



JavaScript: Un Curso Rápido

Parte I: Fundamentos

Traducción de Dr. Roberto Solís Robles

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Temas

- **Visión General**
- **Referencias**
- **Incrustación en el navegador**
- **Versiones de HTML**
- **Sintáxis Básica**
- **Arreglos**
- **Strings y expresiones regulares**



Intro

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Libros

- ***JavaScript the Definitive Guide***
 - De David Flanagan, O'Reilly. La una referencia verdaderamente completa sobre el lenguajeJavaScript. Completa y bien escrita.
- ***JavaScript: The Good Parts***
 - De Douglas Crockford, O'Reilly
 - Guia avanzada sobresaliente sobre las mejores prácticas en JavaScript, especialmente funciones, objetos y expresiones regulares.
- ***Pro JavaScript Techniques***
 - De John Resig (of jQuery fame), APress
 - Guía excelente sobre las mejores prácticas en Javascript, no es una referencia completa
- ***DOM Scripting***
 - De Jeremy Keith, FriendsOf Press
 - Se enfoca en manipular DOM y CSS

Referencias en Linea

- **JavaScript tutorial (language syntax)**
 - <http://www.w3schools.com/js/>
 - http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Guide
- **JavaScript API references (builtin objects)**
 - <http://www.w3schools.com/jsref/>
 - <http://www.devguru.com/technologies/ecmascript/QuickRef/>
 - <http://www.devguru.com/technologies/JavaScript/>
 - <http://www.javascriptkit.com/jsref/>
 - http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference
- **HTML DOM reference (with JavaScript Examples)**
 - http://www.w3schools.com/html/dom_reference.asp
- **Official ECMAScript specification**
 - <http://www.ecma-international.org/publications/standards/Ecma-262.htm>

Firebug

- **Instale Firebug en Firefox**
 - <http://getfirebug.com/>
- **Use la línea de comando de Firebug para pruebas interactivas**
 - <http://getfirebug.com/commandline>
- **Se puede usar Firebug Lite en IE, Opera, Chrome**
 - No es muy bueno, pero es mejor que nada
 - <http://getfirebug.com/firebuglite>



Incrustando JavaScript en HTML

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Carga de Scripts

- **script con src**

- `<script src="my-script.js" type="text/javascript"></script>`

- Propósito

- Definir funciones, objetos y variables.
 - Las funciones serán activadas posteriormente por botones, otros eventos de usuario, etiquetas de script en línea con el contenido de la sección body, etc.

- **script con el contenido de la sección body**

- `<script type="text/javascript">Codigo JavaScript </script>`

- Propósito

- Directamente invocar el código que se ejecutara conforme la página se carga
 - Por ejemplo, para mostrar contenido HTML construido en JavaScript
 - No use este enfoque para definir funciones o hacer cosas que pudiera ser hechas en archivos externos.
 - Mas lento (no hay caching del browser) y menos reusable

Ejemplo (phish.js)

```
function getMensaje() {  
    var cantidad = Math.round(Math.random() * 100000);  
    var mensaje =  
        "Ganaste $" + cantidad + "!\n" +  
        "Para recoger tus ganancias, envia tu numero de  
        tarjeta de credito\n" +  
        "y cuenta de banco a loteria@phisher.com.";  
    return(mensaje);  
}  
function muestraGanancias1() {  
    alert(getMensaje());  
}  
function muestraGanancias2() {  
    document.write("<h1><blink>" + getMensaje() +  
        "</blink></h1>");  
}
```

“alert” muestra un cuadro de dialogo

“document.write” inserta texto en la pagina en la ubicación actual

Ejemplo (carga_script.html)

```
<!DOCTYPE ...><html xmlns="http://www.w3.org/1999/xhtml">  
<head><title>Loading Scripts</title>
```

...

```
<script src="./scripts/phish.js"  
      type="text/javascript"></script>
```

Carga el script de la página anterior

```
</head>
```

```
<body>
```

...

```
<input type="button" value="Cuanto ganaste?"  
      onclick='muestraGanancias1()' />
```

Llama a muestraGanancias1 cuando el usuario presiona el botón. Pone el resultado en un cuadro de diálogo.

...

```
<script type="text/javascript">  
muestraGanancias2()</script>
```

...

```
</body></html>
```

Llama a muestraGanancias2 cuando la página se carga en el navegador. Pone el resultado en la ubicación actual.

Ejemplo(Resultados)

Cargando Scripts - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

HTML 4.0... JQuery F... eSAT - C... HTML X... Blue Flax... Downloa... Impleme... apache fr... Carga...

localhost/cursouaz/js_sintaxis/carga-script.html

Most Visited (0 unread) Yahoo! Mail,... Getting Started Latest Headlines https://www.facebook.c... Problem loading page

parallels workstation SEARCH

Cargando Scripts

Invocando Funcion desde el Boton

Cuanto ganaste?

Invocando Funcion desde Etiqueta script

Gasaste \$1344! Para recoger tus ganancias, envia tu numero de tarjeta de credito y cuenta de banco a loteria@phisher.com.

Gasaste \$8679!
Para recoger tus ganancias, envia tu numero de tarjeta de credito y cuenta de banco a loteria@phisher.com.

OK

All code from the [coreservlets.com J2EE tutorials \(servlets, JSP, Struts, JSF, Ajax, GWT, Spring, Hibernate, JPA, & Java 6 programming\)](#). There are also live instructor-led [training courses on the same J2EE topics \(servlets, JSP, Struts, JSF, Ajax, GWT, Spring, Hibernate, JPA, & Java 6 programming\)](#). Traducción de Dr. Roberto Solis Robles.

McAfee

Cargando Scripts: Casos Especiales

- **Defecto de Internet Explorer**
 - Scripts con src no cargan si usa `<script.../>`.
 - Debe usar `<script src="..." ...></script>`
- **XHTML: Scripts con contenido en body**
 - Es un error que el cuerpo del script contenga caracteres XML especiales tales como `&` o `<`
 - Por ejemplo, `<script...>if (a<b) { esto(); } else { aquello(); }</script>`
 - Por tanto, se usa la sección CDATA a menos que el contenido sea simple y no tenga caracteres especiales
 - `<script type="text/javascript"><![CDATA[
Codigo JavaScript
]]></script>`



Versiónes de HTML y JavaScript

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Resumen

- **XHTML**

- La versión más común usada en aplicaciones con Ajax apps o HTML dinámico (DHTML, aplicaciones JavaScript que manipulan el DOM)
- Sigue la sintaxis de XML, etiquetas en minúscula

- **HTML 5 (algo así)**

- Creciendo en popularidad para apps Ajax o DHTML.
- Version usada ahora es básicamente XHTML pero con un DOCTYPE más simple
 - No importa si el navegador soporta en realidad HTML 5

- **HTML 4**

- Muy común en apps que no usan JavaScript
- No recomendada para apps Ajax

XHTML

- **Resumen**

- Sigue la sintaxis XML. Etiquetas en minúscula, se requieren etiquetas de cierre, comillas alrededor de valores de atributos.

- **Estructura Basica**

```
<!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><title>...</title></head>
<body> ... </body></html>
```

- **Pros**

- El código corresponde muy directamente con la representación internal (DOM) que usa el navegador

- **Cons**

- DOCTYPE y etiqueta <html> inicial son largas y tediosas

Pseudo-HTML 5

- **Resumen**

- Siguel la sintaxis XML. Sintaxis XHTML (transitional) pero con un DOCTYPE y <html> inicial mas simples.

- **Estructura básica**

<!DOCTYPE html>

<html>

<head><title>...</title></head>

<body> ... </body></html>

- **Pros**

- El código corresponde muy directamente con la representación internal (DOM) que usa el navegador

- **Cons**

- No cumple estrictamente con la especificación. Puede generar advertencias de validadores formales, especialmente con formateo que no sea CSS.

HTML 4

- **Resumen**

- No sigue la sintaxis XML. Las etiquetas pueden ser en mayúscula o minúscula. Las etiquetas de cierre y comillas alrededor de valores de atributos son opcionales.

- **Estructura básica**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD><TITLE>...</TITLE></HEAD>
<BODY> ... </BODY></HTML>
```

- **Pros**

- Código simple. Ampliamente usado en apps que no usan Ajax.

- **Cons**

- Código fuente y representación interna del navegador pueden ser sustancialmente diferentes, requiriendo una traducción mental cuando se piensa en como manipular DOM desde JavaScript.



Sintaxis Básica de JavaScript

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Variables

- **Se declaran con “var”**
 - Tanto para variables globales como locales.
 - Argumentos de función no llevan “var”
- **No se declaran tipos de datos**
 - Algunas personas dicen que JavaScript es un lenguaje “sin tipos”, pero técnicamente es un lenguaje de “tipos dinámicos”
 - JavaScript es *muy* liberal sobre la conversión de tipos
- **Solo hay dos alcances(scopes)**
 - Alcance global
 - Hay que ser muy cuidados con este cuando se use Ajax.
 - Puede causar condiciones de competencia.
 - Alcance de función
 - **No** hay alcance de bloque como en Java

Operadores y Sentencias

- **Casi el mismo conjunto de operadores que en Java**
 - + (adición y concatenación de String), -, *, /
 - &&, ||, ++, --, etc
 - El == es como el “equals” de Java
 - El operador === (menos usado) es como el == de Java
- **Sentencias**
 - Los puntos y coma son técnicamente opcionales
 - Pero altamente recomendados
 - Consider
 - return x
 - return
x
 - No son identicos! El segundo regresa, y entonces evalua x. Debería actuar como si los puntos y coma fueran requeridos como en Java.
- **Comentarios**
 - Igual que en Java (/* ... */ y // ...)

Condicionales y Ciclos Simples

- **if/else**

- Casi idéntico a Java excepto que la prueba no necesitar ser estrictamente true/false
 - “false”: false, null, undefined, "" (string vacío), 0, NaN
 - “true”: cualquier otra cosa(incluyendo el string "false")

- **Ciclo for básico**

- Idéntico a Java excepto por las declaraciones de variable
 - for(**var** i=0; i<someVal; i++) { doLoopBody(); }

- **Ciclo while**

- Igual que en Java excepto que la prueba no necesita ser true/false estrictamente
 - while(someTest) { doLoopBody(); }

- **Ciclo do/while**

- Igual que en Java excepto por la prueba

Otros Condicionales y Ciclos

- **switch**

- Difiere de Java en dos formas
 - El “case” puede ser una expresión
 - Los valores no necesitan ser int

- **Ciclo for/in**

- Por encima, parece similar al for/each de Java, pero
 - Para los arreglos, los valores pueden ser índices del arreglo, no los valores del arreglo
 - Use este ciclo para objetos (para ver nombres de propiedad), no para arreglos!
 - Para objetos, los valores son los nombres de las propiedades
- `var persona = { primerNombre: “Pedro”, apellido: “Rivera”};`
`for(var propiedad in persona) {`
 `hazAlgoCon(persona[propiedad]);`
`}`

La clase Math

- **Casi idéntica a la de Java**

- Como en Java, los métodos son static (Math.cos, Math.random, etc.)
 - Como veremos posteriormente, estos no son *realmente* métodos static, pero su sintaxis es similar a la de los metodos static en Java.
- Como en Java, los logaritmos son base e, las funciones trigonométricas son en radianes

- **Funciones**

- Math.abs, Math.acos, Math.asin, Math.atan, Math.atan2, Math.ceil, Math.cos, Math.exp, Math.floor, Math.log, Math.max, Math.min, Math.pow, Math.random, Math.round, Math.sin, Math.sqrt, Math.tan

- **Constantes**

- Math.E, Math.LN10, Math.LN2, Math.LOG10E, Math.PI, Math.SQRT1_2, Math.SQRT2



Arreglos

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Fundamentos de Arreglos

- **Asignación en un solo paso**
 - `var primos = [2, 3, 5, 7, 11, 13];`
 - `var nombres = ["Juan", "Jaime", "Jose", "Jorge"];`
 - No hay coma después del ultimo elemento
- **Asignación en dos pasos**
 - `var nombres = new Array(4);`
`nombres[0] = "Juan";`

`...`
`nombres[3] = "Jorge";`
- **Indice inicial es 0 como en Java**
 - `for(var i=0; i<nombres.length; i++) {`
 `hazAlgoCon(nombres[i]);`
 `}`

Iterando sobre Arreglos en JavaScript

- **Ciclo for estilo Java**

- Practicamente igual que en Java. No olvide el “var”!
for(var i=0; i<arreglo.length; i++) {
 var valor = arreglo[i];
 hazAlgoCon(valor);
}

- **Ciclo for específico a JavaScript**

- Se apoya en el hecho de que un índice no existente en el arreglo resulta en un valor undefined (no en una excepción) y que undefined significa “false” en una prueba.
for(var i=0, valor; valor=arreglo[i]; i++) {
 hazAlgoCon(valor);
}

- **Ciclo for-in**

- *No* recomendado para iterar sobre arreglos normales.
 - Regresa índices, no valores
 - Los objetos parecidos a arreglos pueden tener propiedades extra

Mas Sobre Arreglos

- **Los arreglos puede ser poco densos**
 - `var nombres = new Array();`
`nombres[0] = "Juan";`
`nombres[100000] = "Jorge";`
- **Los arreglos pueden ser redimensionados**
 - Independientemente de como se crea el arreglo, puedes hacer lo siguiente:
 - `arreglo.length = nuevaLongitud;`
 - `arreglo[cualquierNumero] = nuevoValor;`
 - `arreglo.push(nuevoValor)`
- **Los arreglos tienen métodos**
 - `push`, `pop`, `join`, `reverse`, `sort`, `concat`, `slice`, `splice`, etc.
 - Consulte la referencia del API
- **Los objetos regulares pueden ser tratados como arreglos**
 - Puede usar números (índices) como propiedades del objeto

Ejemplo

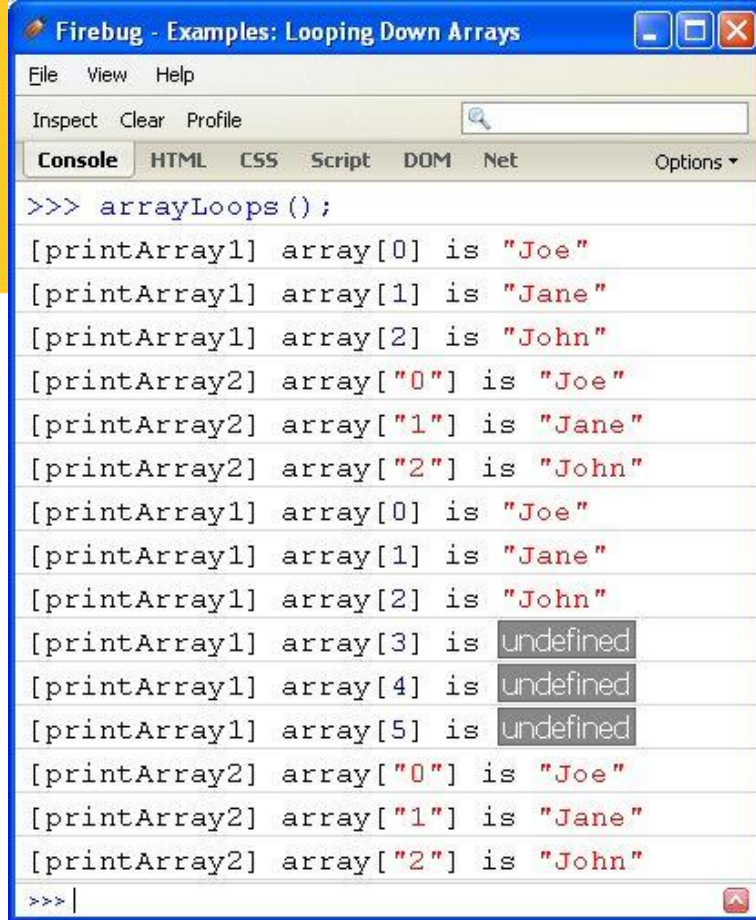
```
function arrayLoops() {  
    var names =  
        ["Joe", "Jane", "John"];  
    printArray1(names);  
    printArray2(names);  
    names.length = 6;  
    printArray1(names);  
    printArray2(names);  
}
```

```
function printArray1(array) {  
    for(var i=0; i<array.length; i++) {  
        console.log("[printArray1] array[%o] is %o", i, array[i]);  
    }  
}
```

```
function printArray2(array) {  
    for(var i in array) {  
        console.log("[printArray2] array[%o] is %o", i, array[i]);  
    }  
}  
arrayLoops();
```

console.log es una forma de mostrar salida en la ventana de Consola de Firebug. Solo para pruebas/depuración.

Llamada directa para pruebas interactivas en la consola de Firebug. (Pase todo el código a la línea de comando de la consola)





Strings y Expresiones Regulares

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Fundamentos de Strings

- **Puede usar comillas dobles o sencillas**
 - `var nombres = ["Juan", 'Jaime', "Jose", 'Jorge'];`
- **Puede acceder la propiedad length**
 - Por ejemplo, `"foobar".length` regresa 6
- **Los números pueden ser convertidos a strings**
 - Conversión automática durante las concatenaciones.
 - `var val = 3 + "abc" + 5; // Resultado es "3abc5"`
 - Conversión con precisión fija
 - `var n = 123.4567;`
`var val = n.toFixed(2); // Resultado es 123.46 (no 123.45)`
- **Los strings se pueden comparar con ==**
 - `"foo" == 'foo'` regresa true
- **Los strings pueden ser convertidos a números**
 - `var i = parseInt("37 blah"); // Resultado es 37 – ignora blah`
 - `var d = parseFloat("6.02 blah"); // Ignora blah`

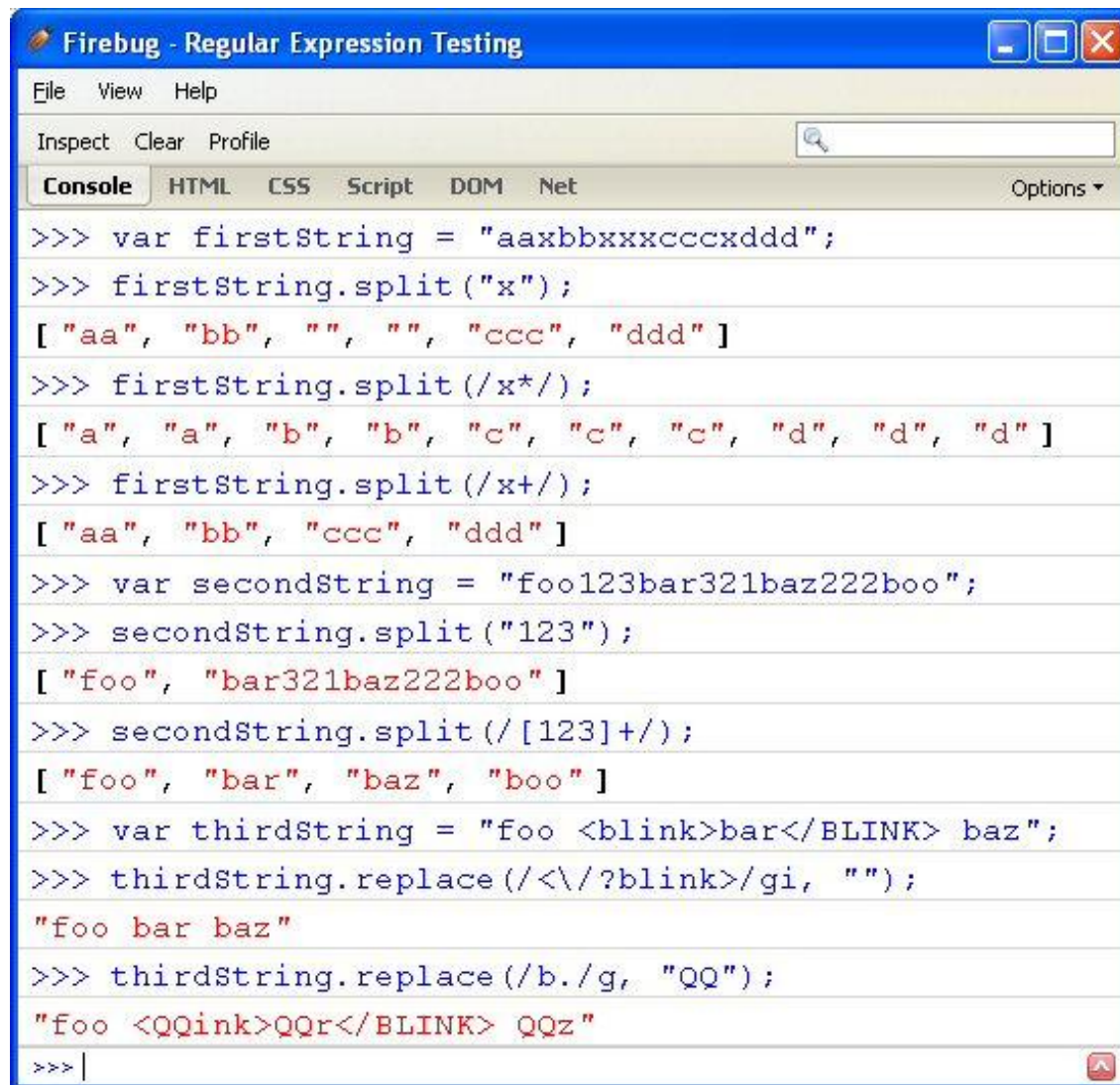
Métodos Centrales de String

- **Métodos simples similares a Java**
 - charAt, indexOf, lastIndexOf, substring, toLowerCase, toUpperCase
- **Métodos que usan expresiones regulares**
 - match, replace, search, split
- **Métodos HTML**
 - anchor, big, bold, fixed, fontcolor, fontsize, italics, link, small, strike, sub, sup
 - “prueba”.bold().italics().fontcolor("red") regresa '`<i>prueba</i>`'
 - Estos son métodos no estándar, pero son soportados en todos los navegadores principales
 - Preferible construir strings HTML de manera explícita

Expresiones Regulares

- **Especifica una expreg con /pattern/**
 - *No* con un String como en Java
- **La mayoría de los caracteres especiales iguales que en Java/Unix/Perl**
 - ^, \$, . – inicio, fin de string, cualquier otro caracter
 - \ – quite el significado a un caracter especial
 - *, +, ? – 0 o mas, 1 o mas, 0 o 1 ocurrencias
 - {n}, {n,} – exactamente n, n o mas ocurrencias
 - [] – agrupamiento
 - \s, \S – espacio en blanco, no espacio en blanco
 - \w, \W – caracter de palabra(letra o numero), no caracter de palabra
- **Modificadores**
 - /patron/g – realiza concordancia global (encuentra todas las concordancias, no solo la primera)
 - /patron/i – realiza concordancia sin hacer diferencia entre mayusculas/minusculas
 - /patron/m – realiza concordancia multilínea

Expresiones Regulares: Ejemplos



```
Firebug - Regular Expression Testing
File View Help
Inspect Clear Profile
Console HTML CSS Script DOM Net Options
>>> var firstString = "aaxbbxxxcccddd";
>>> firstString.split("x");
["aa", "bb", "", "", "ccc", "ddd"]
>>> firstString.split(/x*/);
["a", "a", "b", "b", "c", "c", "c", "d", "d", "d"]
>>> firstString.split(/x+/);
["aa", "bb", "ccc", "ddd"]
>>> var secondString = "foo123bar321baz222boo";
>>> secondString.split("123");
["foo", "bar321baz222boo"]
>>> secondString.split(/[123]+/);
["foo", "bar", "baz", "boo"]
>>> var thirdString = "foo <blink>bar</BLINK> baz";
>>> thirdString.replace(/<\/?blink>/gi, "");
"foo bar baz"
>>> thirdString.replace(/b./g, "QQ");
"foo <QQink>QQr</BLINK> QQz"
>>> |
```

Mas Información sobre Expresiones Regulares

- **Referencias a la API en línea dadas anteriormente (Vea la clase RegExp)**
 - http://www.w3schools.com/jsref/jsref_obj_regexp.asp
 - <http://www.devguru.com/technologies/ecmascript/QuickRef/regexp.html>
- **Tutoriales sobre Expresiones Regulares**
 - http://www.evolt.org/article/Regular_Expressions_in_JavaScript/17/36435/
 - <http://www.javascriptkit.com/javatutors/re.shtml>



Resumen

- **Use Firebug para probar y depurar**
- **Guarde las referencias en un bookmark**
 - <http://www.w3schools.com/js>
- **Incrustación en el navegador**
 - `<script src="blah.js" type="test/javascript"></script>`
 - Use sintaxis XHTML o pseudoHTML 5
- **Sintaxis básica de JavaScript**
 - Declare variables locales con var. No hay declaración de tipos.
 - Ciclos y condicionales similares a Java.
- **Arreglos en JavaScript**
 - Los arreglos son muy diferentes a los de Java. Pueden tener propiedades extra. Se pueden redimensionar. Pueden ser poco densos.



Preguntas?

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.