

The Web, revisited

WEB 2.0

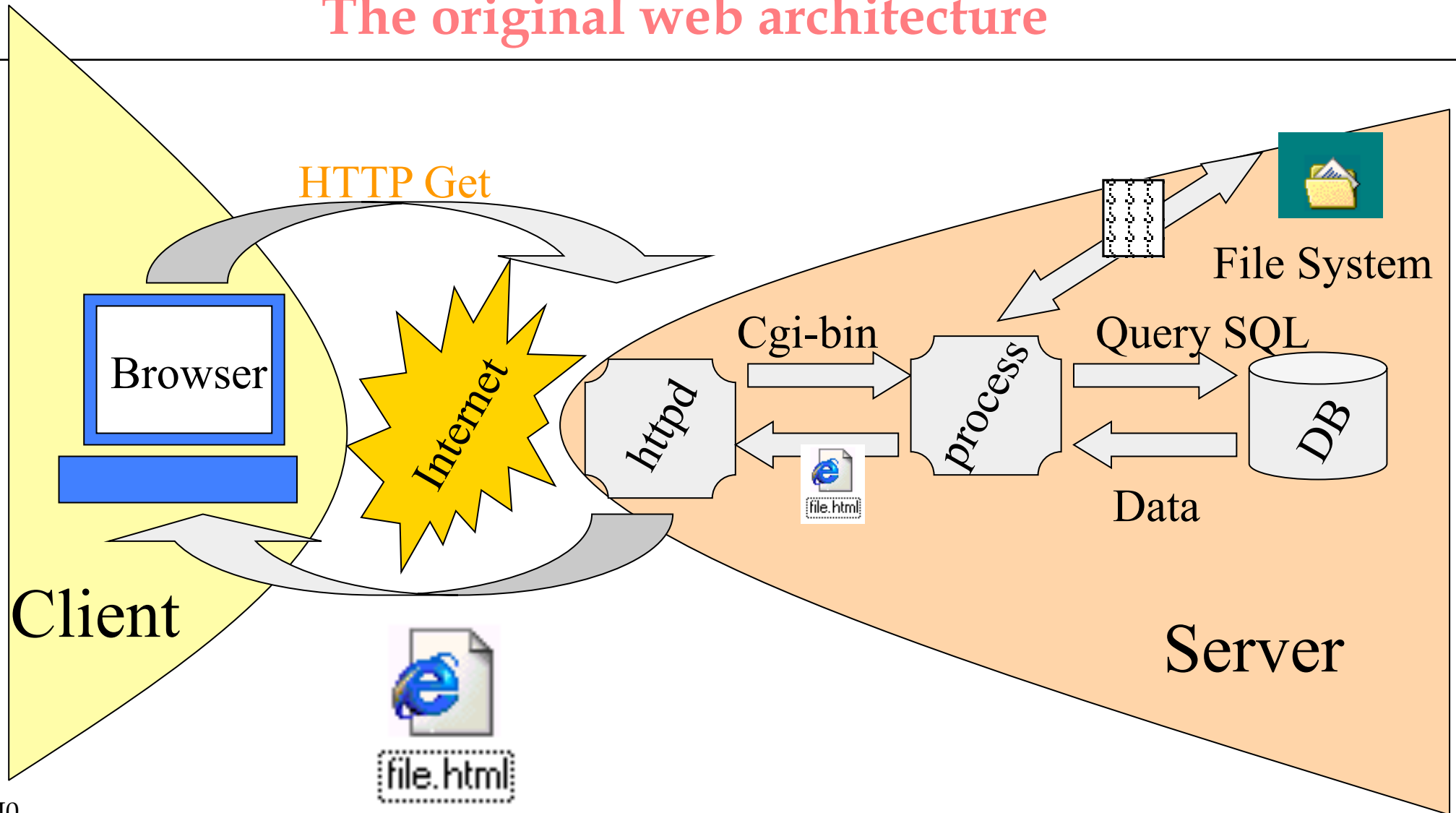
marco.ronchetti@unitn.it

The old web: 1994

HTML pages (hyperlinks)
+ static graphics
+ cgi (active engines)
+ some separated dynamic graphics
(Applets)

HTTP is a stateless protocol: cookies

The original web architecture



Evolution of the web: 1

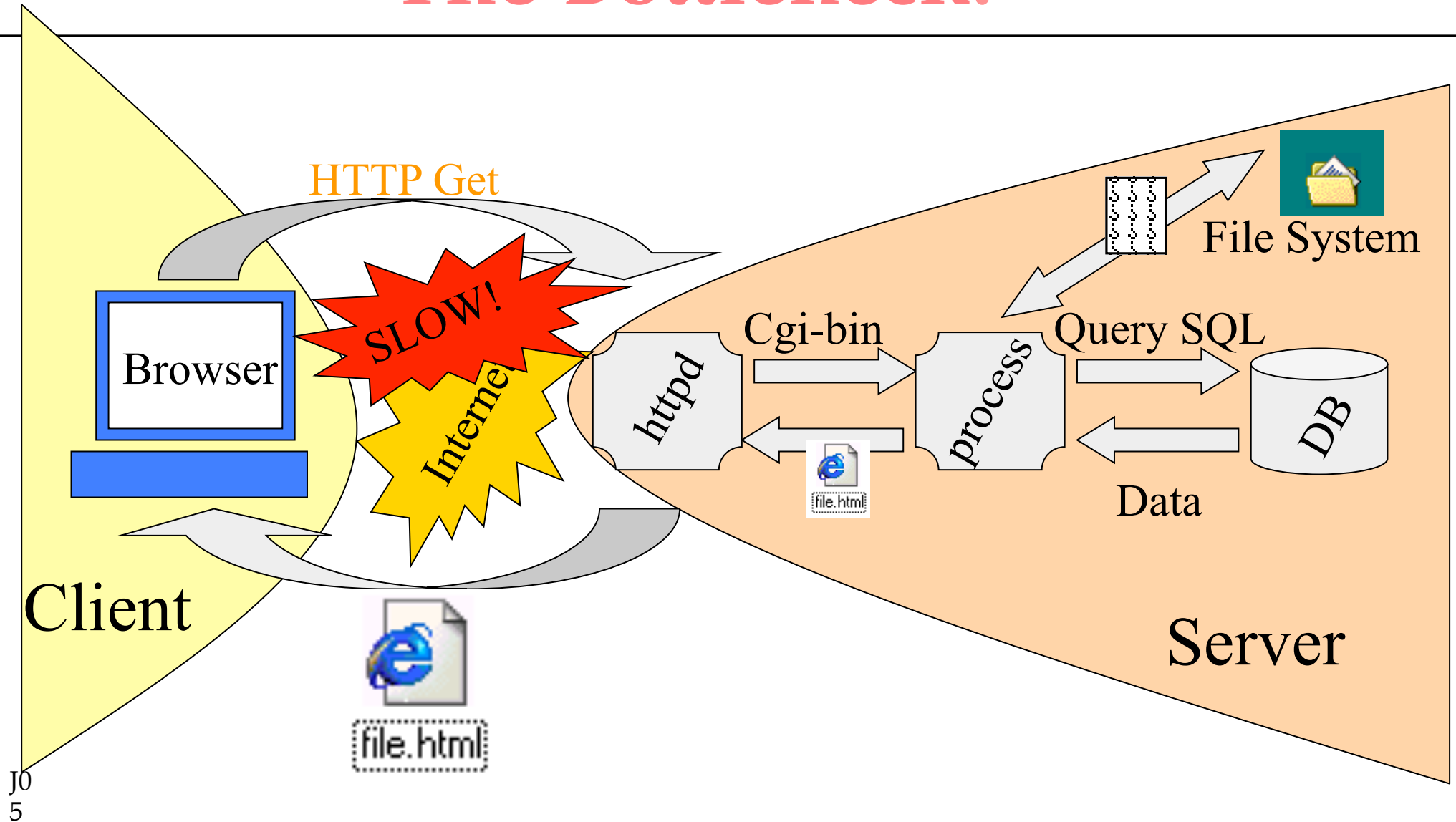
Better dynamic engines

Servlets, ASP, JSP (+ Php, Perl, Python...)

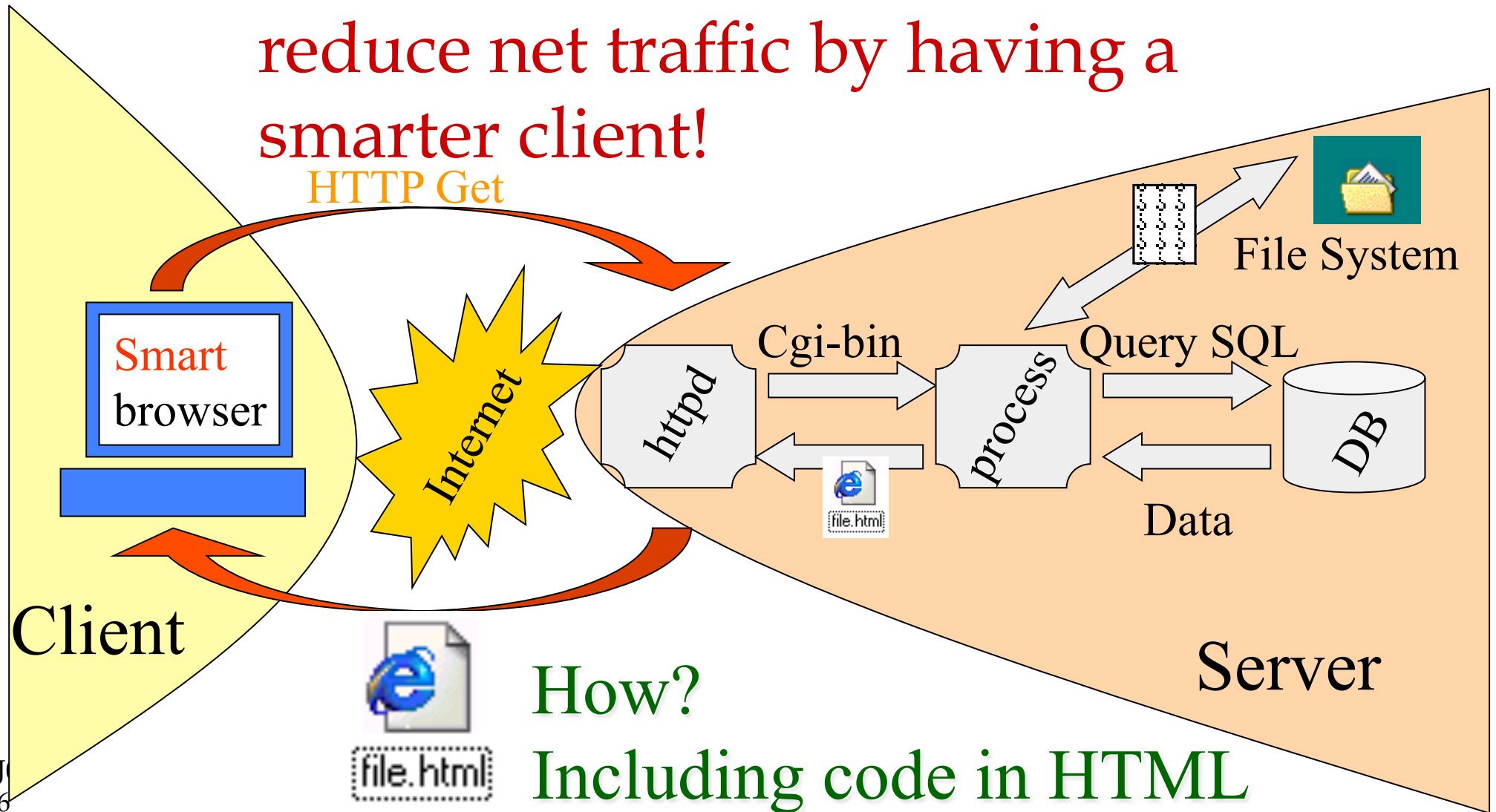
Better Server-side organization

EJB, frameworks (Struts, Hybernate, Spring)

The Bottleneck!



The solution:



Evolution of the web: 2

Better control of the browser

Javascript + DOM

Applet-Javascript interaction

Better separation of content and presentation

CSS (DHTML=HTML4+Javascript+DOM
+CSS)

XML+XSLT, Cocoon (XHTML)

Evolution of the web: 3

Better construction of interfaces (widgets)

.Net

Java Server Faces

Scripting the Web

Client Side ECMAScript & Document Object Model

JavaScript History

- Java was born as “**LiveScript**” at the beginning of the 94’ s.
- Name changed into JavaScript (name owned by Netscape)
- Microsoft responds with Vbscript
- Microsoft introduces JScript (dialect of Javascript)
- A standard is defined: ECMAScript (ECMA-262, ISO-16262)

JavaScript Myths

JavaScript is NOT simple

Simple tasks are indeed simple

JavaScript is NOT Java

	Java	JavaScript
Browser Control	NO	YES
Networking	YES	NO
Graphics	YES	Partial

JavaScript is...

Scripted (not compiled)

Powerful

Object-based

Cross-Platform

Client and Server

JavaScript allows...

Dynamic Web Sites

Dynamic HTML (DHTML)

Interactive Pages/Forms

Server-Side CGI Functionality

Application Development

JavaScript can...

Build Objects

Use Events

Enforce Security

Embed or Componentize

- Syntax is C-like (C++-like, Java-like)
case-sensitive,
statements end with (optional) semicolon ;
`//comment` `/*comment*/`
operators (`=`, `*`, `+`, `++`, `+=`, `!=`, `==`, `&&`, ...)
- Basic data types
integer, floating point, strings (more later)
- Loosely typed variables (Basic-like) `var x=3;`

- `if (expression) {statements} else {statements}`
- `switch (expression) {
 case value: statements; break;
 ...
 default: statements; break;
}`
- `while (expression) {statements}`
- `do (expression) while {statements}`
- `for (initialize ; test ; increment) {statements}`

Between `<SCRIPT>` and `</SCRIPT>` tags

Between `<SERVER>` and `</SERVER>` tags

In a `<SCRIPT SRC="url"></SCRIPT>` tag

In an event handler:

`<INPUT TYPE="button" VALUE="Ok"`

`onClick="js code">`

`<B onMouseOver="Jscore">hello<B>`

a="foo"; b='tball'

Useful methods:

a+b => football

a<b => true

a.charAt(0) => f

indexOf(substring), lastIndexOf(substring)

charCodeAt(n), fromCharCode(value,...)

concat(value,...), slice(start,end)

toLowerCase(), toUpperCase()

replace(regex,string), search(regex)

a="foo";

TAG-related methods:

a.bold() => foo

big(), blink(), fontcolor(), fontsize(), small(),

strike(), sup()

anchor(), link()



```
function f(x) {return x*x}
```

```
function add(x,y) {return x+y};
```

```
function multiply(x,y) {return x*y};
```

```
function operate(op,x,y) {return op(x,y)};
```

```
operate(add,3,2); => 5
```

<HTML>

<HEAD>

<SCRIPT>

```
function fact(n) {  
    if (n==1) return n;  
    return n*fact(n-1);  
}
```

</SCRIPT>

</HEAD>

...

<BODY>

<H2>Table of Factorial Numbers </H2>

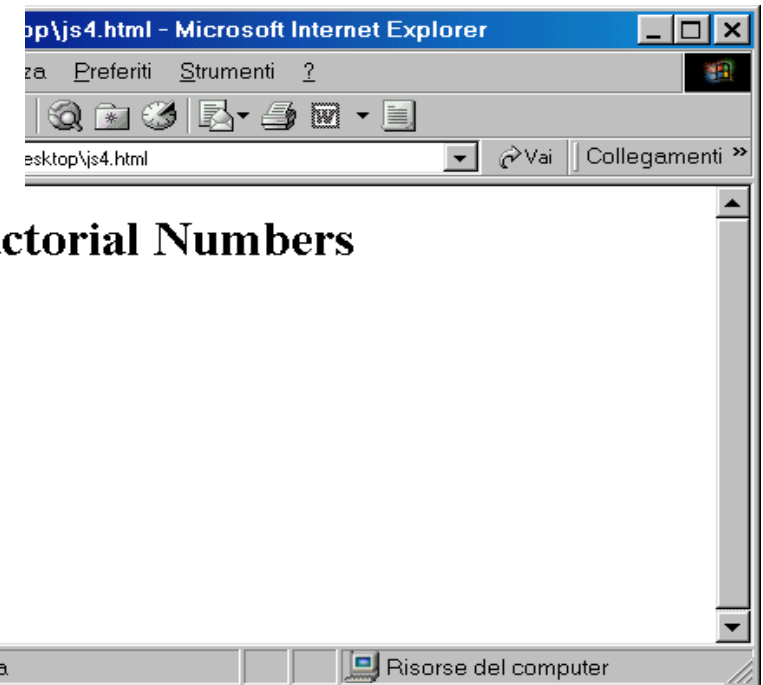
<SCRIPT>

```
for (i=1; i<10; i++) {  
    document.write(i+"!="+fact(i));  
    document.write("<BR>");  
}
```

</SCRIPT>

</BODY>

</HTML>



<BODY>

<SCRIPT>

n=window.prompt("Give me the value of n",3)

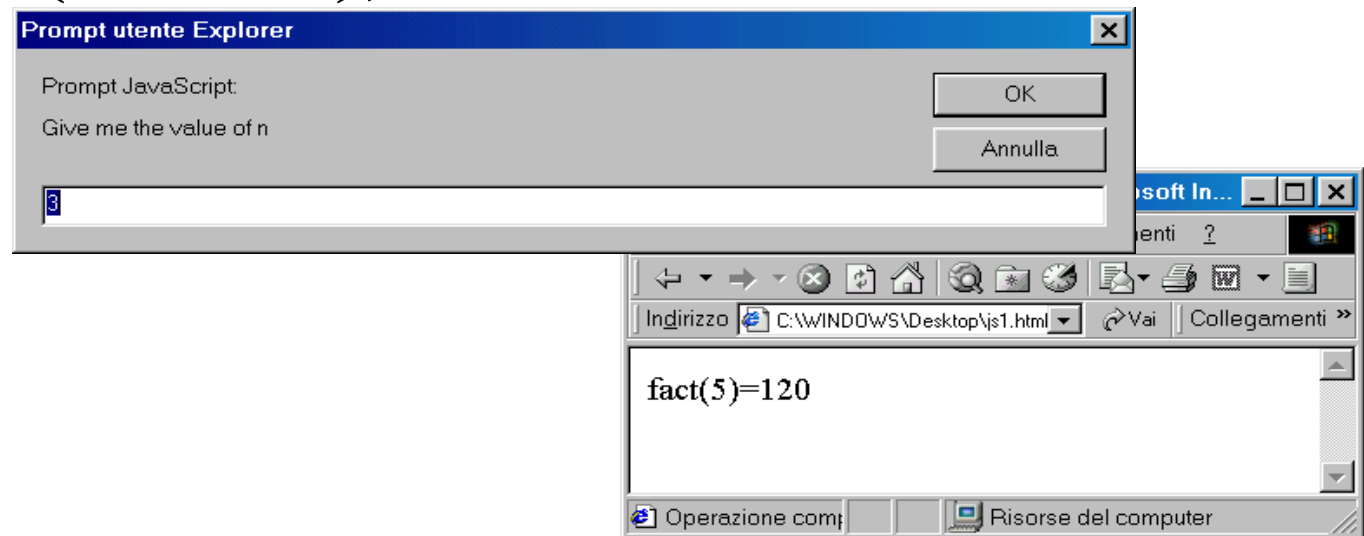
document.write("fact("+n+")="+fact(n));

**document.write("
");**

</SCRIPT>

</BODY>

</HTML>



Object: A data structure with methods;
a special method is the “constructor”.

```
function Rectangle(w, h) {  
  this.width=w;  
  this.height=h;  
  this.area=function(){return this.width*this.height}  
}
```

Instance variables

method

```
a=new Rectangle(3,4);    a.area() => 12    a.width => 3
```

Actually, JavaScript does NOT have classes and inheritance.

Moreover, the approach we have shown is not the most efficient in terms of memory allocation.

It would be better to use the “prototype” feature, which can be considered a STATIC object

```
Rectangle.prototype.area=function(){return this.w*this.h}
```

```
a = new Array()
```

```
a[0]=3; a[1]="hello"; a[10]=new Rectangle(2,2);
```

```
a.length() => 11
```

Arrays can be

SPARSE, INHOMOGENEOUS, ASSOCIATIVE

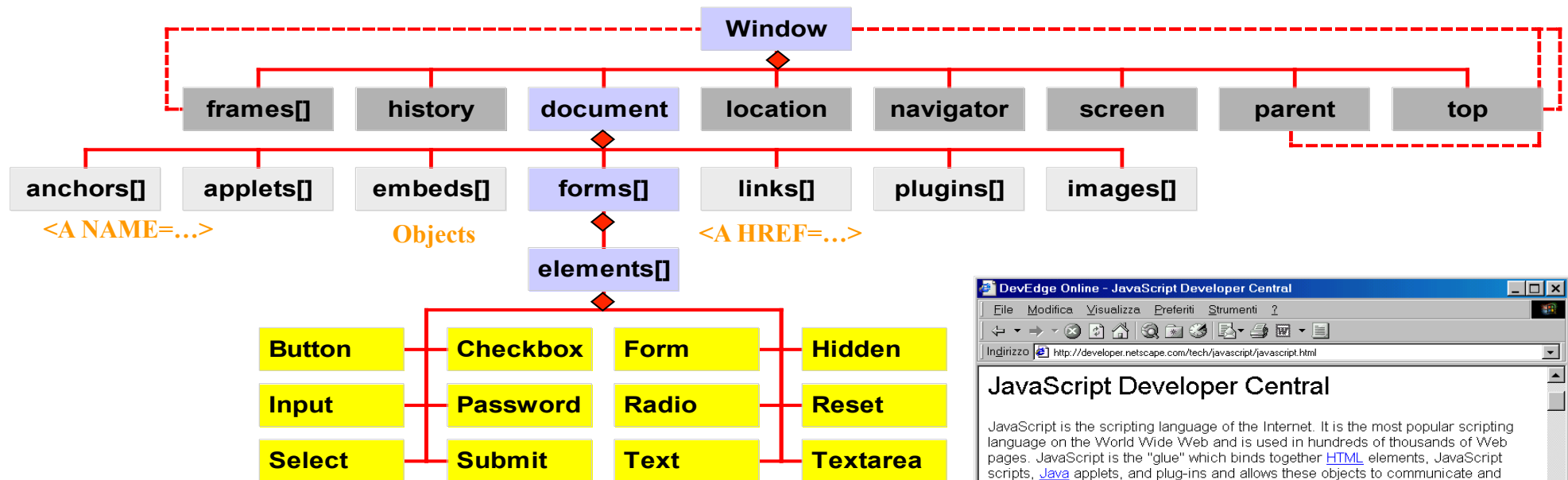
```
a["name"]="Jaric"
```

```
z=new Rectangle(3,4);  z["width"] ⇔ z.width
```

DOM

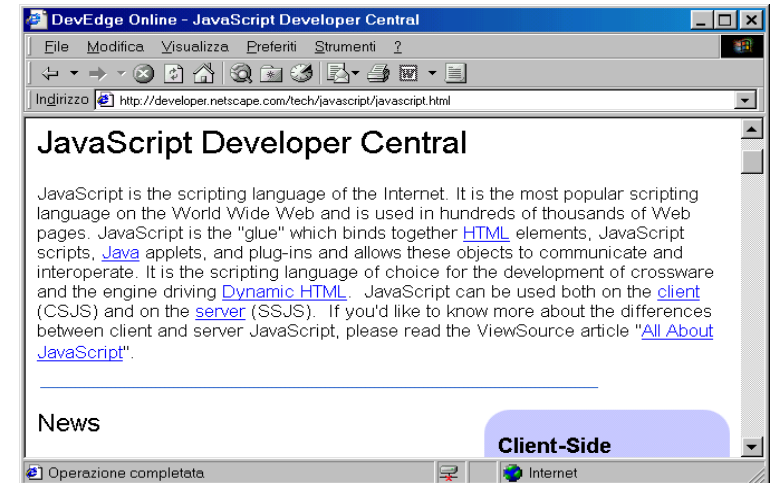
Object hierarchy

The Most
Important Slide



Symbol **◆** means containment (has-a)

Dashed line means "is an instance of"



“A web browser window or frame”

Main properties

Objects

history

frames[]

document

location

navigator

screen

parent – top

Other properties

status – defaultStatus

name

Main methods

alert(), prompt(), confirm()

focus(), blur()

moveBy(), moveTo()

resizeBy(), resizeTo()

scroll(), scrollBy(), scrollTo()

setInterval(), clearInterval()

setTimeout(), clearTimeout()

“Information about the display”

Main properties

availHeight, availWidth

height, width

colorDepth, pixelDepth

hash

DOM

Navigator

“Information about the browser in use”

Main properties

appName

appVersion

Platform

Main methods

javaEnabled()

Other properties

Info on available plugins, but only in Netscape
Navigator!

“The URL history of the browser”

Main properties

length

Main methods

back()

forward()

go(+/-n)

go(target_substring)

“The specification of the current URL”

Main properties

href

protocol, hostname, port

search

hash

Main methods

reload()

replace()

“An HTML document”

Main properties

Arrays of Component Objects

anchors[]

applets[]

embeds[]

forms[]

links[]

plugins[]

Other properties

bgColor, fgColor, linkColor, vlinkColor

lastModified

title, URL, referrer, cookie

Main methods

open()

close()

clear()

write()

“An image embedded in an HTML document”

Main properties

border *[width in pixels]*

height

width

src *[URL of the image to be displayed]*

onClick	User clicks once. (*)	Link, button
onDbClick	User clicks twice	Document, Image, Link, button
onMouseDown	User presses mouse button (*)	Document, Image, Link, button
onMouseUp	User releases mouse button (*)	Document, Image, Link, button
onMouseOver	Mouse moves over element	Link, Image, Layer
onMouseOut	Mouse moves off element	Link, Image, Layer
onKeyDown	User presses key (*)	Document, Image, Link, Text elements
onKeyUp	User releases key	Document, Image, Link, Text elements
onKeyPress	KeyDown+KeyUp (*)	Document, Image, Link, Text elements

onFocus	Element gains focus	TextElement, Window, all form elements
onBlur	Element loses focus	TextElement, Window, all form elements
onChange	User selects/deselects a text and moves focus away	Select, text input elements
onError	Error while loading image	Image
onAbort	Loading interrupted	Image
onLoad	Document or image finishes loading	Window, Image
onUnload	Document is unloaded	Window
onResize	Window is resized	Window
onReset	Form reset requested (*)	Form
onSubmit	Form submission requested(*)	Form

(*) Return false to cancel default action

DOM

Properties

Input

Methods

Objects

Button				X		X			X	X	X	X	X		X		X	X
Checkbox	X	X		X		X			X	X	X	X	X		X		X	X
Radio	X	X		X		X			X	X	X	X	X		X		X	X
Reset				X		X			X	X	X	X	X		X		X	X
Submit				X		X			X	X	X	X	X		X		X	X
Text			X	X		X			X	X	X		X	X	X	X		X
Textarea			X	X		X			X	X	X		X	X	X	X		X
Password			X	X		X			X	X	X		X	X	X	X		X
FileUpload			X	X		X			X	X	X		X	X	X	X		X
Select				X	X	X	X	X	X		X	X	X		X	X		X
Hidden				X		X			X	X								

Event Handlers

J0

Properties

checked

defaultChecked

name

options[]

selectedIndex

type

value

onblur

onchange

onclick

onfocus

“An HTML input form”

Main properties

action *[destination URL]*

method *[get/post]*

name *[name of Form]*

name *[destination Window]*

Main methods

reset()

submit()

Elements[] *[list ; of contained elements]*

DOM

Events

```
<HTML>
<HEAD>
<TITLE>Form Example</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
function setColor() {
    var choice;

    choice = document.colorForm.color.selectedIndex;

    switch(choice) {
        case 0: document.bgColor = "FF0000"; break;
        case 1: document.bgColor = "00FF00"; break;
        case 2: document.bgColor = "0000FF"; break;
        case 3: document.bgColor = "FFFFFF"; break;
        case 4: document.bgColor = "FFFF00"; break;
        case 5: document.bgColor = "FF00FF"; break;
    }
}
</SCRIPT>
```



<BODY>

<CENTER><H1>Color Changer</H1></CENTER>

**

**

Select Your Favorite Background Color:

<FORM NAME="colorForm">

<SELECT NAME="color" onChange=setColor() >

<OPTION VALUE="red">Red

<OPTION VALUE="green">Green

<OPTION VALUE="blue">Blue

<OPTION VALUE="white">White

<OPTION VALUE="yellow">Yellow

<OPTION

VALUE="purple">Purple

</SELECT>

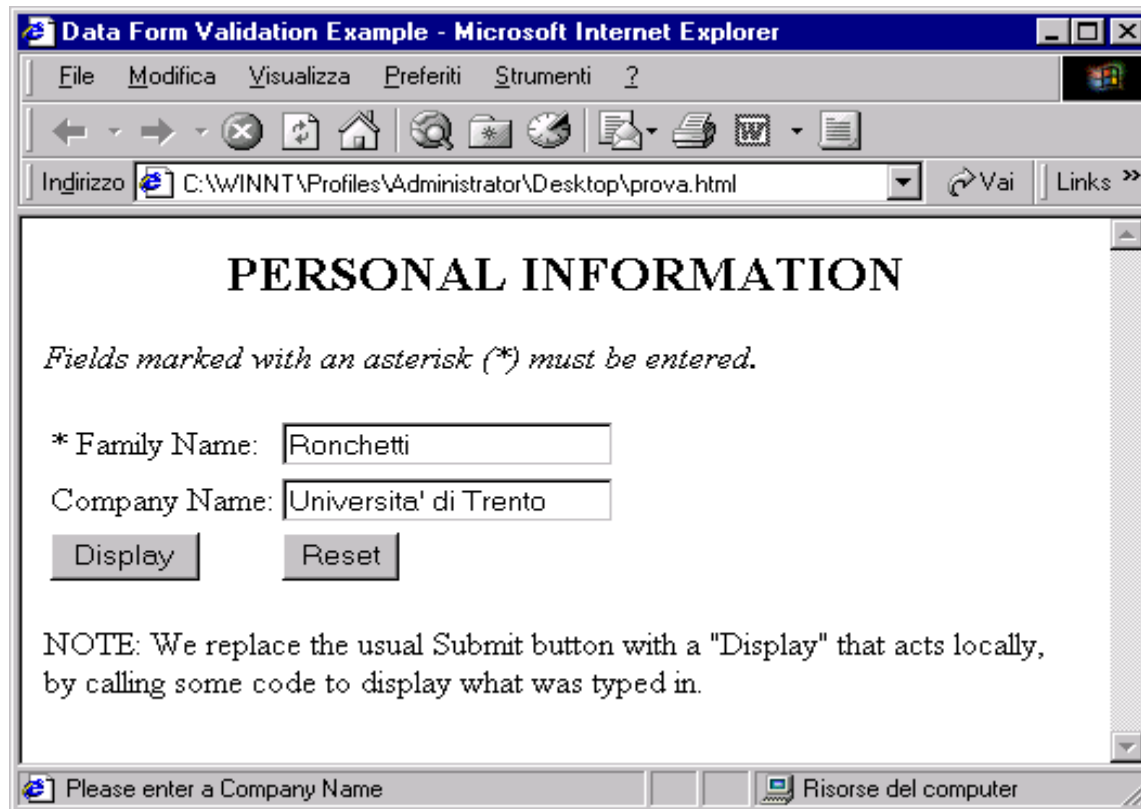
</FORM>

</BODY>

</HTML>

A more complex example -1

A simple data entry validation page



PERSONAL INFORMATION

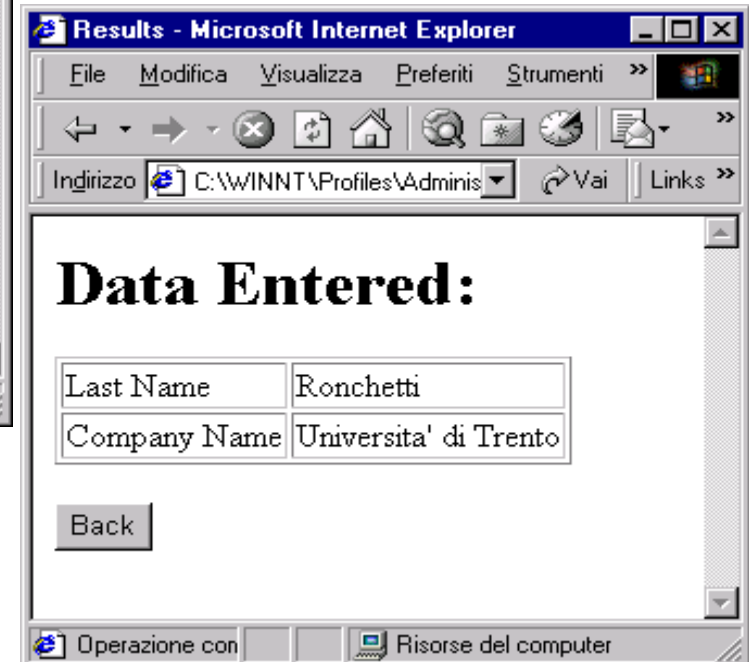
Fields marked with an asterisk () must be entered.*

* Family Name:

Company Name:

NOTE: We replace the usual Submit button with a "Display" that acts locally, by calling some code to display what was typed in.

Please enter a Company Name



Data Entered:

Last Name	Ronchetti
Company Name	Universita' di Trento

Operazione con

A more complex example -2

Start of file “FormValidation.html”

<HTML>

<HEAD>

<TITLE>Data Form Validation Example</TITLE>

<SCRIPT LANGUAGE="JavaScript1.1" SRC="FormCheck.js"></SCRIPT>

Load file “FormCheck.js”,
which contains several JavaScript functions

A more complex example -3

```
function isEmpty(s)  
{ return ((s == null) || (s.length == 0))  
}
```

Check that the string
“s” is not empty

```
function warnEmpty (theField, s)  
{  
  var mPrefix = "You did not enter a value into the ";  
  var mSuffix = " field. This is a required field. Please enter it now.";  
  theField.focus();  
  alert(mPrefix + s + mSuffix);  
  return false;  
}
```

Issue a warning
message

All this is contained in the file “FormCheck.js”

A more complex example -4

```
function promptEntry (s)
{ window.status = "Please enter a " + s;
}
```

Type a message in the status bar

```
function validatePersonalInfo(form)
```

Validate the form

```
{ return (
    checkString(form.elements["LastName"],sLastName)
)
}
```

(should run over all fields
And perform suitable checks)

```
function checkString (theField, s)
```

```
{
    if (isEmpty(theField.value)) return warnEmpty (theField, s);
    else return true;
}
```

Check that “theField”
is not empty

All this is contained in the file “FormCheck.js”

A more complex example -5

<SCRIPT>

Global variables

**var sCompany="Company Name"; var sLastName="Last Name"; var
form="PersonallInfo";**

Value-printing
function

function displayPersonallInfo(form)

```
{ var outputTable = "<HTML><HEAD><TITLE>Results</TITLE></HEAD>" +  
  "<BODY><H1>Data Entered:</H1><TABLE BORDER=1>" +  
  "<TR><TD>" + sLastName + "</TD><TD>" + form.elements["LastName"].value + "</  
  TD></TR>" +  
  "<TR><TD>" + sCompany + "</TD><TD>" + form.elements["Company"].value + "</  
  TD></TR></TABLE><FORM>" +  
  "<INPUT TYPE=\"BUTTON\" NAME=\"Back\" VALUE=\"Back\" onClick=  
  \"history.back()\"> </FORM></BODY></HTML>"  
  document.writeln(outputTable)  
  document.close()  
  return true  
} </SCRIPT>
```

Add a Button to go
back in history

</HEAD>

End of “HEAD” portion of “FormValidation.html”

A more complex example -6

```
<BODY BGCOLOR="#ffffff">
<CENTER><H2>PERSONAL INFORMATION </H2></CENTER>
<P><P><I>Fields marked with an asterisk (*) must be entered.</I>
<FORM NAME="PersonallInfo">
<TABLE>
<TR>
  <TD>* Family Name:</TD>
  <TD><INPUT TYPE="text" NAME="LastName"
    onFocus="promptEntry(sLastName)"
    onChange="checkString(this,sLastName)" ></TD>
</TR>
<TR>
  <TD>Company Name:</TD>
  <TD><INPUT TYPE="text" NAME="Company"
    onFocus="promptEntry(sCompany)"></TD>
</TR>
```

First Field

Second Field

Start of “BODY” portion of “FormValidation.html”

A more complex example -7

```
<TR>
  <TD>
    <INPUT TYPE="BUTTON" NAME="fakeSubmit" VALUE="Display"
      onClick="if (validatePersonalInfo(this.form)) displayPersonalInfo(this.form); ">
    </TD>
    <TD><INPUT TYPE = "reset" VALUE = "Reset">
    </TD>
</TR>
</TABLE>
<P> NOTE: We replace the usual Submit button with a "Display" that acts locally,
<BR>by calling some code to display what was typed in.
</FORM>
</BODY>
</HTML>
```

First Button

Second Button

End of file “FormValidation.html”

DOM

Applet

“An applet embedded in a Web page”

Properties

Same as the public fields
of the Java applet

Methods

Same as the public methods
of the Java applet

Server-Side JavaScript

Not discussed here!

A substitute for CGI.

Server-dependent technology to process the Web page *before* passing it to the client.

(The Netscape SSJS object model is different from the Microsoft ASP object model, although JavaScript can be used as SSLanguage for ASP)

See [http://en.wikipedia.org/wiki/Server-side JavaScript](http://en.wikipedia.org/wiki/Server-side_JavaScript)

Layout engines

A **web browser engine** (sometimes called **layout engine** or **rendering engine**) is a software component that takes

- marked up content (such as HTML, XML, image files, etc.) and
- formatting information (such as CSS, XSL, etc.)

and displays the formatted content on the screen.

It draws onto the content area of a window, which is displayed on a monitor or a printer.

- Blink (Chromium)
- Gecko (Mozilla)
- Trident (IE)
- Webkit (Safari, Android)

See:

[https://en.wikipedia.org/wiki/Comparison_of_layout_engines_\(ECMAScript\)](https://en.wikipedia.org/wiki/Comparison_of_layout_engines_(ECMAScript))

ECMAScript Engine

An ECMAScript engine is a program that
**executes source code written in a version
of the ECMAScript** language standard

See https://en.wikipedia.org/wiki/List_of_ECMAScript_engines

Examples:

- V8 (chrome)
- SpiderMonkey (Mozilla)
- Chakra (I.E.)
- Squirrelfish/Nitro (Apple)
- Nashorn (Oracle – JDK)

References

Standard ECMA-262 ECMAScript Language Specification:

[http://www.ecma-international.org/publications/standards/
Ecma-262.htm](http://www.ecma-international.org/publications/standards/Ecma-262.htm)

Books:

- D.Flanagan “Javascript. The definitive guide” O’ Reilly.
- D.Goodman “Dynamic HTML. The definitive reference” O’ Reilly