



Departamento de Lenguajes y  
Sistemas Informáticos



Universitat d'Alacant  
Universidad de Alicante

# Ajax Technology in Web Programming

**Sergio Luján Mora**

Ajax Technology in Web Programming

Basic syntax and features

# JAVASCRIPT

## Index

- Introduction
- Including JavaScript in web pages
- Language syntax
- Browsers' tools
- More information

## Introduction

- JavaScript
- Applications
- Security restrictions
- JavaScript and Java applets
- Versions
  - Netscape, Mozilla, and Firefox
  - Microsoft
  - ECMAScript
- What do I need?

## JavaScript

- Original name: Mocha → LiveScript (1995)
  - Created by Brendan Eich
  - Netscape 2.0B3 (December 1995)
  - JavaScript: agreement with Sun Microsystem
- Microsoft:
  - Internet Explorer 3 → JScript
- Most standard language for web browser programming

## JavaScript

- Suitable for C, C++, and Java programmers
- JavaScript ≠ Java (Sun Microsystems)
- Microsoft:
  - JScript
  - JScript.NET

## JavaScript

- JavaScript is used in:
  - Client: Internet Explorer, Netscape Navigator, Opera, Mozilla, etc.
  - Server: ASP, Netscape Enterprise Server
  - JavaScript's LiveConnect →Java
  - Java 6: javax.script
  - Adobe PDF
  - Adobe ActionScript based on JavaScript

## Applications

- During many years, JS was the only available technology for programming web browsers
- Nowadays, there exists many other alternative technologies (VBScript, Adobe Flash, etc.), but JS is the only standard technology for every web browser

## Applications

- JavaScript, DOM (*Document Object Model*) and BOM (*Browser Object Model*) allow to program web browsers
- Main applications in web pages:
  - Validates users' input in web forms:
    - Reduced workload in web server
    - Reduces delays when users' input is wrong
    - Provides more interactivity
  - Shows alerts and messages
  - Updates web pages (e.g., web forms)

## Applications

- Main applications in web pages:
  - Interacts with Java applets and other resources (ActiveX, Adobe Flash, etc.)
  - DHTML (*Dynamic HTML*): HTML + CSS + JavaScript → Dynamism and interactivity
  - AJAX (*Asynchronous JavaScript and XML*)

## Security restrictions

- It is not allowed to:
  - Access local resources (files, printer, etc.)
  - Connect to other servers (only the server where the script is stored)
- These restrictions are imposed, they are not technological restrictions:
  - A digitally signed script can skip these restrictions

## JavaScript and Java applets

JavaScript	Java applets
Interpreted	Compiled to bytecodes
Based on objects	Based on classes
Embedded code in web page	Applet referenced in web page
Dynamic types Weak types	Static types Strong types

## Netscape versions

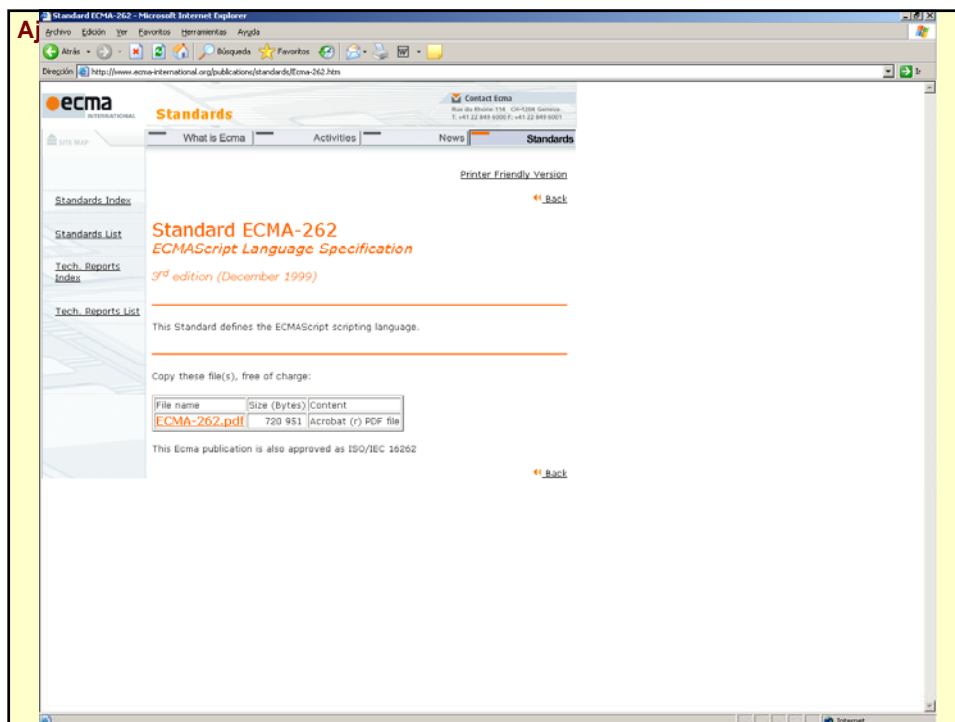
JavaScript version	Browser version
JavaScript 1.0	Navigator 2.0
JavaScript 1.1	Navigator 3.0
JavaScript 1.2	Navigator 4.0 – 4.05
JavaScript 1.3	Navigator 4.06 – 4.78
JavaScript 1.4	No browser
JavaScript 1.5	Navigator 6.x, Mozilla Application Suite, Firefox 1.0
JavaScript 1.6	Firefox 1.5
JavaScript 1.7	Firefox 2
<i>JavaScript 2.0</i>	<i>Under development</i>

## Microsoft versions

Host Application	1.0	2.0	3.0	4.0	5.0	5.1	5.5	5.6	.NET	8.0
Microsoft Internet Explorer 3.0	x									
Microsoft Internet Information Server 3.0		x								
Microsoft Internet Explorer 4.0			x							
Microsoft Internet Information Server 4.0			x							
Microsoft Internet Explorer 5.0					x					
Microsoft Internet Explorer 5.01						x				
Microsoft Windows 2000						x				
Microsoft Internet Explorer 5.5							x			
Microsoft Windows Millennium Edition							x			
Microsoft Internet Explorer 6.0								x		
Microsoft Windows XP								x		
Microsoft Windows Server 2003								x		
Microsoft .NET Framework 1.0									x	

## ECMAScript

- ECMA (*European Computer Manufactures Association*)
- ECMAScript → ECMA 262
  - Ed. 1: June 1997
  - Ed. 2: June 1998
  - Ed. 3: December 1999
  - <http://www.ecma-international.org/publications/standards/Ecma-262.htm>



## ECMAScript

- ECMAScript overview:

“ECMAScript is an object-oriented programming language for performing computations and manipulating computational objects within a host environment. ECMAScript as defined here is not intended to be computationally self-sufficient; indeed, there are no provisions in this specification for input of external data or output of computed results.”

## ECMAScript

- ECMAScript overview:

“Instead, it is expected that the computational environment of an ECMAScript program will provide not only the objects and other facilities described in this specification but also certain environment-specific *host* objects, whose description and behaviour are beyond the scope of this specification except to indicate that they may provide certain properties that can be accessed and certain functions that can be called from an ECMAScript program.”

## ECMAScript

- ECMAScript overview:

“A **scripting language** is a programming language that is used to manipulate, customise, and automate the facilities of an existing system. In such systems, useful functionality is already available through a user interface, and the scripting language is a mechanism for exposing that functionality to program control. In this way, the existing system is said to provide a host environment of objects and facilities, which completes the capabilities of the scripting language. A scripting language is intended for use by both professional and nonprofessional programmers. To accommodate non-professional programmers, some aspects of the language may be somewhat less strict.”

## ECMAScript

- ECMAScript overview:

“ECMAScript was originally designed to be a **Web scripting language**, providing a mechanism to enliven Web pages in browsers and to perform server computation as part of a Web-based client-server architecture. ECMAScript can provide core scripting capabilities for a variety of host environments, and therefore the core scripting language is specified in this document apart from any particular host environment.”

## ECMAScript

- ECMAScript overview:

“A **web browser** provides an ECMAScript host environment for client-side computation including, for instance, objects that represent windows, menus, pop-ups, dialog boxes, text areas, anchors, frames, history, cookies, and input/output. Further, the host environment provides a means to attach scripting code to events such as change of focus, page and image loading, unloading, error and abort, selection, form submission, and mouse actions. Scripting code appears within the HTML and the displayed page is a combination of user interface elements and fixed and computed text and images. The scripting code is reactive to user interaction and there is no need for a main program.”

## ECMAScript

- Submitted to ISO:
  - April 1998 ISO/IEC-16262
- JavaScript and JScript are specific implementations of ECMAScript

## What do I need?

- Text editor:
  - Syntax highlight
  - Syntax wrapping
- Browser

## Including JS in web pages

- Three ways:
  - Between `<script>` and `</script>` in head or body sections
  - Event attributes in HTML tags: `onclick`, `onblur`, `onchange`, ...
  - In an URL (pseudoprotocol):  
`<a href="javascript:">...</a>`
- Important: the code must be completely loaded before calling it

## Including JS in web pages

- `<script></script>`:
  - `charset`: the set of chars
  - `src`: URL of the code
    - Extension of the file: normally \*.js
  - `type`: MIME type identifies the programming language
  - `defer="defer"`: the script doesn't generate content (no `document.write`)
- Important: don't use the old attribute language

## Including JS in web pages

- Example:

```
<script type="text/javascript"
  src="http://someplace.com/progs/calc.js">
</script>
```
- How to define the default programming language in a web page:

```
<meta http-equiv="Content-Script-Type"
  content="text/javascript" />
```

## Including JS in web pages

- For old browsers don't understand `<script>` tag:

```
<script type="text/javascript">
<!-- Hide the code to old browsers
function square(i) {
    return i * i;
}
document.write("The square of 5 is " + square(5));
// The code is hidden with an HTML comment -->
</script>

<noscript>
<p>Your browser doesn't have support for scripting</p>
<p>Alternative access to <a
    href="http://someplace.com/data">the data</a>
</noscript>
```

## Language Syntax (I)

- Basic Syntax
- Variable Declaration
- Variable Scope
- Special Characters
- Operators
- Keywords
- Conditional Statements
- Iteration Statements
- Object Manipulation Statements

## Language Syntax (and II)

- Function Declaration
- Objects
  - Object Declaration
  - Global Object
  - Array Objects
  - String Objects
  - Math Object
  - Date Objects

## Basic Syntax (I)

- Syntax based on C, C++, and Java
- Case sensitive
- Semicolon (;) at the end:
  - Semicolons are automatically inserted into the source...
  - ...but certain statements (do-while, continue, break, etc.) must be terminated with semicolons
  - Better to write semicolon always
- Block of code: { . . . }

## Basic Syntax (and II)

- Comments:
  - Single line:  
`// only one line`
  - Multi line:  
`/* a comment with  
two lines */`

## Variable Declaration (I)

- Not necessary to declare
- Variable statement:
  - **var** variable1, variable2, ...;
- Rules:
  - Letters, numbers, \$ or \_
  - First character: no number
  - Variable name  $\neq$  Keyword
- Initial value  $\rightarrow$  **undefined**

## Variable Declaration (and II)

- Loosely typed
- Six language data types:
  - Undefined → undefined
  - Null → null
  - Boolean → true and false
  - String (“ and “”)
  - Number (integer and double-precision)
  - Object

## Variable Scope

- Global (program) or local (function)
- You have to use **var** to declare a local variable with the same name as a global variable

## Special Characters

Carácter	Significado
\b	Retroceso ( <i>backspace</i> )
\f	Salto de página ( <i>form feed</i> )
\n	Salto de línea ( <i>new line</i> )
\r	Retorno de carro ( <i>carriage return</i> )
\t	Tabulador
\'	Apóstrofe o comilla simple
\"	Comilla doble
\\	Barra invertida ( <i>backslash</i> )
\XXX	El carácter de la codificación Latin-1 especificado por los tres dígitos octales entre 0 y 377.
\xxx	El carácter de la codificación Latin-1 especificado por los dos dígitos hexadecimales entre 00 y FF.
\uXXXX	El carácter Unicode especificado por los cuatro dígitos hexadecimales entre 0000 y FFFF.

## Operators

Precedencia de los operadores	
Tipo de operador	Operador
Coma	,
Asignación	= += -= *= /= %= <<= >>= >>>= &= ^=  =
Condicional	?:
O lógico (OR)	
Y lógico (AND)	&&
O bit a bit (OR)	
O exclusiva bit a bit (XOR)	^
Y bit a bit (AND)	&
Igualdad	== != === !==
Relacional	< <= > >=
Desplazamiento bit a bit	<< >> >>>
Suma y resta	+ -
Multiplicación y división	* / %
Negación e incremento	! ~ - ++ -- typeof void delete
Llamada a función	()
Creación de instancias	new
Miembro	· []

## Keywords

Palabras reservadas de JavaScript			
abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

## Conditional Statements

```
if(condicion) {  
    sen1  
}  
[else {  
    sen2  
}]
```

```
switch(expression) {  
    case et1 :  
        sen1;  
        [break;]  
    case et2 :  
        sen2;  
        [break;]  
    ...  
    [default :  
        sen;]  
}
```

## Iteration Statements

```
for ([expInicial]; [condicion]; [expIncremento]) {  
    sentencias  
}
```

```
do {  
    sentencias  
} while (condicion)
```

```
while (condicion)  
{  
    sentencias  
}
```

**break** → Ends iterations

**continue** → Follow to the next iteration

## Object Manipulation Statements

```
for(variable in objeto)  
{  
    sentencias  
}
```

```
with(obj)  
{  
    sentencias  
}
```

## Function Declaration (I)

```
function nombre([arg1 [, arg2 [, ...]]) {  
    sentencias  
}
```

**return** → Ceases execution and returns a value to the caller

## Function Declaration (and II)

- Parameters:
  - Basic data types (boolean, string, number):  
by value (a copy of the caller's value)
  - Complex data types (arrays and objects):  
by reference (a pointer to the caller's value)

## Objects (I)

- JavaScript is based on objects but it is not a pure object oriented language → No classes, no inheritance, no polymorphism, etc.
- Object manipulation statements:
  - for (... in ...)
  - with()
- Operator “.” o “[ ]” (associative array notation)

```
window.status = "Welcome to JavaScript";  
window.alert("2 + 2 = " + (2 + 2));
```

```
window["status"] = "Welcome to JavaScript";  
window["alert"]("2 + 2 = " + (2 + 2));
```

## Objects (II)

- Object creation:
  - From values:  

```
object = {prop1:val1, ..., propN:valN};
```
  - From a function constructor:  

```
function ObjConstructor(arg1, ..., argN)  
{  
  this.prop1 = arg1; ...; this.propN = argN;  
}
```

```
object = new ObjConstructor(val1, ..., valN);
```

## Objects (III)

- Object methods:
  - Assign the name of a function to a property
  - Use `this` to access properties and methods of the object
- Properties can be added to objects dynamically by assigning values to them:
  - Constructors are not required to name or assign values to all properties
- Remove an object:
  - `delete`

## Objects (IV)

- Example:

```
function Person1(name, surname) {
  this.name = name;
  this.surname = surname;

  this.fullName = function() {
    return this.name + " " + this.surname;
  }
}

var peter = new Person1("Peter", "Smith");
document.writeln(peter.fullName());
```

## Objects (V)

- Problem:
  - Each object has its own function  
`fullName()`
  - Poor performance and consumes more resources

## Objects (VI)

- Solution: `prototype`
  - It is a global property shared by all the objects of the same type

- Example:

```
function Person2(name, surname) {  
    this.name = name;  
    this.surname = surname;  
}  
Person2.prototype.fullName = function() {  
    return this.name + " " + this.surname;  
}
```

## Objects (VII)

- Inheritance:
  - Prototype-based inheritance
  - Different from class-based object-oriented language

## Objects (VIII)

- Example:

```
Object.extend = function(destination, source) {
  for(var property in source)
    destination[property] = source[property];
  return destination;
}

function Person(name, surname) {
  this.name = name;
  this.surname = surname;
}

Person.prototype.fullName = function() {
  return this.name + " " + this.surname;
}
```

## Ajax Technology in Web Programming

```
function Student(name, surname, course) {
  this.name = name;
  this.surname = surname;
  this.course = course;
}

Object.extend(Student, Person);

Student.prototype.fullDescription = function() {
  return this.name + " " + this.surname + ": " +
  this.course;
}
```

## Ajax Technology in Web Programming

### Objects (and IX)

- Built-in objects available:
  - Global object
  - Array objects
  - String objects
  - Boolean objects
  - Number objects
  - Math object (only one)
  - Date objects
  - RegExp objects
  - Error objects

## Global Object

- Available in all the scopes, it does not have a name
- Properties:
  - NaN
  - Infinity
  - undefined
- Methods:
  - eval(x)
  - parseInt(string, radix)
  - parseFloat(string)
  - isNaN(number)
  - isFinite(number)

## Array Objects

- Properties:
  - length
- Methods:
  - concat(), join(), pop(), push(), reverse(), shift(), slice(), sort(), splice(), unshift()

## String Objects (I)

- Strings in JavaScript:
  - Constant strings (" and "")
  - String object
- JavaScript automatically converts constant strings to `String` objects when it is needed
- Properties:
  - `length`

## String Objects (and II)

- Methods:
  - `charAt(index): 0 ... length - 1`
  - `charCodeAt(index): 0 ... length - 1`
  - `concat(cad1, cad2, ..., cadn) → "+"`
  - `indexOf(searchValue, position)`
  - `lastIndexOf(searchValue, position)`
  - `replace(searchValue, replaceValue)`
  - `slice(start, end)`
  - `split(separator, limit)`
  - `substring(start, end)`
  - `toLowerCase(), toUpperCase()`
  - ...

## Math Object

- A single object: it is not possible to use the Math object as a constructor with the new operator
- Properties
  - E, LN10, LN2, LOG2E, LOG10E, PI, SQRT1\_2, SQRT2
- Methods:
  - abs(), acos(), asin(), atan(), cos(), sin(), tan(), exp(), ceil(), floor(), log(), max(), min(), pow(), random(), round(), sqrt(), ...

## Date Objects

- Indicates a particular instant in time (measured in milliseconds since 01 January, 1970)
- Constructor:
  - new Date()
  - new Date(year, month, date, hours, minutes, seconds, ms)
- Methods:
  - getTime(), getFullYear(), getMonth(), getDate(), getDay(), getHours(), getMinutes(), getSeconds(), getMilliseconds(), ...
  - setTime(), setFullYear(), setMonth(), setDate(), setDay(), setHours(), setMinutes(), setSeconds(), setMilliseconds(), ...

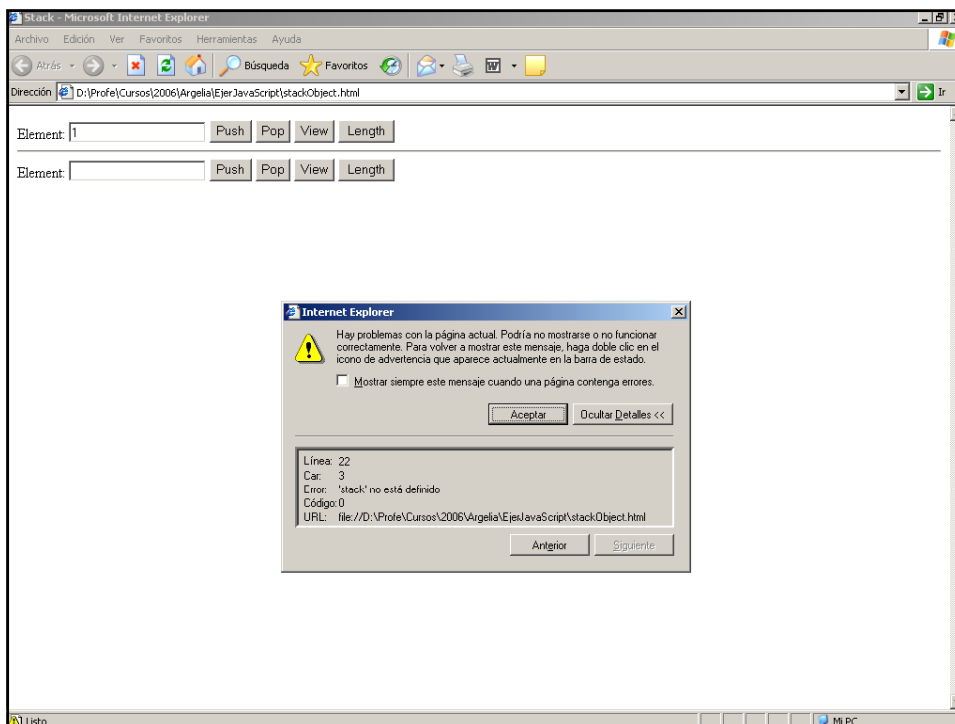
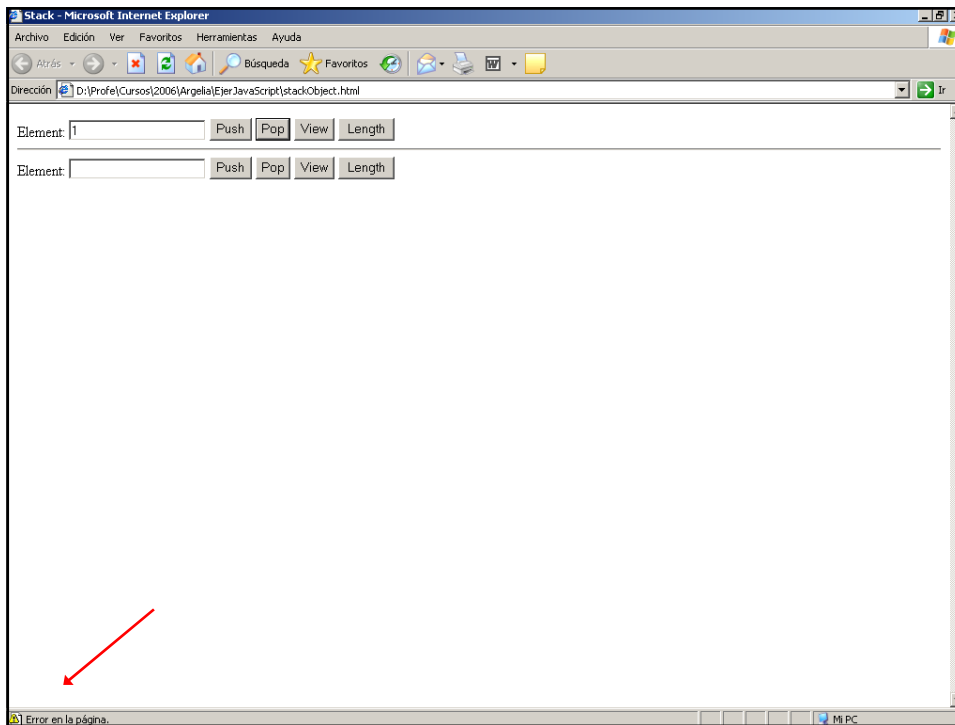
## Browsers' tools

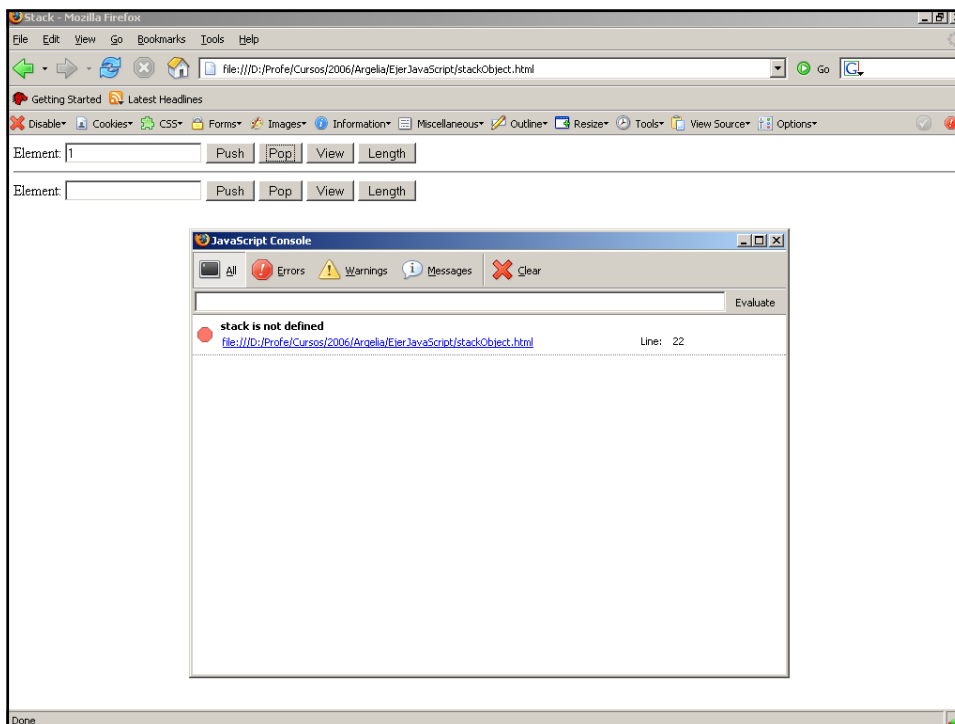
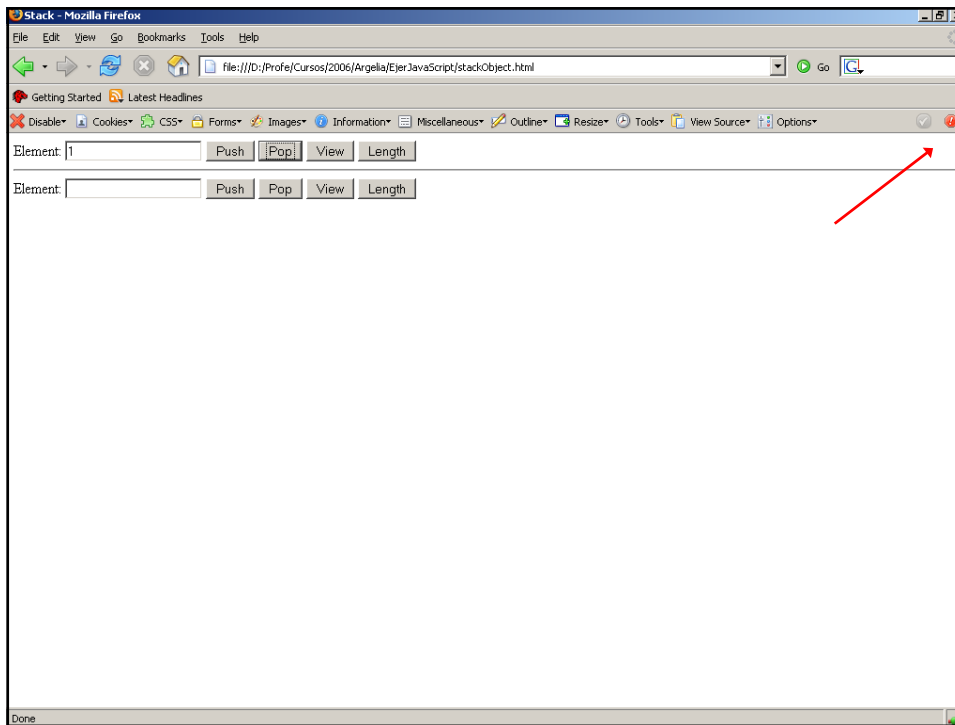
- Microsoft Internet Explorer 6.0
- Mozilla FireFox 1.5
  - Tools → JavaScript Console
- Opera 8.52
  - Tools → Advances → JavaScript Console

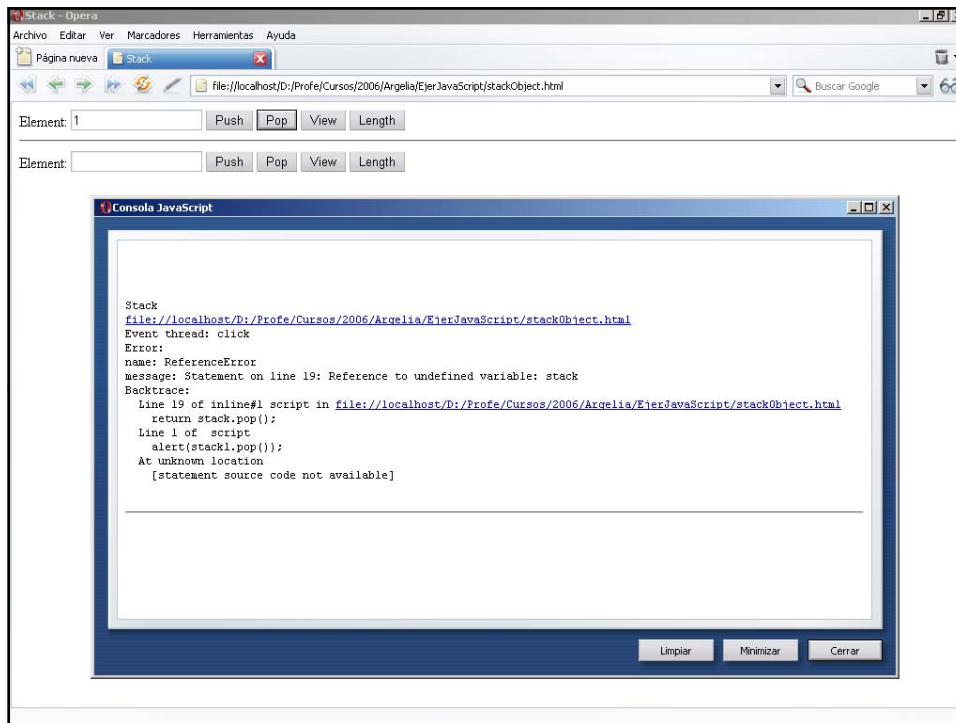
## Browsers' tools

- Calling function from an undefined object:

```
function pop()  
{  
    return stack.pop();  
}
```







## Ajax Technology in Web Programming

### More Information

- <http://www.alvit.de/handbook/>
- <http://www.w3schools.com/>

