajas:

- **Facilidad de uso:** solo requiere un conocimiento básico de HTML y CSS
- **Responsive y Mobile-first:** permite crear sitios que se ajusten automáticamente para verse bien en distintos dispositivos
- **Atractivo visual:** basado en diseños elaborados y reutilizables
- **Compatibilidad entre navegadores:** es compatible con todos los navegadores modernos (Chrome, Firefox, Internet Explorer, Safari, y Opera)

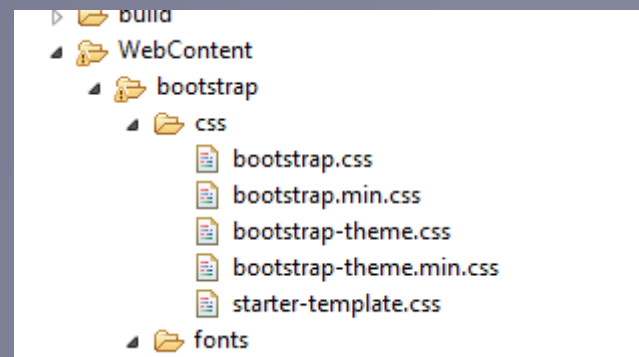


Instalación

- Descargar archivos precompilados de Bootstrap: <http://getbootstrap.com/getting-started/#download>
- Referenciar estilo css:

```
<link href="bootstrap/css/bootstrap.min.css" rel="stylesheet">
```
- Referenciar Javascript:

```
<script  
src="https://ajax.googleapis.com/ajax/libs/jquery/1.1  
1.3/jquery.min.js"></script>  
<script src="bootstrap/js/bootstrap.min.js"></script>
```
- Incluir archivos en el proyecto:



Un template básico...

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap 101 Template</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <!-- jQuery (necesario para los plugins Javascript de Bootstrap) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
    <!-- Incluir todos los plugins compilados (abajo), o incluir archivos individuales según sea necesario -->
    <script src="js/bootstrap.min.js"></script>
  </head>
  <body>
    <div class="container">
      <h1>Hola Mundo!</h1>
    </div>
  </body>
</html>
```

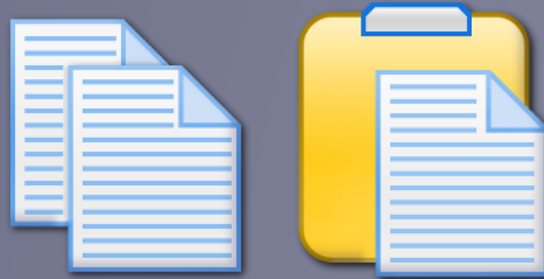
¿Cómo utilizar Bootstrap?

Sistema de grilla de 12 columnas

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints		
Container width	None (auto)	750px	970px	1170px
Class prefix	<code>.col-xs-</code>	<code>.col-sm-</code>	<code>.col-md-</code>	<code>.col-lg-</code>
# of columns	12			
Column width	Auto	~62px	~81px	~97px
Gutter width	30px (15px on each side of a column)			
Nestable	Yes			
Offsets	Yes			
Column ordering	Yes			

¿Cómo utilizar Bootstrap?

- Templates de ejemplo
- Sistema Grid de 12 columnas
- Markup API (Agregar clases al código html)
- Componentes
- JavaScript



Para probar con ejemplos

- [Grid System](#)
- [Imágenes](#)
- [Alertas](#)
- [Formularios](#)
- [Barra de navegación](#)
- [Glyphicon Components](#)
- [Bootstrap JS Carousel](#)
- [Bootstrap JS Tooltip](#)

[Ver todos los ejemplos W3C](#)

