

JavaScript





Porque programación cliente?

- El PHP permite la creación de paginas dinámicas. A que nos sirve la programación de cliente?
- Beneficios programación cliente:
 - Eficiencia : permite modificar una pagina sin tener que esperar la respuesta del cliente (interficie mas rápida)
 - Respuesta a los eventos: puede responder a las acciones del usuario como click y presión de teclas.
- Beneficio programación servidor:
 - Seguridad: permite acceder a datos privados del servidor; el cliente no puede ver el código fuente.
 - Compatibilidad: no depende de las compatibilidad del browser
 - Flexibilidad: permite escribir ficheros, abrir conexiones a servidores, y bases de datos, ...



Que es JavaScript?

- Un lenguaje de programación usado para crear paginas interactivas
 - Añade texto dinámico en el HTML (ej : nombre de usuario)
 - Reacciona a eventos (ej: clic del usuario)
 - Recupera informaciones del ordenador del usuario (ej: tipo de browser)
 - Ejecuta cálculos en el ordenador del usuario (ej: validación de form)
 - Es un standard web (pero no suportado de la misma forma por todos los browsers)
 - No esta relacionado con Java, a parte alguna similitudes sintáctica

```
<script>  
alert("hola mundo");  
</script>
```

Incrustar código JavaScript (JS)



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

- El código JS puede estar en un fichero externo o incrustado en el HTML adentro de los tags `<script>`



Variables (ejemplos)

```
var clientName = "Connie Client";  
var enrollment = 99;  
var medianGrade = 2.8;  
var credits = 5 + 4 + (2 * 3);  
var s = "Connie Client";  
var fName = s.substring(0, s.indexOf(" ")); //  
    "Connie"  
var len = s.length; // 13
```



métodos (mas ejemplos)

- métodos: [charAt](#), [charCodeAt](#), [fromCharCode](#), [indexOf](#), [lastIndexOf](#), [replace](#), [split](#), [substring](#), [toLowerCase](#), [toUpperCase](#)
- Para conversiones entre números y strings:

```
var count = 10;  
var s1 = "" + count;           // "10"  
var s2 = count + " bananas!";  // "10 bananas"  
var n1 = parseInt("42 is the answer"); // 42  
var n2 = parseFloat("hello");   // NaN
```

- Para acceder a caracteres de un string:

```
var firstLetter = s[0];  
var firstLetter = s.charAt(0);  
var lastLetter = s.charAt(s.length - 1);
```



Bucles For y if/else

- for (*initialization*;
condition; *update*) {
 código; }

```
var sum = 0;
```

```
for (var i = 0; i < 100; i++)  
{ sum = sum + i; }
```

```
if (condition) {  
    código;  
} else if (condition) {  
    código;  
} else {  
    código;  
}
```



Arrays

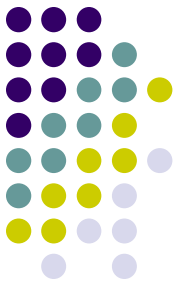
```
var name = []; // empty array
var name = [value, value, ..., value]; // pre-filled
name[index] = value; // store element
```

Ejemplos:

```
var ducks = ["Huey", "Dewey", "Louie"];

var stooges = []; // stooges.length is 0
stooges[0] = "Larry"; // stooges.length is 1
```


métodos



- `var a = ["Stef", "Jason"];`
`// Stef, Jason`
- `a.push("Brian");`
`// Stef, Jason, Brian`
- `a.unshift("Kelly");`
`// Kelly, Stef, Jason, Brian`
- `a.pop();`
`// Kelly, Stef, Jason`
- `a.shift();`
`// Stef, Jason`
- `a.sort();`
`// Jason, Stef`

- Split / Join

```
var s = "the quick brown fox";  
var a = s.split(" ");  
// ["the", "quick", "brown", "fox"]  
a.reverse();  
// ["fox", "brown", "quick", "the"]  
s = a.join("!");  
// "fox!brown!quick!the"
```

PD: Estilo programacion a objetos `a.metodo()`;

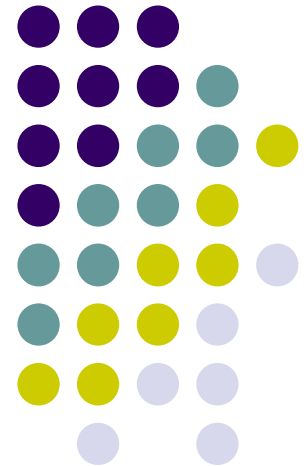


funciones

- El código JS en general permite describir funciones

```
function myFunction() {  
    alert("Hello!");  
    alert("How are you?");  
}
```

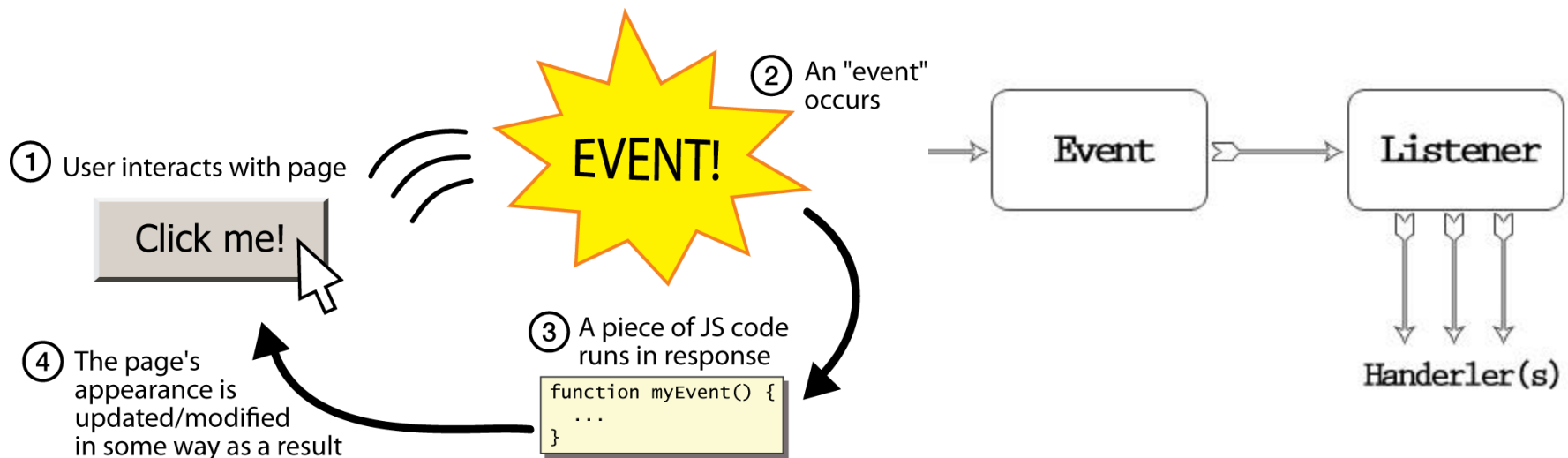
Eventos





Programacion a evento

- Los programas de JS no contienen un “main”. Es un codigo que reacciona a una accion del usuario llamada “evento”





Handler del evento

- Cuando se interacciona con un elemento, la funcione se ejecuta
- Onclick es un evento asociado a un botón. Existen muchos mas.

```
<button>Click me!</button>
```

```
<div onclick="myFunction();" >Click me!</div>
```



Como programar a eventos

`<button>Click me!</button>`

Para crear un botón activo, o otros controles UI:

- Elegir el control (ej. el botón) y el evento (ej: clic del mouse)
- Escribir una función JavaScript que se lanzara cuando pasa el evento.
- Acoplar la función del evento al control.

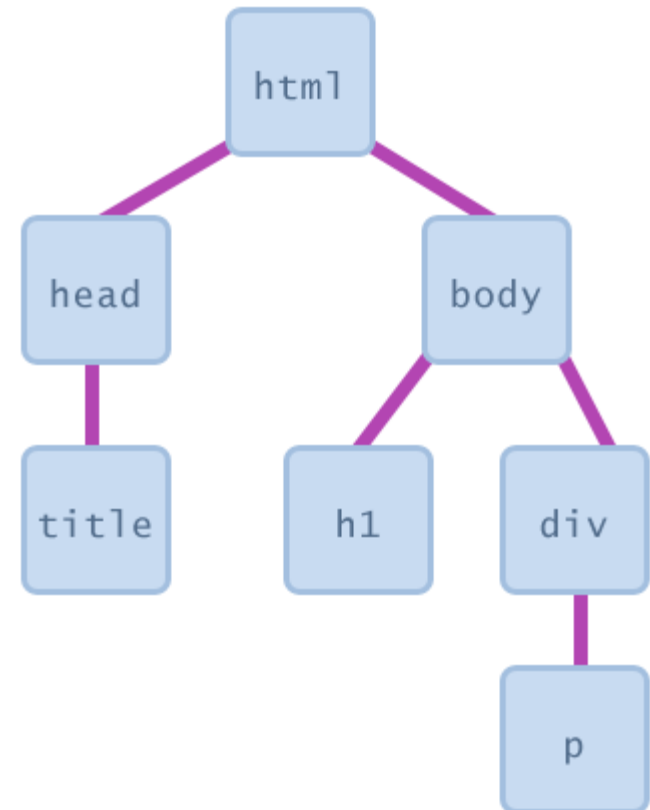
Ejemplo: Una alerta que crea una ventana pop-up puede molestar al usuario

- Una solución mejor podría ser de tener un mensaje que aparece en la pagina...



Document Object Model

- La mayoría del código JS modifica elementos de una pagina HTML
- Podemos examinar el estado de elementos
 - ej. mirar se un box esta marcado
- Podemos cambiar su estado
 - ej. Insertar nuevo texto en un div
- Podemos cambiar su estado
 - ej. Hacer un párrafo rojo





Document Object Model

- Cada elemento de una pagina tiene su objeto DOM correspondiente
- Se puede acceder a cualquier elemento de la pagina con *objectName.attributeName*

HTML

```
<p>
  Look at this octopus:
  
  Cute, huh?
</p>
```

DOM Element Object

Property	Value
tagName	"IMG"
<u>src</u>	"octopus.jpg"
alt	"an octopus"
id	"icon01"

JavaScript

```
var icon = document.getElementById("icon01");
icon.src = "kitty.gif";
```


getElementById (como recuperar una variable)



```
var name = document.getElementById("id");
```

```
<button onclick="changeText();">Click me!</button>
```

```
<input id="output" type="text" value="replace me" />
```

```
<script>
```

```
function changeText() {
```

```
    var textbox = document.getElementById("output");
```

```
    textbox.value = "Hello, world!";
```

```
}
```

```
</script>
```

Click me!	replace me
-----------	------------

Click me!	Hello, world!
-----------	---------------



DOM propiedades de objetos

```
<div id="main" class="foo bar">
  <p>See our <a href="sale.html" id="saleslink">Sales</a>today!</p>
  
</div>
```

```
var mainDiv = document.getElementById("main");
var icon    = document.getElementById("icon");
var theLink = document.getElementById("saleslink");
```

Propiedad	Descripción	Ejemplo
tagName	El tag del elemento HTML	mainDiv.tagName es un "DIV"
className	La clase CSS del elemento	mainDiv.className es "foo bar"
innerHTML	El contenido del elemento	mainDiv.innerHTML es "\n <p>See our <a hr...
src	El URL target de una imagen	icon.src es "images/borat.jpg"
href	URL target of a link	theLink.href es "sale.html"

DOM propiedades del formulario



//HTML

```
<input id="sid" type="text" size="7" maxlength="7" />
```

```
<input id="frosh" type="checkbox" checked="checked" /> Freshman?
```

```
<script>
```

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

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

```
</script>
```

 ☒ Freshman?

Propiedad	Descripción	Ejemplo
value	El texto/valor elegido por el usuario	sid.value puede ser "1234567"
checked	Si un box es activado	frosh.checked es true
disabled	Si un control es desactivo (boolean)	frosh.disabled es false
readOnly	Si un box es read-only	sid.readOnly es false



La propiedad innerHTML

```
# HTML
<button onclick="addText();" >Click me!</button>
<span id="output">Hello </span>

# JS
function addText() {
    var span = document.getElementById("output");
    span.innerHTML += " bro";
}
```

Click me! Hello

- La propiedad innerHTML modifica el texto de un argumento



Errores comunes en DOM

```
<button id="clickme">Color Me</button>
```

```
window.onload = function() {  
    document.getElementById("clickme").onclick =  
    changeColor;  
};  
function changeColor() {  
    var clickMe = document.getElementById("clickme");  
    clickMe.style.color = "red";  
}
```



Errores comunes en DOM

- Para cambiar un estilo se usa `.style`

```
var clickMe = document.getElementById("clickme");
```

NO-> `clickMe.color = "red";`

SI-> `clickMe.style.color = "red";`

- Las propiedades de estilo son mayusculas y se escriben:

NO-> `clickMe.style.font-size = "14pt";`

SI-> `clickMe.style.fontSize = "14pt";`

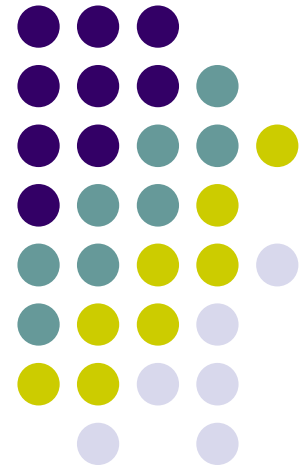
- La propiedad DEBE ser un strings, y a veces con unidades al final

NO-> `clickMe.style.width = 200;`

SI-> `clickMe.style.width = "200px";`

SI-> `clickMe.style.padding = "0.5em";`

Objetos globales del DOM





Objetos globales del DOM

- En JS podemos referenciar los siguientes objetos globales:

nombre	descripción
document	La pagina HTML corriente y su contenido
history	La lista de paginas que el usuario ha visitado
location	La URL de la pagina HTML actual
navigator	Informaciones respecto al navegador que se esta usando
screen	Información con respecto a la área de la pantalla utilizada por el navegador
window	La ventana del navegador



El objeto Window

- Representa las propiedades de la ventana de la pagina web (el objeto mas alto del DOM)
- Las propiedades asociadas son:
 - `document`, `history`, `location`, `name`
- Los métodos
 - `alert`, `confirm`, `prompt` (box popup)
 - `setInterval`, `setTimeout`, `clearInterval`, `clearTimeout` (timers)
 - `open`, `close` (abriendo una nueva ventana popping)
 - `blur`, `focus`, `moveBy`, `moveTo`, `print`, `resizeBy`, `resizeTo`, `scrollBy`, `scrollTo`



Ejemplo

- Apertura de una ventana popup

```
window.open("http://example.com/bar.html", "mi ventana de ejemplo",  
"width=900,height=600,scrollbars=1");
```

- P.d. Algunos programas blocker de popup impiden la ejecucion de este codigo



El objeto documento

- Representa la pagina web actual y los elemento que están contenidos.
- Las propiedades asociadas son:
anchors, body, cookie, domain, forms, images, links, referrer, title, URL
- Los métodos
 - getElementById
 - getElementsByName, getElementsByTagName
 - querySelector, querySelectorAll
 - close, open, write, writeln



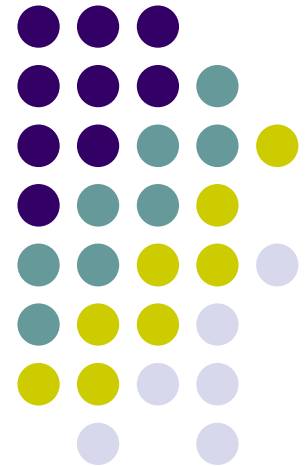
El objeto navigation

- Informaciones sobre el navegador
- Las propiedades asociadas son:
- `appName`, `appVersion`, `browserLanguage`, `cookieEnabled`, `platform`, `userAgent`
- Ejemplo de uso:

```
if (navigator.appName === "Microsoft Internet Explorer") { ...
```

Interacciones con JS

Eventos-Handlers





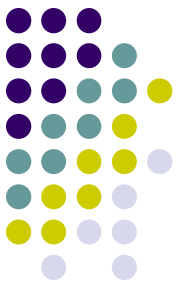
Eventos del Mouse

- El evento `onclick` es solo uno de muchos posible eventos que pueden ser gestionados
- Otros ejemplos de handlers:
 - `abort`, `blur`, `change`, `click`, `dblclick`, `error`, `focus`, `keydown`, `keypress`, `keyup`, `load`, `mousedown`, `mousemove`, `mouseout`, `mouseover`, `mouseup`, `reset`, `resize`, `select`, `submit`, `unload`



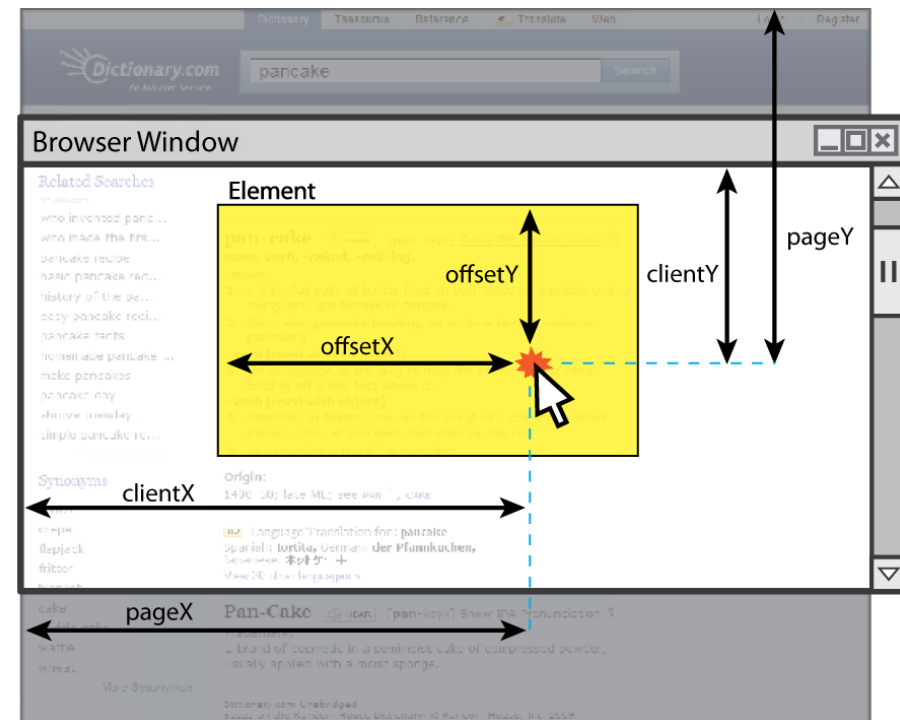
Eventos del Mouse

- | | |
|--------------|---|
| 1. Click | 1. El usuario presiona/libera el botón del mouse en el elemento |
| 2. dblclick | 2. El usuario presiona/libera el botón del mouse dos veces |
| 3. mousedown | 3. El usuario presiona el botón |
| 4. mouseup | 4. El usuario libera el botón en el elemento |
| 5. mouseover | 5. El mouse entra en un box de un elemento |
| 6. mouseout | 6. El mouse sale de un box de un elemento |
| 7. mousemove | 7. El mouse se mueve adentro de un box de un elemento |



Posicionamiento mouse

- `clientX`, `clientY`
coordenadas en el “browser window”
- `screenX`, `screenY`
coordenadas de la pantalla
- `offsetX`, `offsetY`
coordenadas de un unico objeto
- `pointerX()`, `pointerY()`
coordenadas de la pagina web entera
- `isLeftClick()` verdadero si el botón izquierdo esta presionado





Ejemplo

```
<pre id="target">pasa el mouse sobre esta zona!</pre>
```

```
window.onload = function() {  
  var target = document.getElementById("target");  
  target.onmousemove = target.onmousedown = showCoords;  
};
```

```
function showCoords(event) {  
  document.getElementById("target").innerHTML =  
    + "screen : (" + event.screenX + ", " + event.screenY + ")\n"  
    + "client : (" + event.clientX + ", " + event.clientY + ")\n"  
    + "button : " + event.button;  
}
```

```
screen : (333, 782)  
client : (222, 460)  
button : 0
```

Eventos relacionados a paginas y ventanas



1. load, unload

2. resize

3. error

4. contextmenu

1. El browser carga/sale de la pagina

2. La ventana del browser se redimensiona

3. Un error se activa cuando se carga un documento en la pagina

4. El usuario clic-botón-derecho para crear un menú contextual pop-up



Eventos de texto

1. Keydown
 2. keyup
 3. keypress
 4. focus
 5. blur
 6. select
- El usuario presiona una tecla cuando el focus esta sobre el elemento
 - El usuario libera una tecla cuando el focus esta sobre el elemento
 - el usuario presiona y libera una tecla cuando el elemento esta con focus
 - El focus va sobre el teclado
 - El focus se va del teclado
 - Este elemento de texto es seleccionado o de-seleccionado



Ejemplo texto

- keyCode = recupera el caractere Ascii que ha sido presionado
- altKey, ctrlKey, shiftKey = teclas especiales

```
document.getElementById("textbox").onkeydown = textKeyDown;
```

```
...
```

```
function textKeyDown(event) {  
    var key = String.fromCharCode(event.keyCode);  
    If (key == 's' && event.altKey) {  
        alert("Save the document!");  
        this.value = this.value.split("").join("-");  
    }  
}
```



Códigos teclado útiles

tecla	event keyCode
Backspace	8
Tab	9
Enter	13
Escape	27
Page Up, Page Down, End, Home	33, 34, 35, 36
Left, Up, Right, Down	37, 38, 39, 40
Insert, Delete	45, 46
Windows/Command	91
F1 - F12	112 - 123



Eventos de un Form

1. submit
 2. reset
 3. change
1. El form ha sido enviado
 2. El form ha sido recargado
 3. El texto o el estado de un form ha cambiado



Parar un evento

- Para parar un envío de form u otro event se puede usar **event.preventDefault();**

```
<form id="exampleform" action="http://foo.com/foo.php">...</form>
```

```
window.onload = function() {  
    var form = document.getElementById("exampleform");  
    form.onsubmit = checkData;  
};
```

```
function checkData(event) {  
    if (document.getElementById("state").length != 2) {  
        alert("Error, invalid city/state."); // show error message  
        event.preventDefault();  
        return false;           // stop form submission  
    }  
}
```

Asociar varios listeners al mismo evento



- AddEventListener asocia mas que un listener al mismo evento
- Cuando se clicha dos veces el primero listener remplaza el primero
 - ATENCION se usa “click” y no “onclick”

```
var button = document.getElementById("mybutton");  
button.addEventListener("click", func1);  
        // button.onclick = func1;  
button.addEventListener("click", func2);  
        // button.onclick = func2;
```


Timer





Timer

- setTimeout (*función*, *RetrasoMS*); se ocupa de llamar una función definida después de un retraso definido
- setInterval (*función*, *RetrasoMS*); se ocupa de llamar una función repetitivamente con cada *delayMS* milisegundos
- clearTimeout (*timerID*) / clearInterval (*timerID*); para el timer indicado para que no llame mas esa función



Timer (ejemplo)

```
<button onclick="delayMsg();">Click me!</button>  
<span id="output"></span>  
<script>
```

```
function delayMsg() {  
    setTimeout(boom, 5000);  
    document.getElementById("output").innerHTML = "espera...";  
}
```

```
function boom() { // llamada cuando el timer se apaga  
    document.getElementById("output").innerHTML = "BOOOOM!";  
}  
</script>
```

Timer (ejemplo con parámetros)



```
function delayedMultiply() {  
    // los valores 6 y 7 se pasan a la funcion  
    // cuando el timer se apaga  
    setTimeout(multiply, 2000, 6, 7);  
}  
  
function multiply(a, b) {  
    alert(a * b);  
}
```



Errores comunes

NO `setTimeout(boom(), 2000);`

SI `setTimeout(boom, 2000);`

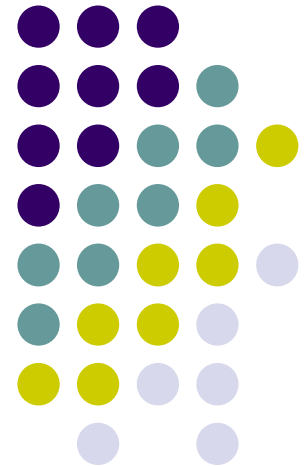
- La función se llama sin paréntesis!

NO `setTimeout(multiply(num1 * num2), 2000);`

SI `setTimeout(multiply, 2000, num1, num2);`

- Llama la función inmediatamente, sin esperar 2000ms!

Programación no obstructiva





Programación no obstructiva

- Motivación?
 - Evitar bloqueos/retrasos en cargar la pagina relacionados al evento
- Permite la separación del código en 3 categorías:
 - Contenido (HTML) – Vista
 - Presentación (CSS) – Apariencia
 - Comportamiento (JS) – Como responde al evento



Handler obstructivo (MAL)

// HTML

```
<button onclick="okayClick();">OK</button>
```

// JS

```
function okayClick() {  
    alert("Boom");  
}
```

- Programación incorrecta, el JS depende del HTML que se encuentra en el cuerpo (body) de la pagina. Si hay un problema en el JS la pagina interrumpe la visualización

Adjuntar el handler del evento al JS



#HTML

```
<button id="ok">OK</button>
```

#JS

```
var okButton = document.getElementById("ok");  
okButton.onclick = okayClick;
```

- En este caso el evento es independiente de la ejecución del body del HTML
- **IMPORTANTE** no se ponen paréntesis después del nombre de función!



Ejemplo (MAL): obstructivo

```
<html> // HTML
<head>
  <script src="myfile.js" type="text/javascript"></script>
</head>
<body>
  <div><button id="ok">OK</button></div>
```

myfile.js

```
var ok = document.getElementById("ok");
ok.onclick = okayClick; // error: null
```

- En este caso el **fichero JS** se ejecuta al momento en el cual se ejecuta script (antes del body).
- **IMPORTANTE** Ninguno de los objetos del DOM ha sido creado aun.
- Habrá un error porque el botón aun no existe



Evento window.onload

- Existe un evento global llamado window.onload que se ejecuta cuando el body se ha cargado completamente

#HTML

```
<button id="ok">OK</button>           <!-- (1) -->
```

#JS

```
window.onload = pageLoad;               // (2)
```

```
function pageLoad() {
```

```
    var ok = document.getElementById("ok"); // (3)
```

```
    ok.onclick = okayClick;
```

```
// llama okayClick solo cuando el boton OK has sido presionado
```

```
}
```

```
function okayClick() {
```

```
    alert("Boom");                       // (4)
```

```
}
```

Errores comunes en código no obstructivo



- Los eventos son lowercase

MAL = window.~~onLoad~~ = pageLoad;

BIEN = window.onload = pageLoad;

- No se usan las paréntesis cuando se llama el handler

MAL = ok.onclick = okayClick~~(~~;

BIEN = ok.onclick = okayClick;

- No se puede llamar directamente una función (como alert), se tiene que integrarla en otra función

MAL = ok.onclick = ~~alert("BOOM")~~;

BIEN = ok.onclick = okayClick;

function okayClick() { alert("BOOM");

Variables globales (peligroso)



```
var count = 0;
function incr(n) {
  count += n;
}
function reset() {
  count = 0;
}
```

```
incr(4);
incr(2);
console.log(count);
```

- El uso de variables globales es peligroso porque otro código JS podría recuperar su valor o modificarlas



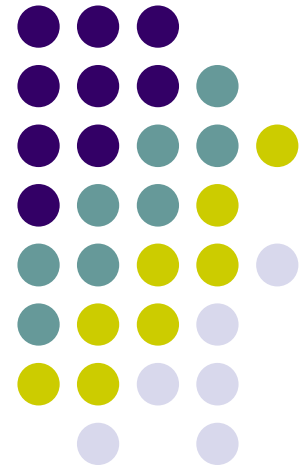
Variables globales (seguro)

```
function everything() {  
  var count = 0;  
  function incr(n) {  
    count += n;  
  }  
  function reset() {  
    count = 0;  
  }  
  
  incr(4);  
  incr(2);  
  console.log(count);  
}
```

```
everything();
```

- Para solucionar el problema se puede crear una función local que se ejecuta y que permite ocultar las variables locales

Librerias JS





Librerías JS

- La programación avanzada en Java Script puede ser difícil y requerir mucho tiempo.
- Para simplificar la programación, se han desarrollado librerías **JavaScript o framework**
- Las mas populares son :
 - jQuery
 - Prototype (script.aculo.us)

Estos frameworks contienen funciones para actuar en tareas de tipo JavaScript como animaciones, manipulaciones de DOM, y uso de Ajax.



Prototype

- **Prototype** es una librería de JS que proporciona una simple API (Application Programming Interface) para ejecutar tareas web.
 - Por ejemplo permite manipular fácilmente el DOM del HTML
 - Prototype mejora JavaScript porque proporciona clases y herencias.
 - Simplifica el uso de Ajax (asynchronous JavaScript and XML)
- En general se asocia con **script.aculo.us** que es un open-source framework para efectos visuales y gestión de respuestas de interfaces.



<http://prototypejs.org/learn/>

API Reference

The documentation for the latest stable version of Prototype will always be located at <http://api.prototypejs.org>.

Tutorials

This area contains **narrative documentation** you can use to discover Prototype.

- [Defining classes and inheritance](#)

Learn how to define classes and subclasses in Prototype and how to make supercalls.

- [How Prototype extends the DOM](#)

Learn how Prototype adds custom methods to DOM element nodes — and how you can define your own custom methods.

- [Introduction to Ajax](#)

Learn how Prototype simplifies the most common kinds of Ajax requests.

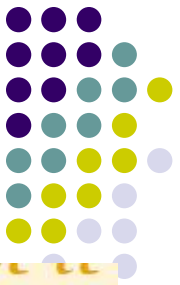
- [Introduction to JSON](#)

Learn about Prototype's support for JSON encoding and decoding.

- [Event delegation](#)

Learn about Prototype's support for event delegation: an advanced technique for event-driven programming.

<http://script.aculo.us/>



web.reload!

script.aculo.us provides you with **easy-to-use, cross-browser** user interface **JavaScript libraries** to make your web sites and web applications fly.

What's inside? animation framework, drag and drop, Ajax controls, DOM utilities, and unit testing.

It's an add-on to the fantastic Prototype framework.

who uses it?

Freckle time tracking · NASA · Apple
IKEA · Basecamp · Gucci · Shopify
Charm Customer Support · Backpack
Ruby on Rails

And many others!

Be sure to check out these demos, too:
puzzle game
list morphing

curr
script.ac
Decen

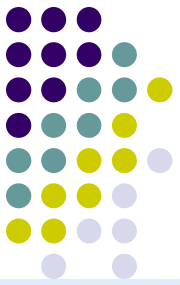
Dow
Gett


jQuery



- **jQuery** es el framework mas popular en Internet.
- Usa selectores CSS para acceder y manipular elementos HTML (objetos DOM) en una pagina web.
- Ejemplos de compañías que usan jQuery son:
 - Google
 - Microsoft
 - IBM
 - Netflix

jQuery






write less, do more.

PluginsContributeEventsSupportjQuery Foundation


DownloadAPI DocumentationBlogPluginsBrowser Support

Search




Lightweight Footprint

Only 32kB minified and gzipped.
Can also be included as an AMD module




CSS3 Compliant

Supports CSS3 selectors to find elements as well as in style property manipulation



Cross-Browser

IE, Firefox, Safari, Opera, Chrome, and more



Download jQuery

v1.11.0 or v2.1.0

[View Source on GitHub →](#)
[How jQuery Works →](#)

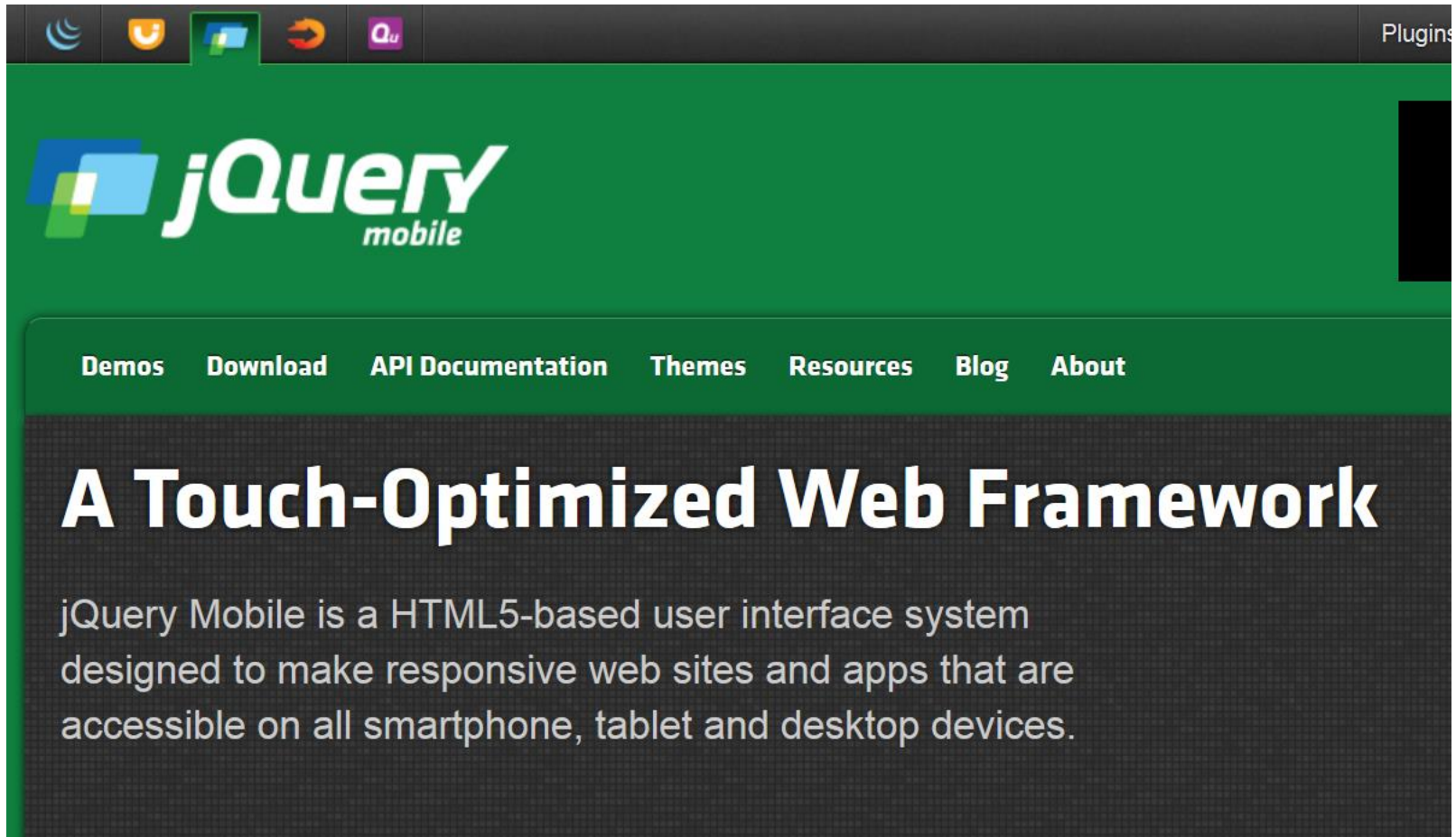
What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

Resources

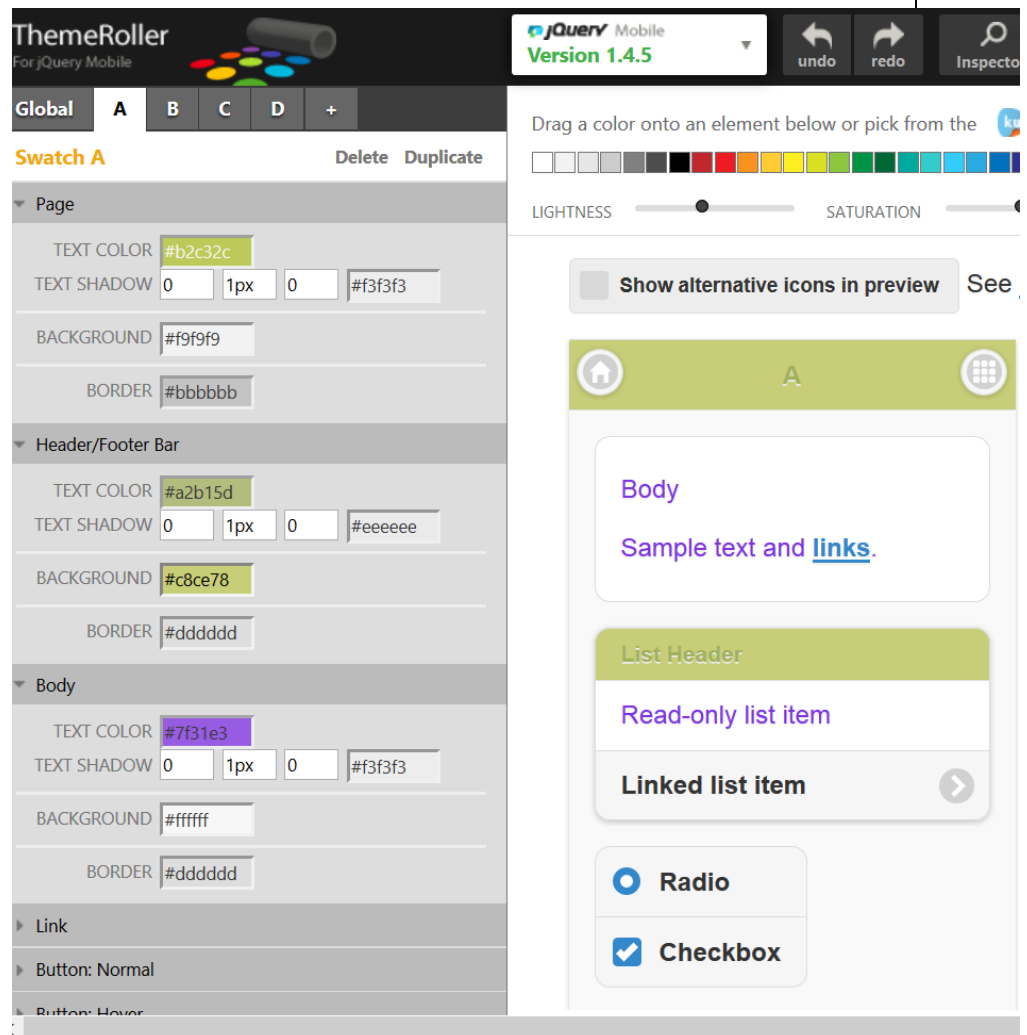
- [jQuery Core API Documentation](#)
- [jQuery Learning Center](#)
- [jQuery Blog](#)
- [Contribute to jQuery](#)

jQuery Mobile



jQuery Mobile

- Theme roller





Usar las librerías

- Las librerías de JavaScript son fáciles de integrar. No es necesario instalarlas en el ordenador, o configurarlas.
 - Se puede copiar en la directory de trabajo (si no se dispone de internet)
 - O se puede referenciar la librería desde la pagina web.



Testear Prototype / jQuery

- Para testear una librería JavaScript, se tiene que incluir en la pagina web un URL.
- Usar el tag <script> con el atributo src definido como la URL de la librería:

```
<!DOCTYPE html>
<html>
  <head> <script>
    src="http://ajax.googleapis.com/ajax/libs/prototype/1.7.1.0/prototype.js">
  </script> </head>
  <body> </body>
</html>
```

- Otros enlaces librerías JavaScript web.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js">
<script src="https://ajax.googleapis.com/ajax/libs/scriptaculous/1.9.0/scriptaculous.js"></script>
<link rel="stylesheet"
  href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.4/themes/smoothness/jquery-ui.css">
```

Librerías online

<https://developers.google.com/speed/libraries/#prototype>