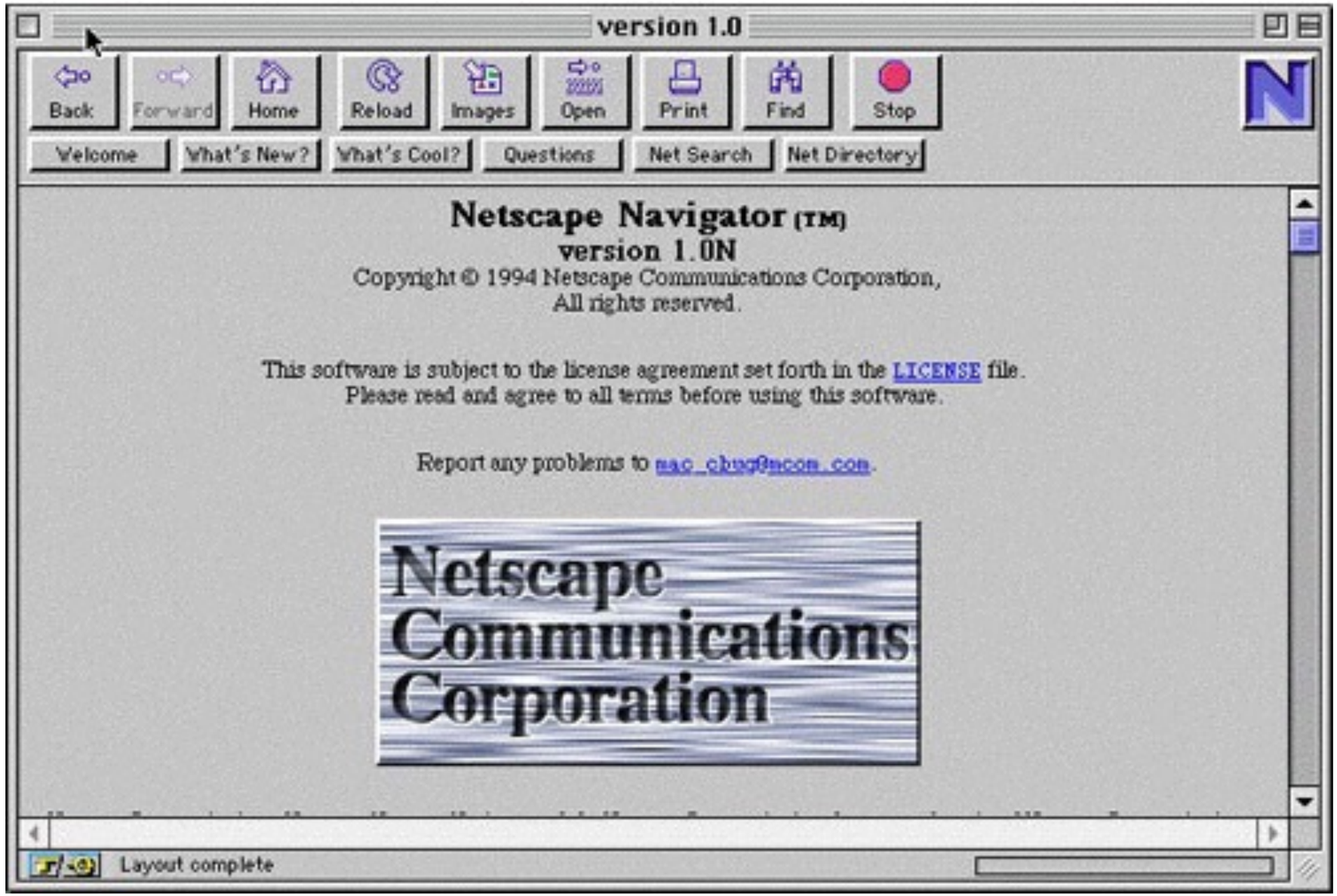
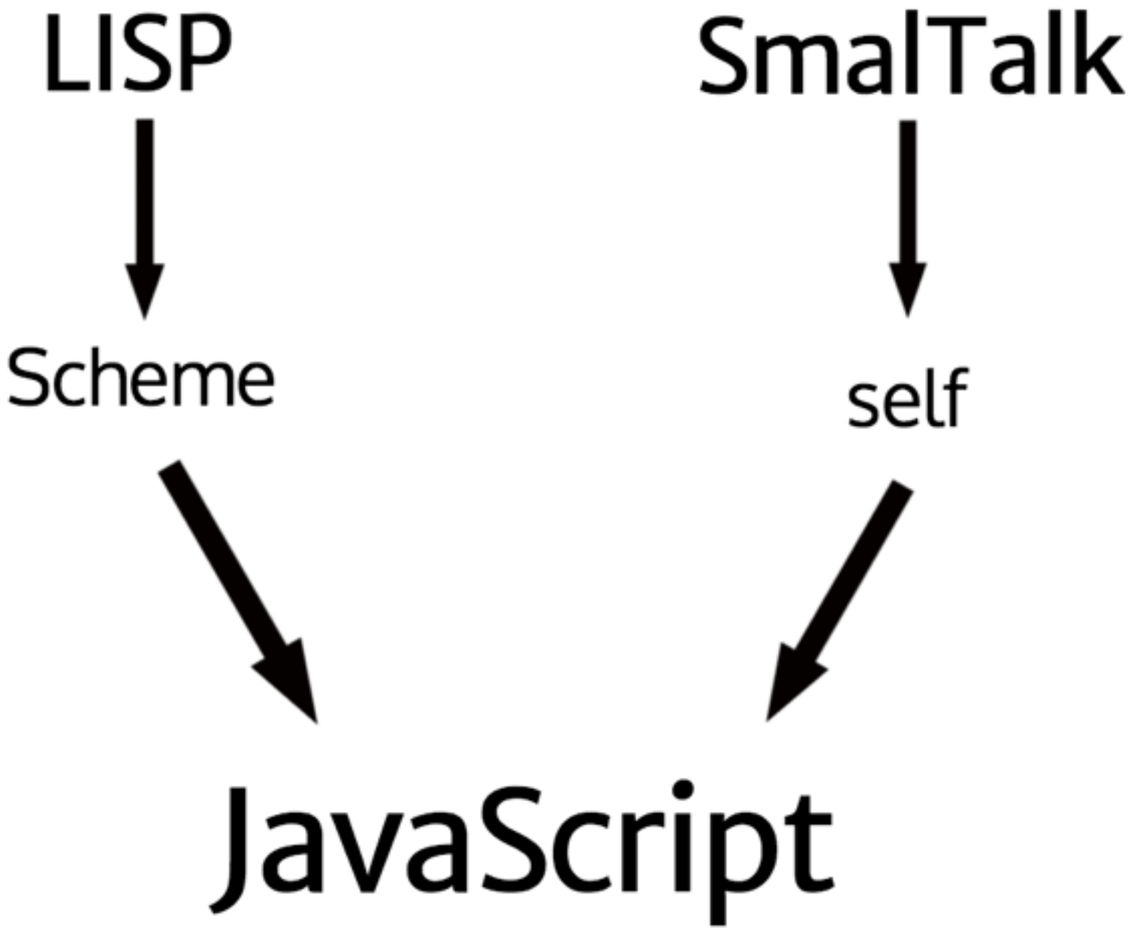


Introducción

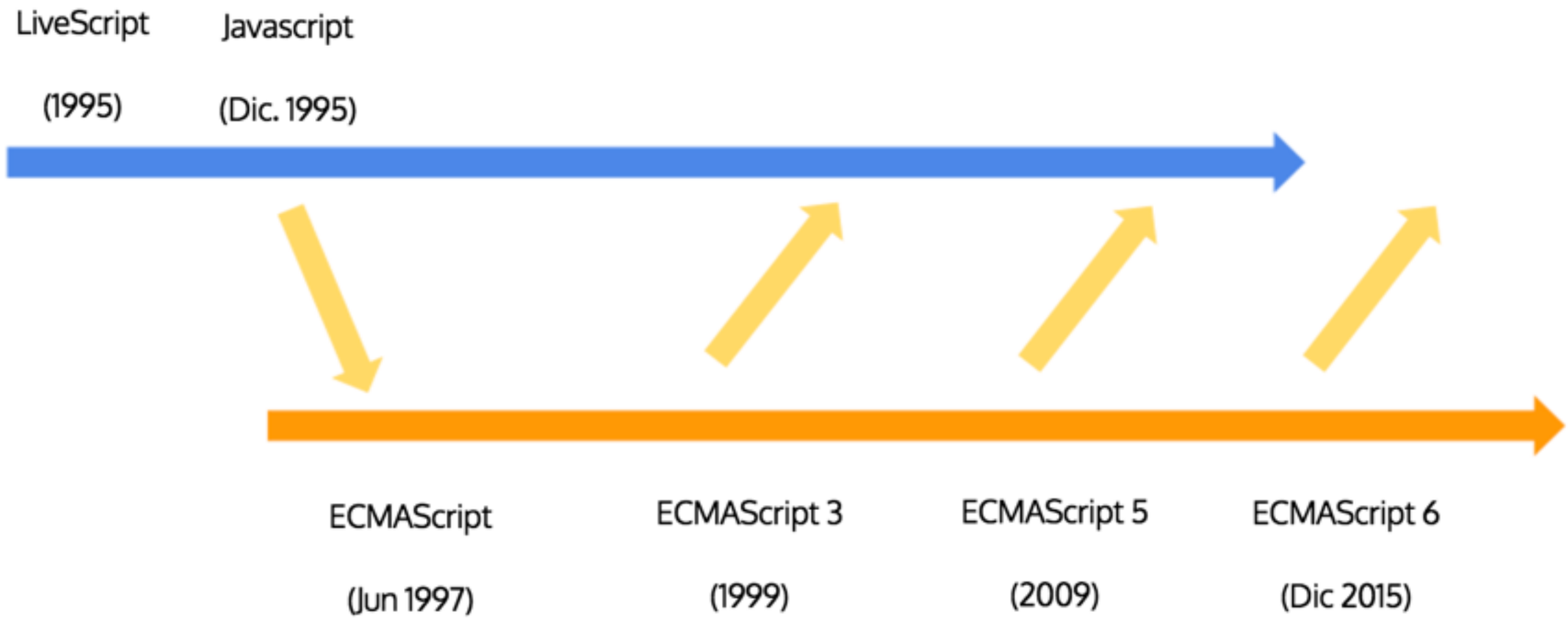


Introducción

Características:

- Tipado dinámico (habitual en los lenguajes de script)
- Funcional
- Orientado a objetos (basado en prototipos)

Introducción



http://www.ecma-international.org/

Introducción

Con JavaScript se puede modificar la página y ejecutar código cuando se interactúa con ella (a través del modelo de objetos del documento DOM).

Se hacen peticiones al servidor web en segundo plano y se actualiza el contenido de la web (AJAX).

También es posible desarrollar en el lado del servidor o para automatizar tareas.

¿Por qué JavaScript?

Pros:

- El lenguaje del navegador
- Porque en parte es funcional
- Es fácil de “echarlo a andar”

Contras:

- Obligatorio para el desarrollo web
- Lenguaje aparentemente incompleto
- Implementación cuestionable (this, prototype, coercion, ...)
- Inconsistente o impredecible

Referenciar JavaScript

En el propio HTML

```
<!DOCTYPE html>
<html>
<head></head>
  <body>
    <script>
      alert('Hola!');
    </script>
  </body>
</html>
```

Referenciar JavaScript

Desde un fiechero externo

```
<!DOCTYPE html>
<html>
  <head></head>
    <body>
      <script src="main.js"></script>
    </body>
</html>
```

main.js

```
alert('Hello world!');
```

Aspectos del Lenguaje

Trazas de log

```
console.log('traza de log');
console.debug('traza de debug');
console.info('traza de info');
console.warn('traza de aviso');
console.error('traza de error');
```

Alertas

```
alert('Hello world!');
```

Aspectos del Lenguaje

Comentarios en el código

```
/*
 * Comentar un bloque
 * de líneas
 */

// Comentar una línea
```

Tipos de Datos: Variables

```
var empty = 10;
var cadena = 'Esto es una cadena';
empty = false;
console.log(notDefined); // undefined
```

```
functionNameLikeThis;
variableNamesLikeThis;
_internalVariablesLikeThis;
SYMBOLIC_CONSTANTS_LIKE_THIS;
```

Expresiones

```
3 + 1
'hola' < 'adios'
64 >= 'a'
1 == '1'
1 === '1'
1.0 !== 1
```

Condicionales

```
if ( condition ) {
   // statements
}
```

```
if ( condition ) {
   // statements
} else {
   // statements
}
```

```
var aux = condition? true : false;
```

Bucles

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

```
var elementos = ['uno', 'dos', 'tres'];
elementos.forEach(function(elemento) {
  console.log(elemento);
});
```

Bucles

```
while ( condition ) {
   // statements
}
```

```
do {
   // statements
} while ( condition )
```

Tipos de datos

Tipos de datos

Números

Strings

Booleanos

Null

Undefined

Objetos

Tipos de Datos: Números

Tipo único de datos numérico

```
10.0 === 10
```

Precisión de 64 bits

NaN

```
parseInt('not a number');
isNaN()
```

Infinity

```
infinity = 1,79769313486231570e+308
```

Tipos de Datos: Cadenas

Delimitados por **"** o **'**

'****' para escapar carácteres

'**\u**' para carácteres Unicode

```
'A' === '\u0041'
```

No hay typo "**char**"

Tipos de Datos: Cadenas

string.replace(search, replace)

```
var str="My name is John!";
var n=str.replace("John","Peter");
// "My name is Peter!"
```

string.match(regexp)

```
var str = 'The rain in SPAIN stays mainly in the
plain';
var n = str.match(/ain/g);
// [ 'ain', 'ain', 'ain' ]
```

string.split()

```
var str = "How are you doing today?";
var n = str.split(" ");
// ['How','are','you','doing','today?']
```

Tipos de Datos: Cadenas

string.slice(start, end)

```
var text = 'and it he says "Any damn fool
could'
var a = text.slice(18)
// '"Any damn fool could'
text.slice(0,3)
// 'and'
```

string.toLowerCase

string.toUpperCase()

Tipos de Datos: Cadenas

```
parseInt('10', 10)
```

```
parseInt('10 cows', 10)
```

```
parseInt('cows 10', 10)
```

```
parseInt('09/2013', 10)
```

Tipos de Datos: Coercion

```
console.log(1 +  "2" + "2");
console.log(1 +  +"2" + "2");
console.log(1 +  -"1" + "2");
console.log(+"1" +  "1" + "2");
console.log( "A" - "B" + "2");
console.log( "A" - "B" + 2);
```

Tipos de Datos: Booleanos

true

false

Tipos de Datos: Booleanos

true

Todo lo que no es "falsy"

false

''

0

null

undefined

NaN

Tipos de Datos: Evaluación perezosa

```
console.log("0 || 1 = "+(0 || 1));
console.log("1 || 2 = "+(1 || 2));
console.log("0 && 1 = "+(0 && 1));
console.log("1 && 2 = "+(1 && 2));
```

Tipos de Datos: Undefined

DUNNO

Tipos de Datos: Declarar funciones

```
function suma(op1, op2) {
   return op1 + op2;
}
suma(1, 2);
```

```
var suma = function(op1, op2) {
  return op1, + opt2;
}
suma(1, 2);
```

```
var obj = {
  nombre: "Pepito",
  saludo: function () {
    return "Hola, Mundo!";
  }
};
obj.saludo();
```

```
function what(op1, opt2) {
  return function() {
    return opt1 + opt2;
  }
}
What(1, 2)();
```

```
$("#elemento").click(function() {
  alert('Woho!');
});
```

KEEP CODING

Tipos de Datos: Invocar funciones

De forma directa

```
(function() {
    alert("Hey!");
})();

var foo = function() {
    alert('Hey!');
}
foo();
```

Enviando un mensaje a un objeto

```
objeto.metodo();
```

Como constructor

```
new MiConstructor();
```

Indirectamente (call, apply)

```
fn.call({}, "param");
```

Tipos de Datos: Arrays

```
var empty = []; var numbers = ['one','two','three','four','five'];
```

```
numbers[1]
numbers.length
```

```
var etc = ['string', 0.5, true, false, null, undefined,
           ['nested', 'array'], {object: true}, NaN,
           Infinity];

etc.length?
```

Tipos de Datos: Arrays

```
avar fruits = ["Banana", "Orange", "Apple", "Mango“];
```

array.push

```
fruits.pop();
Banana,Orange,Apple
```

array.pop

```
fruits.push("Kiwi")
Banana,Orange,Apple,Mango,Kiwi
```

Tipos de Datos: Arrays

```
var fruits = ['Banana', 'Orange', 'Lemon', 'Apple', 'Mango'];
```

array.slice(start, end)

```
var citrus = fruits.slice(1,3);
Orange,Lemon
```

array.splice(start, howmany, item1, ..., itemN)

```
fruits.splice(2,1,'Peach','Kiwi');
// output: Lemon
'Banana','Orange','Peach','Kiwi','Apple','Mango' // fruits
```

Tipos de Datos: Arrays

```
array.concat( array1, ..., arrayN )
array.join( separator )
array.reverse()
array.shift()
array.short( function(value1, value2) )
```

Tipos de Datos: Objetos prototipos

```
function Player(firstName, lastName, level) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.level = level;
        this.play = function() { ... };
}
var air = new Player("Michael", "Jordan", "god");
```

Tipos de Datos: Objetos prototipos

```
function Player(firstName, lastName, level) {
   this.firstName = firstName;
   this.lastName = lastName;
   this.level = level;
}
var air = new Player("Michael", "Jordan", "god");

Player.prototype.team = "Chicago Bulls"

air.team; // "Chicago Bulls"
```

Tipos de Datos: Objetos Literales

Conjunto de propiedades propias + heredadas

```
var obj = {
  hola: true,
  adios: true
};
```

```
var obj = {};
obj.nuevaPropiedad = 1;
```

```
var user = {
  name: 'John',
  'e-mail': 'john@mail.com'
}
user.name;
user['e-mail'];
```

Objetos: Acceso a propiedades

```
var obj = {
    nombre: 'Pepito',
    saludo: function () {
      return 'Hola, ' + this.nombre + '!';
    }
};
obj.nombre;
obj.saludo;
obj[“saludo”]();
var fn = obj["saludo"];
fn();
```

Que devuelve obj.saludo()?

Que devuelve fn()?

Objetos: Contexto

```
var obj = {
    nombre: 'Pepito',
    saludo: function () {
      return 'Hola, ' + this.nombre + '!';
    }
};
obj.nombre;
obj.saludo;
obj[“saludo”]();
var fn = obj["saludo"];
fn();
```

Su significado es dinámico

Se decide en tiempo de ejecución

También conocido como "contexto"

Si no hay contexto, apunta al objeto global

Alcance de una función

```
(function(){
  var a = b = 3;
})();

console.log("a defined? " + (typeof a !== 'undefined'));
console.log("b defined? " + (typeof b !== 'undefined'));
```

Las variables definidas en una función son accesibles en las funciones internas.

Las variables definidas en una función no son accesibles desde el exterior.

Hoisting

```
function test() {
   console.log(a);
   console.log(foo());

   var a = 1;
   function foo() {
      return 2;
   }
}

test();
```

¿Qué devuelve la función?

Hoisting

```
function test() {
   console.log(a);
   console.log(foo());

   var a = 1;
   function foo() {
      return 2;
   }
}

test();
```

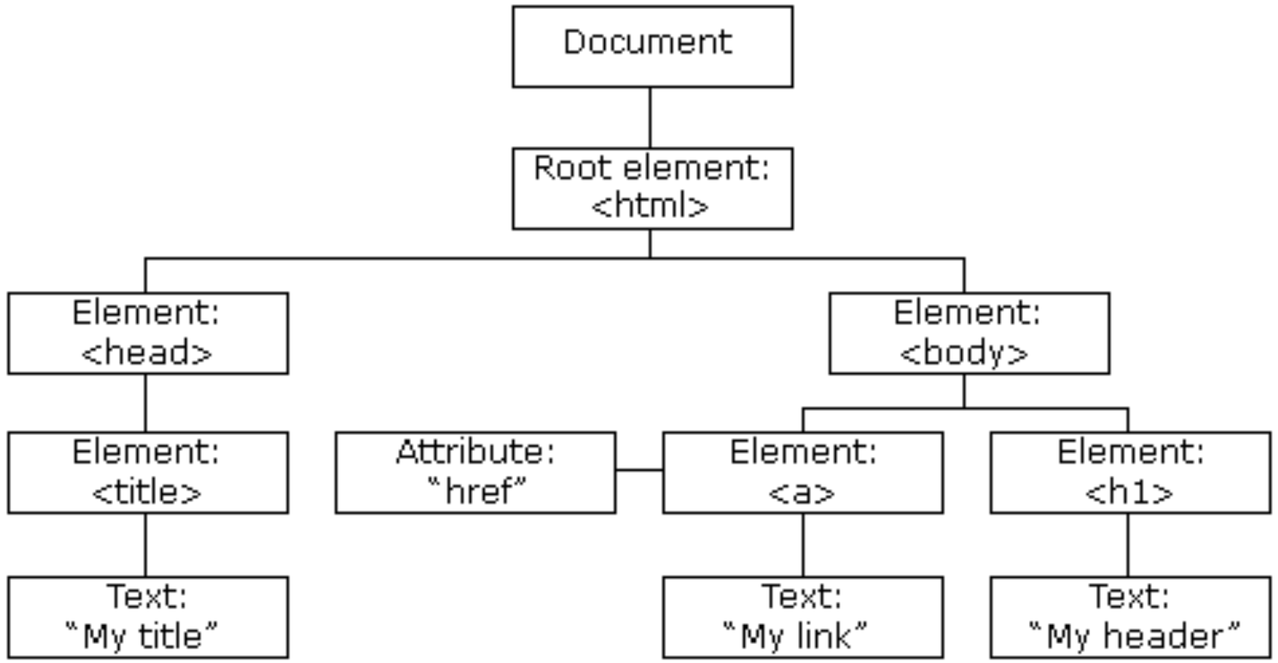
```
function test() {
   var a = 1;
   function foo() {
      return 2;
   }

   console.log(a);
   console.log(foo());
}

test();
```

Javascript mueve la declaración de variables funciones al inicio del contexto.

==> **‘use strict’;** <==

Document Object Model (DOM)

Librería (API) para manipular el documento HTML cargado



DOM: Seleccionando elementos

```
// Encuentra elementos por ID
document.getElementById();

// Encuentra elementos por su tag
document.getElementsByTagName();

// Encuentra elementos por su clase CSS
document.getElementsByClassName();

// Permite utilizar selectores CSS
document.querySelectorAll();
```

DOM: Modificando elementos

```
// String con el contenido HTML de un elmeneto
element.innerHTML;

// Accede al atributo 'attribute' de un elmeneto
element.attribute;

// Crea un atributo a un elemento
element.setAttribute(name, value);
```

DOM: Añadiendo|Eliminando elementos

// Encuentra elementos por tag

document.createElement();

// Elimina un elemento

document.removeChild();

// Añade un elemento

document.appendChild();

// Cambia un elemento por otro

document.replaceChild();

// Escribe la cadena que le pases en ese punto del HTML

document.write();

DOM: Modificando estilos

```
element.style.color = “blue”;

element.style.backgroundImage = “none”;
```

Eventos

JavaScript es un lenguaje asíncrono y orientado a eventos.

La forma de sincronizarse es a través de callbacks o handlers a los eventos.

Al ejecutarse en un único hilo no hay problemas de concurrencia.

Esta es la base para el gran redimiento que ofrecen tecnologías como node.js, que se basa en la asincronía junto con operaciones I/O no bloqueantes.

Eventos

```
<button id='magicButton'>The time is?</button>

<script>

    function doSomething() {...}

    document.getElementById('magicButton')
        .addEventListener("click", doSomething);

    document.getElementById('magicButton')
        .removeEventListener("click", doSomething);

</script>
```

Eventos: Comportamiento por defecto

Hay componentes de HTML a los que el navegador les asigna un comportamiento por defecto.

Por ejemplo, los <a> nos suelen llevar a donde indica su atributo href, pero podemos decirle al navegador que no ejecute ese comportamiento.

```
document.getElementById("a").addEventListener("click", function(event) {
    // do not open the link
    event.preventDefault();
});
```

Eventos: Burbujeo de eventos

Los eventos se disparan en el nivel mas bajo del DOM y se propaga hacia arriba, a los elementos contenedores.

```
<div>
  <p>...</p>
</div>
```

```
document.querySelector('div').addEventListener('click', function() {
    console.log(this);
});

document.querySelector('div p').addEventListener('click', function() {
    console.log(this);
});
```

```
$> <p>...</p>
$> <div>...</div>
```

Eventos: Captura de eventos

Los eventos se disparan en el nivel mas alto posible del DOM y se propaga hacia abajo, a los elementos que contiene.

```
<div>
  <p>...</p>
</div>
```

```
document.querySelector('div').addEventListener('click', function() {
    console.log(this);
}, true);

document.querySelector('div p').addEventListener('click', function() {
    console.log(this);
}, true);
```

```
$> <div>...</div>
$> <p>...</p>
```

Eventos: Parar la propagación

Podemos evitar que el evento se propague.

```
document.getElementById("p").addEventListener("click", function(event) {
    event.stopPropagation();
});
```

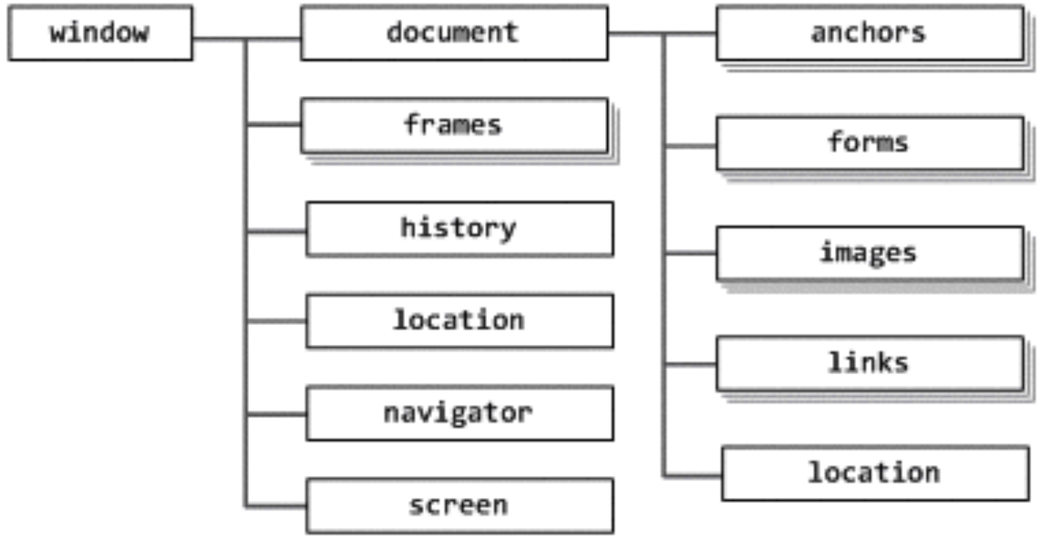
Eventos: Acertijo

```
for (var i = 0; i < 5; i++) {
    var btn = document.createElement('button');
    btn.appendChild(document.createTextNode('Button ' + i));
    btn.addEventListener('click', function() {
        console.log(i);
    });

    document.body.appendChild(btn);
}
```

¿Qué sucede al pulsar los botones? ¿Cómo mejorarlo?

Browser Object Model (BOM)

API no oficial disponible en la mayoría de los navegadores modernos que expone métodos para interactuar con el navegador.



Controlando el tiempo

JavaScript nos da dos funciones para ejecutar código en referencia al tiempo:

setTimeout()

setInterval()

Controlando el tiempo: Timeout

setTimeout: pone una cuenta atrás y ejecuta la función que indicamos

```
var timeoutID = setTimeout(function () {
    alert("Hello");
}, 3000);

// remove timer
clearTimeout( timeoutID );
```

Controlando el tiempo: Interval

setInterval: ejecuta una función cada X tiempo

```
var i = 0;
var intervalId = setInterval(function () {
    alert("Hello");
    i++;
    if (i > 3)
        // remove interval
        clearInterval( intervalId );
}, 3000);
```

Controlando el tiempo: Acertijo

```
(function() {
    console.log(1);
    setTimeout(function(){console.log(2)}, 1000);
    setTimeout(function(){console.log(3)}, 0);
    console.log(4);
})();

for (var i = 0; i < 5; i++) {
  setTimeout(function() { console.log(i); }, i * 1000 );
}
```

¿Qué sucede al ejecutarlo?

Librerías Típicas

jQuery: es un recubrimiento de la API DOM que aporta facilidad de uso, potencia y compatibilidad entre navegadores. Se usa para gestionar el interfaz (la página) y para peticiones ajax.

Librerías Típicas: JQuery

La librería se puede descargar de su página web para desarrollo local (http://jquery.com/)

Es recomendable usar una ruta en una red de distribución de contenido (CDN)

Si la librería está cacheada en el navegador del cliente no se descargará de nuevo (reduciendo el tiempo de carga)

http://code.jquery.com

https://developers.google.com/speed/libraries/devguide#jquery

JQuery

Acceder al DOM

```
var element =$('body');
$('.content').find('article p');
$('td:contains(Henry)').next();
$('td:contains(Henry)').parent().find('td:eq(1)');
```

Escribir en el documento

```
$('p').html('<span>hola!</span>');
$('p').append('<span>hola!</span>');
$('p').prepend('<span>hola!</span>');
```

Eliminar elemento

```
$('article .content span').remove();
```

JQuery

Modificar/acceder a los estilos

```
$('.content').css('width', '30px');
$('.content').css('width'); // '30px'
```

Modificar clases

```
$('p').addClass('claim claim-epic');
$('p').removeClass('claim-epic');
$('p').hassClass('claim-epic'); //false
```

JQuery

Modificar/acceder a los atributos

```
$('button').attr('some-attr', 'value');
$('button').attr('some-attr'); // 'value'
$('button').removeAttr('some-attr');
```

Obtener/establecer valores de entrada (String!)

```
$('input[type="text"]').val('hello!');
$('input[name="radio"]').val('3');
$('input[name="radio"]').val(); //'3'
```

JQuery

Obtener tamaño de elemento

```
$('.content').height();   // 30
$('.content').width();    // 30
```

Obtener posición de elemento

```
$('.content').offset({top:20,left:10});
var offset = $('.content').offset();
offset.top; //20
offset.left; //10
```

JQuery

Mostrar/esconder elemento

```
$('.content').show();
$('.content').hide();
$('.content').slideDown();
$('.content').slideUp();
```

JQuery

Eventos de usuario

```
$('.button-submit').click(function(event) {
  event.preventDefault();
  alert('clicked!');
});
```

blur, change, click, focus, focusin, focusout, hover, keydown, keypress, keyup, mousedown, mouseenter, mouseleave, mousemove, mouseout, mouseover, mouseup, ready, resize, scroll, submit, ...

JQuery

Eventos de usuario

Evita que un formulario se envíe o que se navegue a un enlace

```
$('.button-submit').click(function(event) {
  event.preventDefault();
  alert('clicked!');
});
```

blur, change, click, focus, focusin, focusout, hover, keydown, keypress, keyup, mousedown, mouseenter, mouseleave, mousemove, mouseout, mouseover, mouseup, ready, resize, scroll, submit, ...

JQuery

Ejecutar al cargar el documento

```
$(document).ready(function() {
   // document fully loaded
});
```

http://api.jquery.com/category/events/

JQuery: AJAX

Asynchronous JavaScript And XML (AJAX)

- Técnica web para desarrollo de contenido dinámico
- Se realiza en segundo plano sin bloquear la interfaz del usuario (asíncrono)
- El cliente realiza una petición asíncrona para eventualmente, recibir una respuesta del servidor.
- En nuestros ejemplos accederemos a recursos de tipo JSON

http://api.jquery.com/jquery.ajax/

JQuery: AJAX

Obtener un JSON (GET)

example.json

```
{
  "name": "Sonia",
  "surnames": "Paz"
}
```

```
$.ajax({
  dataType: 'json',
  url: 'example.json'
}).done(function(jsonData) {
  alert('success', jsonData.name );
}).fail(function(error){
  console.error('error', error);
});
```

http://api.jquery.com/jquery.ajax/

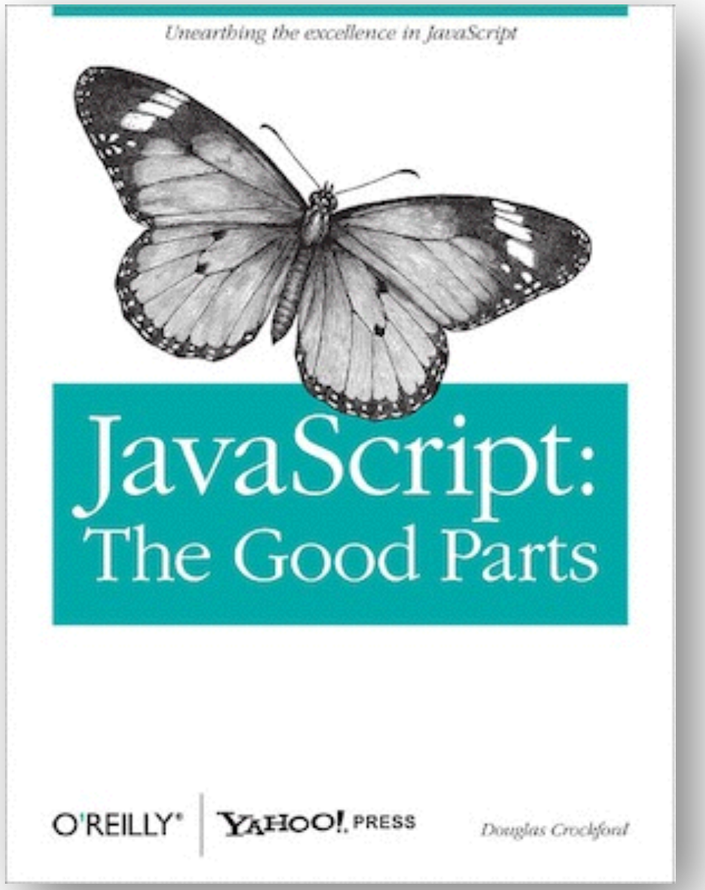
JQuery: AJAX

Enviar un JSON (POST/PUT)

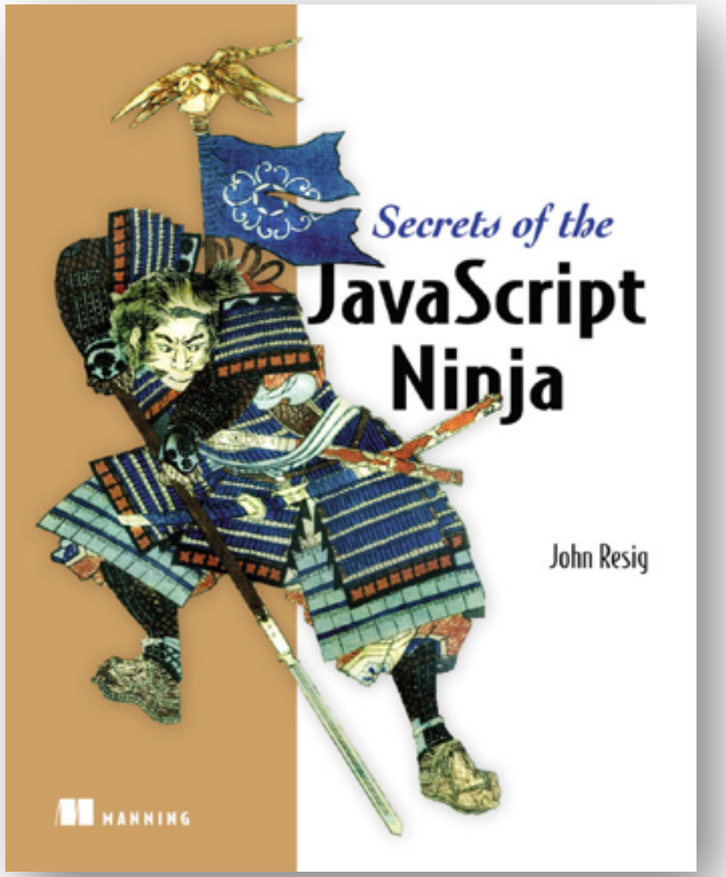
```
$.ajax({
  type: 'POST', // o 'PUT'
  contentType: 'application/json ; charset=utf-8',
  data: JSON.stringify(objectData),
  url: 'http://someplace.com/path'
}).done(function(jsonData) {
  alert('success', jsonData.name );
}).fail(function(error){
  console.error('error', error);
});
```

http://api.jquery.com/jquery.ajax/

Jquery: AJAX

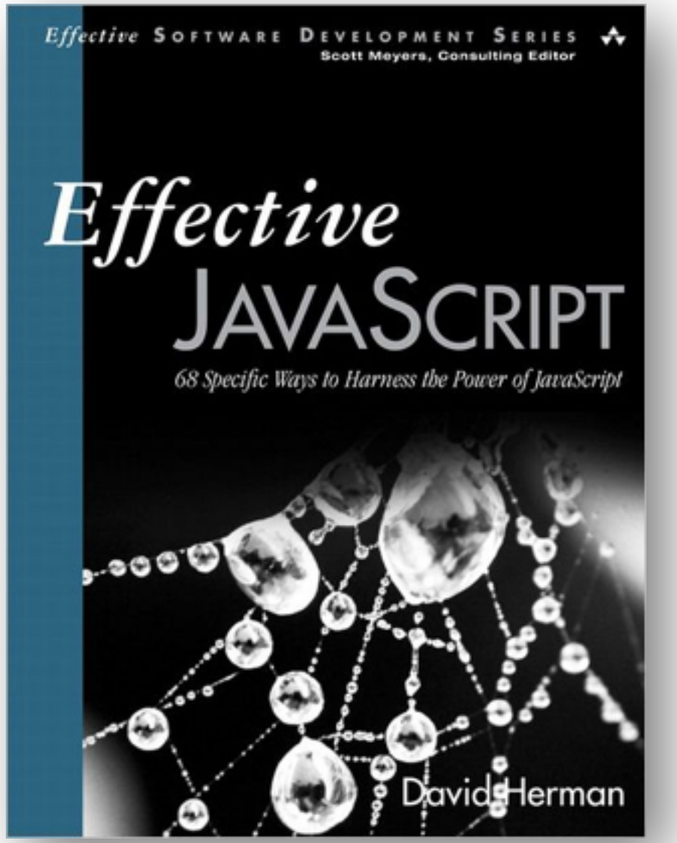


(2008)

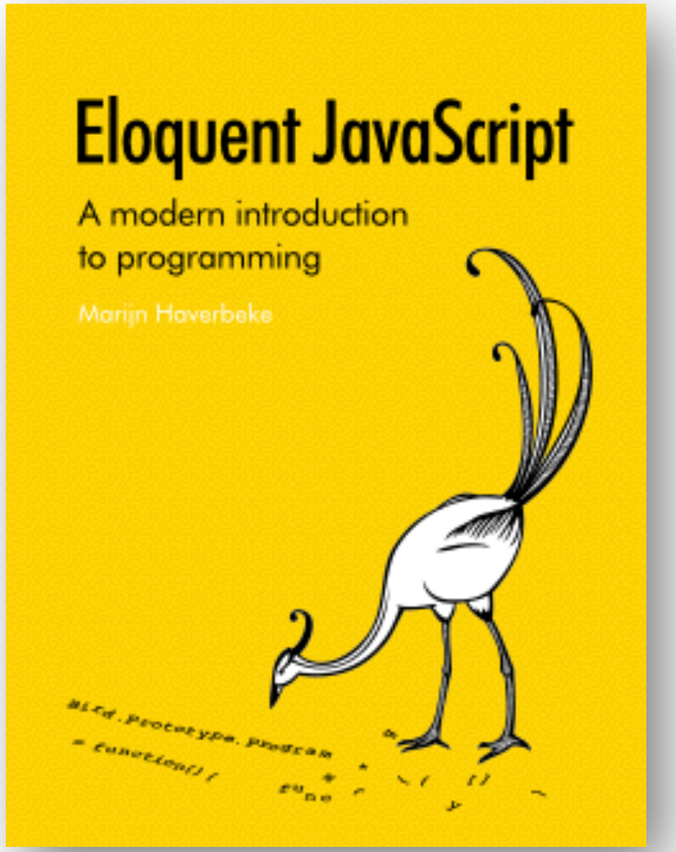


(2012)

Jquery: AJAX



(2012)



(2011) Descarga gratuita

GRACIAS

www.keepcoding.io