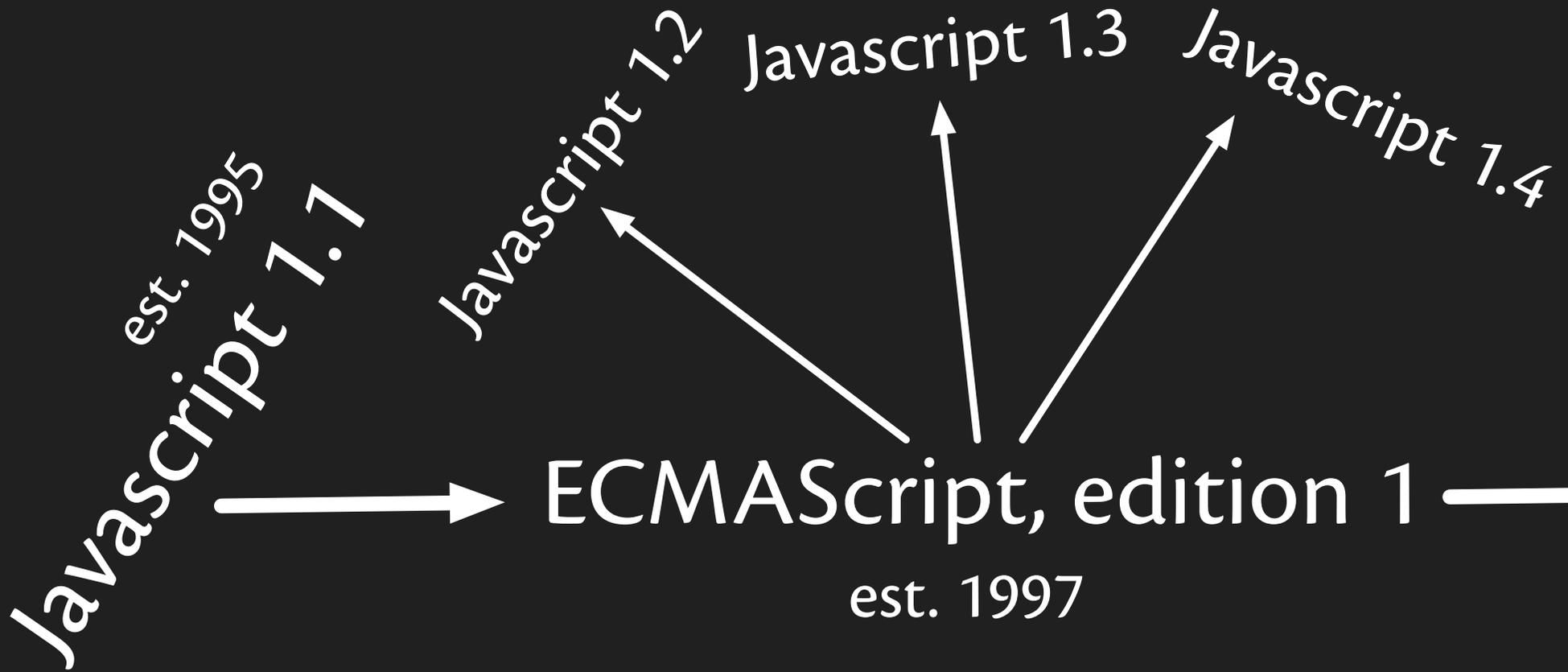


javascript BOOTCAMP

by amy hoy, www.slash7.com

Javascript is a *real*
~~toy~~ language





ECMAScript, edition 3

est. 1999

Javascript 1.5



Javascript 1.6





Follow Along

www.squarefree.com/shell/shell.html

Syntax & Style

////// Semi-colons are not required

```
var foo = getBar()
```

// --> [no error]

```
var foo = getBar() fooBar()
```

// ! SYNTAX ERROR !

////// CamelCase is the norm

```
if(fooBar == bazBat) {}
```

////// Object attributes & methods attached with '.'

```
someObject.someFunction()
```

whirlwind tour

look ma, no primitives!

```
var string = "Hello, I'm a string!"  
// --> "Hello, I'm a string!"
```

```
string.length  
// --> 20
```

```
"Hello, I'm a string!".length  
// --> 20
```

data types are fun!

- * Number
- * String
- * Boolean
- * null
- * undefined
- * RegExp

creating numbers

```
var number = 5
```

```
// --> 5
```

```
var anotherNumber = new Number(5)
```

```
// --> 5
```

```
var pi = 3.14
```

```
// --> 3.14
```

yes, they're all numbers

```
typeof number
```

```
// --> "number"
```

```
typeof anotherNumber
```

```
// --> "number"
```

```
typeof pi
```

```
// --> "number"
```

creating strings

```
var string = "Hello, I'm a string!"  
// --> "Hello, I'm a string!"
```

```
var aRealString = new String("I'm a REAL string!")  
// --> [I, ',m, ,a, ,R,E,A,L, ,s,t,r,i,n,g,!]  
aRealString  
// --> "I'm a REAL string!"
```

```
var anotherString = "Hello, I'm \  
a string with \  
linebreaks!"  
// --> "Hello, I'm  
// a string with  
// linebreaks!"
```

creating regexps

```
var string = "string1 string2 string3"  
string.replace(/string1/, 'String Spectacular')  
// --> "String Spectacular string2 string3"
```

```
var string = "string1 string2 string3"  
string.replace(new RegExp("string1"), 'String Spectacular')  
// --> "String Spectacular string2 string3"
```

booleans - dead simple

```
var booleanVar = false
```

```
// --> false
```

```
var anotherBooleanVar = true
```

```
// --> true
```

```
if(booleanVar) alert("Test!")
```

```
// --> *nothing*
```

```
if(!booleanVar) alert("Test!")
```

```
// --> [alert] Test!
```

```
var anotherBoolean = iReturnABool()
```

null and undefined are special

```
//// null is case-sensitive
```

```
booleanVar = null
```

```
// --> null
```

```
//// undefined (also case-sensitive)
```

```
booleanVar == undefined
```

```
// --> true [because it was set to null]
```

```
booleanVar === undefined
```

```
// --> false
```

doin' stuff with basic types

```
3 * 5 + 9  
// --> 24
```

```
3 * (5 + 9)  
// --> 42
```

```
"string1 string2 string3".length  
// --> 23
```

```
var start = "string1 string2 string3".indexOf('string2')  
// --> 8
```

```
"string1 string2 string3".substr(start)  
// --> "string2 string3"
```

creating arrays

```
var emptyArray = []
```

```
// --> []
```

```
emptyArray.length
```

```
// --> 0 [the number, not false]
```

```
var anArray = ['foo', 'bar', 'baz']
```

```
// --> ['foo', 'bar', 'baz']
```

`var array = [] // --> []`

this deserves repeating

creating simple functions

```
function noArgument() {  
    // do stuff  
    alert("I can't argue; I have no arguments")  
}
```

```
function simpleFunction(arg1,arg2,arg3) {  
    // do stuff here  
    return arg1 + arg2 + arg3  
}
```

```
noArgument()  
// --> [alert] I can't argue; I have no arguments
```

```
simpleFunction(1,2,3)  
// --> 6
```

creating simple objects

```
var newObject = {}  
// --> {} (empty object!)
```

```
newObject.foo = "bar"  
// --> "bar"
```

```
newObject.foo  
// --> "bar"
```

```
var anotherObject = {foo:"bar", baz:"bat"}  
// --> {foo:"bar", baz:"bat"}
```

```
anotherObject.baz  
// --> "bat"
```

```
anotherObject.baz = "zort"  
// --> "zort"
```

```
var newObject = {} // --> {}
```

this deserves repeating

if... or else!

```
if(something) doStuff()
```

```
if(something)
    alert("True!")
else
    alert("Not true!")
```

```
if(something == foobar) {
    alert("equals foobar!")
} else if(something == bazbat) {
    alert("equals bazbat!")
} else {
    alert("equals neither!")
}
```

gonna have to make a switch()

```
switch(something) {  
  case "foobar":  
    alert("Oh no! It's a foobar!")  
    break  
  case "barfoo":  
    alert("Barfoo!")  
    break  
  case "fallthru":  
    alert("Falling through...")  
  case "fellthru":  
    alert("fallen through.")  
    break  
  default:  
    alert("Case not found... here's a default")  
}
```

while()

```
var i = 0
while(i < 3) {
    alert(i)
    i++
}
// --> [alert] 0, [alert] 1, [alert] 2
```

do..while() while() while()

```
var i = 0
do {
    alert(i)
    i++
} while(i < 3)
// --> [alert] 0, [alert] 1, [alert] 2
```

for(), our old friend

```
for(var i = 0; i < 3; i++) {  
    alert(i)  
}
```

```
// --> [alert] 0, [alert] 1, [alert] 2
```

for..in(), a new trick!

```
var theObject = {foo:"bar", baz:"bat", narf:"zoit"}  
// --> [Object object]  
  
for(attribute in theObject) {  
    alert(attribute + " = " + theObject[attribute])  
}  
  
// --> [alert] foo = bar, [alert] baz = bat  
//      [alert] narf = zoit
```

strings

strings act like arrays

```
var string = "Hello, I'm a string!"  
// --> "Hello, I'm a string!"
```

```
string[0]
```

```
// --> H
```

```
string[3]
```

```
// --> l
```

```
for(var i = 0; i < string.length; i++) {  
    alert(string[i])  
}
```

```
// --> [alert] H, [alert] e, [alert] l, etc.
```

concatenation & auto type conversion

```
"string1 " + " string2"  
// --> "string1 string2"
```

```
"string " + 5  
// --> "string 5"
```

```
5 + "5"  
// --> "55"
```

```
"string".length  
// --> 6
```

breakin' up is not hard to do

```
"string1 string2 string3".indexOf('string2')  
// --> 8
```

```
"string1 string2 string3".substr(8)  
// --> "string2 string3"
```

```
var string = "string1 string2 string3"  
string.substr(string.indexOf('string2'))  
// --> "string2 string3"
```

```
//// string was not changed:
```

```
string.split(' ')  
// --> ["string1", "string2", "string3"]
```

the great escape. and unescape.

```
"<h3>Here's a headline!</h3>".escape()  
// ! TypeError !  
var escaped = escape("<h3>Here's a headline!</h3>")  
// --> "%3Ch3%3EHere%27s%20a%20headline%21%3C/h3%3E"  
var unescaped = unescape(escaped)  
// --> "<h3>Here's a headline!</h3>"  
  
var URL = 'http://mysite.com/?stuff="Foo bar!"&bar="stuff" '  
var encodedURL = encodeURIComponent(url)  
// --> http://mysite.com/?stuff=%22Foo%20bar!%22&bar=%22stuff%22  
var decodedURL = decodeURI(encodedURL)  
// --> http://mysite.com/?stuff="Foo bar!"&bar="stuff"
```

RegExps

a bit more on regexps

```
var simpleRegExp = /\sfoo/g
var anotherRegExp = new RegExp("/foo", "g")

"/foobar /foof /foobaz".match(simpleRegExp)
// --> ["/foo", "/foo", "/foo"]

"/foobar /foof /foobaz".match(anotherRegExp)
// --> ["/foo", "/foo", "/foo"]

"/foobar /foof /foobaz".match(/zort/)
// --> null

simpleRegExp.test("/foobar /foof /foobaz")
// --> true
```

functions

closures & callbacks

```
function myFunction(arg1,arg2,arg3) {  
    // use argument values  
    alert("I have an argument! " + arg1)  
    // use an object argument  
    alert(arg2.bar)  
    // call an argument as a function  
    arg3()  
}  
  
myFunction("foo", {bar:"baz"}, function(){ alert("Victory!")})  
// outputs...  
//     [alert] I have an argument! foo  
//     [alert] baz  
//     [alert] Victory!
```

}

}

function handles

//// Creating a function handle

```
var myOtherFunction = function() {just  
    alert("Victory!")  
}
```

```
myFunction("foo", {bar:"baz"}, myOtherFunction)
```

the arguments object

```
function myFunction() {  
    alert("I have an argument! " + arguments[0])  
    alert(arguments[1].bar)  
    arguments[2]()  
}
```

```
myFunction("foo", {bar:"baz"}, myOtherFunction)
```

```
// outputs...
```

```
//     [alert] I have an argument! foo
```

```
//     [alert] baz
```

```
//     [alert] Victory!
```

I

objects

JavaScript Object Notation (JSON)

```
var foo = {  
  bar: "baz",  
  zort: "narf",  
  aNumber: 5,  
  anArray: ['banana', 'plaintain', 'ugli fruit'],  
  doStuff: function() {  
    alert("I'm doing stuff!")  
  }  
}  
  
// --> [Object object]
```

using a JSON object

```
foo.zort
```

```
// --> "narf"
```

```
for(i = 0; i < foo.anArray.length; i++) {  
    alert(foo.anArray[i])  
}
```

```
// --> [alert] banana, [alert] plaintain,  
//      [alert] ugly fruit
```

```
foo.doStuff()
```

```
// --> [alert] I'm doing stuff!
```

turning strings into code (more JSON!)

```
var data = "{banana:'yummy', plaintain:'icky'}"
```

```
function handleAjaxResult(ajaxResult) {  
    eval("var resultData = "+ajaxResult)  
    alert(resultData.banana)  
}
```

```
handleAjaxResult(data)  
// --> [alert] yummy
```

don't forget your commas

```
var foo = {  
  bar:"baz",  
  zort:"narf",  
  aNumber:5,  
  anArray:['banana','plaintain','ugli fruit']  
  doStuff:function() {  
    alert("I'm doing stuff!")  
  }  
}
```

// --> ! SyntaxError ! Line 6: missing } after property list

creating an object constructor

```
function anyFunction() {}  
  
var object = new anyFunction()  
// --> [Object object]
```

creating a better object constructor

```
function Foo() {  
    // use this keyword inside function constructors  
    this.bar = "baz"  
    // run arbitrary code on construction  
    alert("I'm a new Foo!")  
}
```

```
var foo = new Foo()  
// --> [Object object] & [alert] I'm a new Foo!
```

```
foo.bar  
// --> "baz"
```

Javascript has a prototype-based inheritance model

using prototype to extend objects

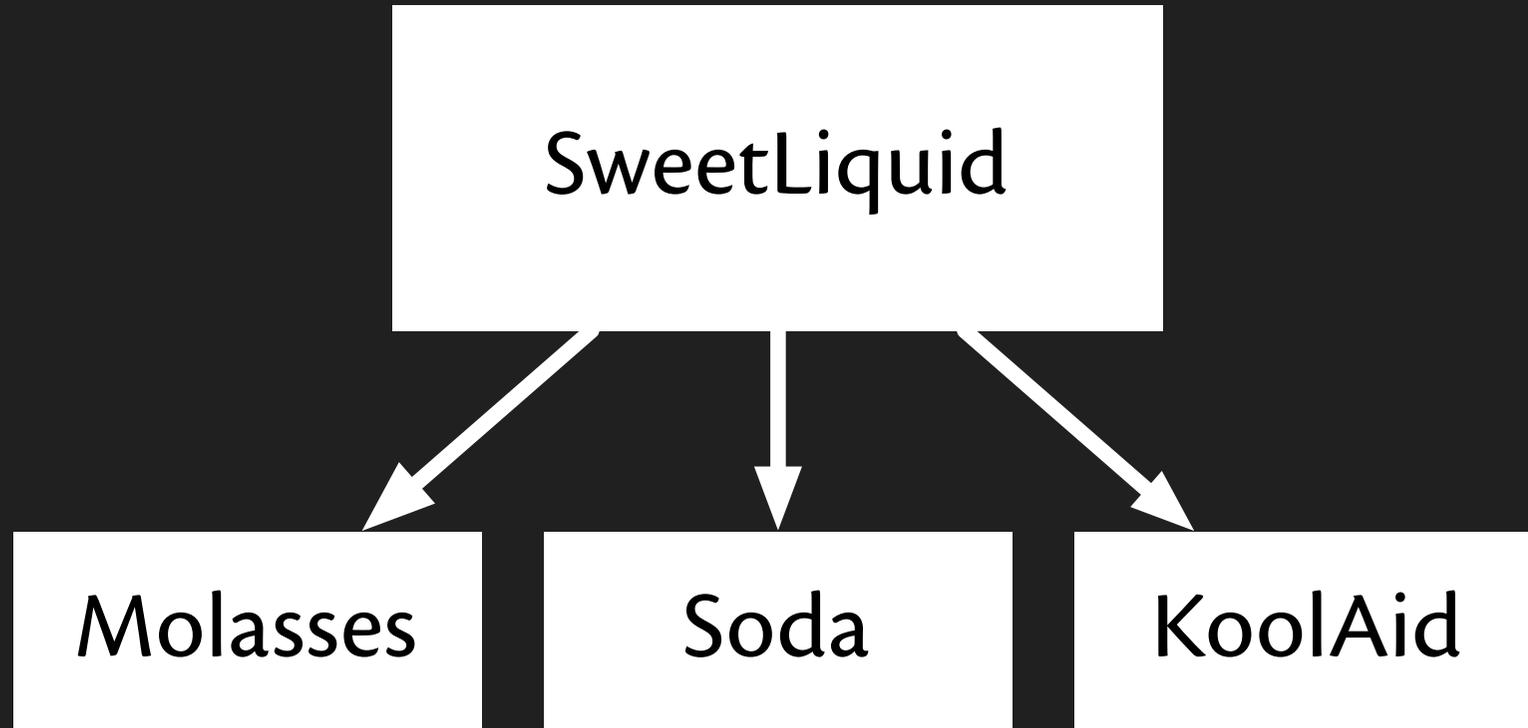
```
function Foo() {  
    this.bar = "baz"  
    alert("I'm a new Foo!")  
}
```

```
var fooInstance = new Foo() // --> [alert] I'm a new Foo!  
fooInstance.bar // --> "baz"
```

```
Foo.prototype.bar = "bat"  
Foo.prototype.fuzz = "boo"  
Foo.prototype.danger = function() { alert("Alert! Alert!" ) }
```

```
fooInstance.danger() // --> [alert] Alert! Alert!  
fooInstance.bar // --> "baz"  
fooInstance.fuzz // --> "boo"
```

creating an object hierarchy



creating the parent object

```
function SweetLiquid() {  
  this.volume = 500  
  this.unit = 'ml'  
  this.drink = function() { this.volume-- }  
  this.sweetnessRating = .5  
}
```

creating the children

```
function Soda() {}  
function Molasses() {  
    this.viscosity = "low"  
}  
  
function KoolAid() {  
    alert("OHH YEAH!")  
    this.mascot = "awesome"  
    this.unit = 'quart'  
    this.volume = '4'  
}
```

creating the relationship

```
Soda.prototype = new SweetLiquid  
Molasses.prototype = new SweetLiquid  
KoolAid.prototype = new SweetLiquid
```

mucking about with your new objects

```
var cola = new Soda // --> [Object object]
cola.volume // --> 500
cola.unit // --> "ml"
```

```
var strawberryKoolAid = new KoolAid
// --> [Object object] ... [alert] "OHH YEAH!"
strawberryKoolAid.mascot // --> "awesome"
strawberryKoolAid.unit // --> "quart"
strawberryKoolAid.volume // --> 4
```

further extending objects

```
KoolAid.prototype.drink = function(amount) {  
  this.volume = this.volume - amount  
  return this.volume  
}
```

```
strawberryKoolAid.drink(1) // --> 3
```

extending specific instances

```
strawberryKoolAid.spill = function(amount) {  
  this.volume = this.volume - (amount * 2)  
  return this.volume  
}
```

```
grapeKoolAid = new KoolAid
```

```
// --> [Object object] ... [alert] "OHH YEAH!"
```

```
strawberryKoolAid.spill(1) // --> 1
```

```
grapeKoolAid.spill(1)
```

```
// ! TypeError ! grapeKoolAid.spill is not a function
```

exceptions

try..catch

```
grapeKoolAid.spill(1)
```

```
// ! TypeError ! grapeKoolAid.spill is not a function
```

```
try {
```

```
    grapeKoolAid.spill(1)
```

```
} catch (e) {
```

```
    alert("Oops! An error of type "+e.name + " occurred. \  
    The message was: " +e.message)
```

```
}
```

```
// --> [alert] Oops! An error of type TypeError occurred.
```

```
//     The message was: grapeKoolAid.spill is not a function
```

fancier try..catch

```
try {
  grapeKoolAid.spill(1)
} catch (e if e.name == "TypeError") {
  alert("Oops! You seemed to have called the wrong \
  kind of thing")
} catch (e) {
  alert("Another kind of error has occurred")
}
// --> [alert] Oops! You seemed to have called the wrong
//      kind of thing
```

throwing custom exceptions

```
grapeKoolAid.spill = function(amount) {  
  if(amount) {  
    throw "You can't spill grape KoolAid! \  
    That's ridiculous!"  
  }  
}  
grapeKoolAid.spill(2)  
// ! CantSpillError ! You can't spill grape KoolAid!  
// That's ridiculous!
```

throwing custom exceptions

```
grapeKoolAid.spill = function(amount) {  
  if(amount) {  
    throw "You can't spill grape KoolAid! \  
    That's ridiculous!"  
  }  
}  
grapeKoolAid.spill(2)  
// ! CantSpillError ! You can't spill grape KoolAid!  
// That's ridiculous!
```

throwing custom exception objects

```
function CantSpillError(beverage) {  
  this.name = "CantSpillError"  
  this.message = "You can't spill "+beverage+"! \  
  That's ridiculous!"  
}
```

```
grapeKoolAid.spill = function(amount) {  
  if(amount) {  
    throw new CantSpillError("grape KoolAid")  
  }  
}
```

catching custom exceptions

```
try {
    grapeKoolAid.spill(1)
} catch (e) {
    alert("Oops! An error of type "+e.name + " occurred. \
    The message was: " +e.message)
}
// --> [alert] Oops! An error of type CantSpillError occurred.
//     The message was: You can't spill grape KoolAid!
//     That's ridiculous!
```

the browser

“Javascript”

ECMAScript

W3C DOM

Document Object Model

the split-personality language

Javascript the language is
mostly compatible

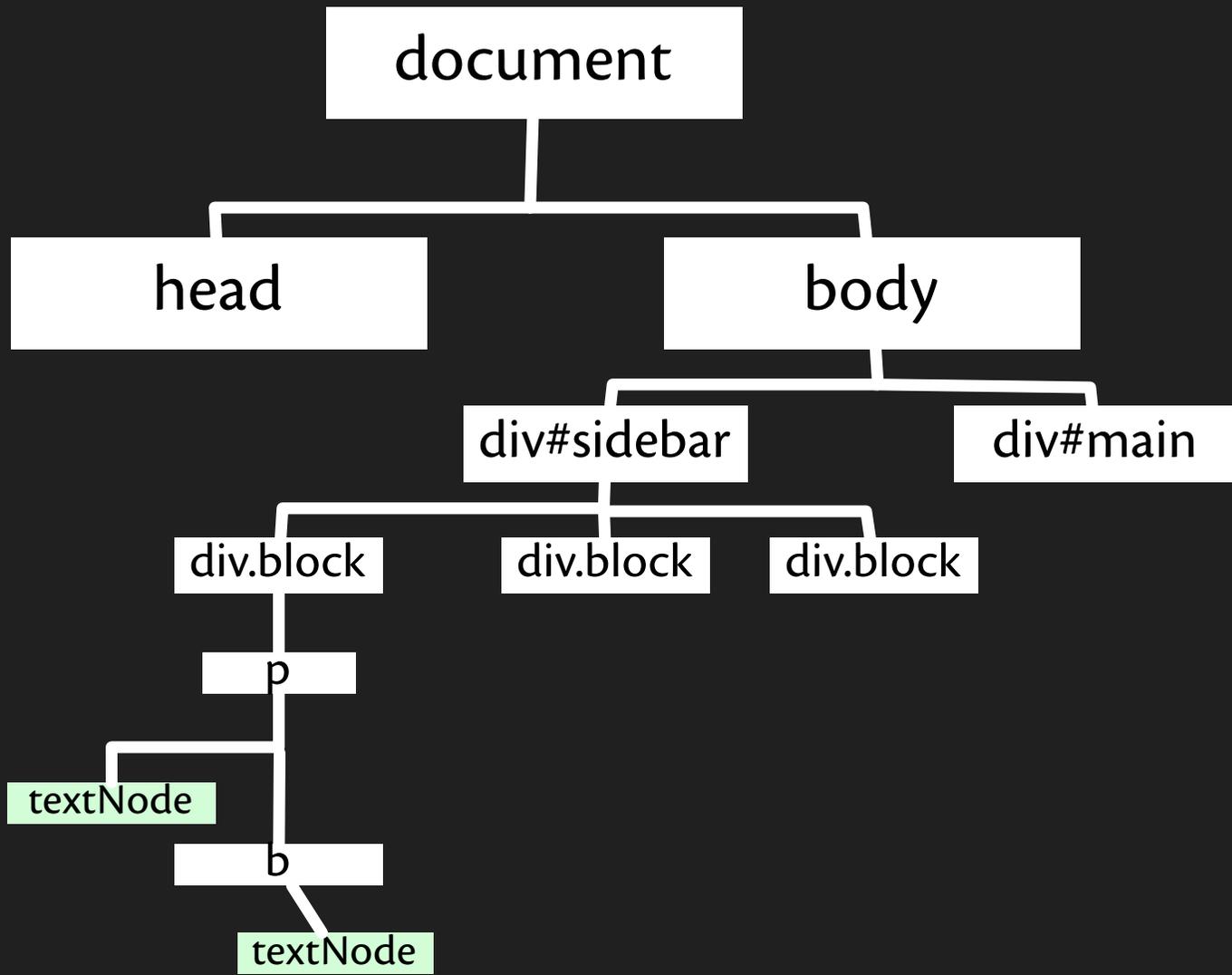
the DOM is the problem *area*

IE is *the problem*, but not as bad as
you'd think

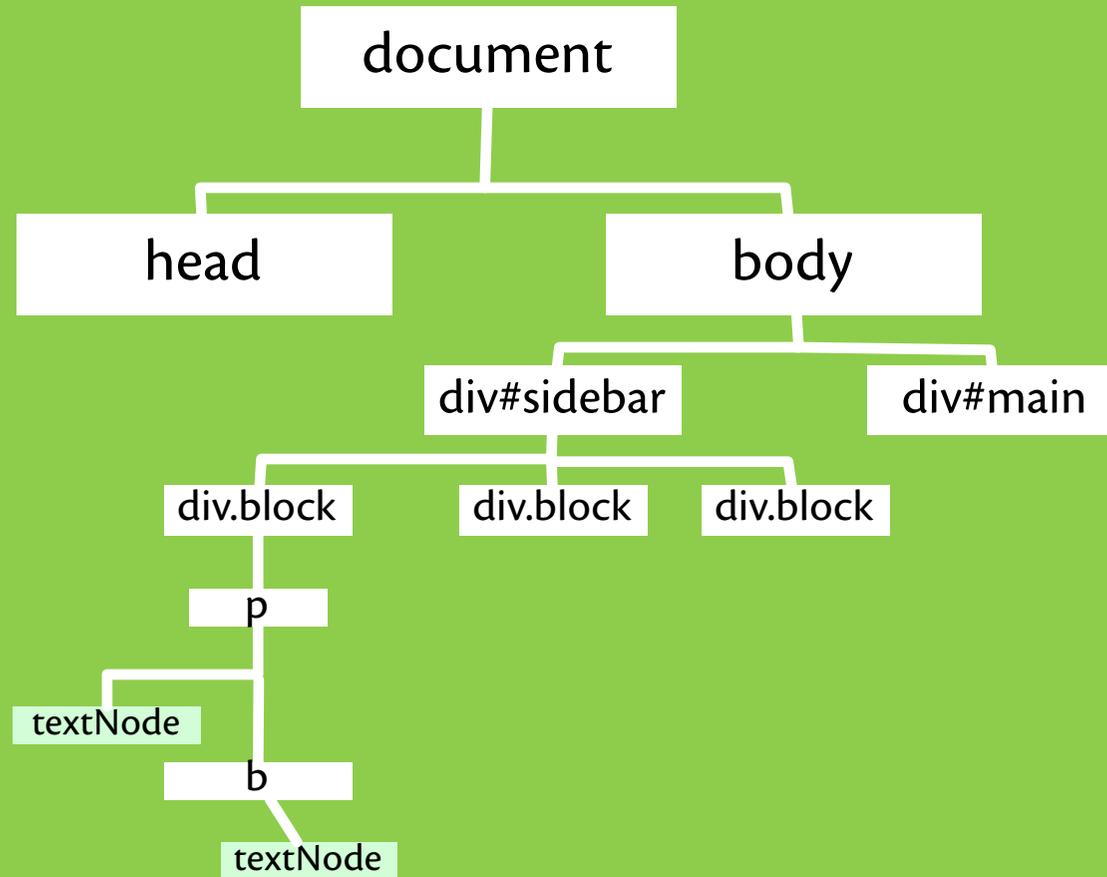
Don't hate the
language, hate the
runtime
environment

the DOM

```
<!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" xml:lang="en" lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>A DOM Tree Grows in Brooklyn</title>
  </head>
  <body>
    <div id="sidebar">
      <div class="block">
        <p>Here's some text. Here's some <b>bold</b> text.</p>
      </div>
      <div class="block">
      </div>
      <div class="block">
      </div>
    </div>
    <div id="main">
      <div class="post">
        <h2>Title</h2>
        <p>
          Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
          sed diam nonumy eirmod tempor invidunt ut labore et dolore
          magna aliquyam erat, sed diam voluptua.
        </p>
        <p>
          Duis autem vel eum iriure dolor in hendrerit in vulputate
          velit esse molestie consequat vel illum dolore eu feugiat.
        </p>
      </div>
    </div>
  </body>
</html>
```



window



Here's some text. Here's some **bold** text.

Title

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat vel illum dolore eu feugiat.

Clear Inspect Options

Console

Debugger

Inspector



```
<html lang="en" xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">
  <head>
  <body>
    <div id="sidebar">
      <div class="block">
        <p>
          <b>bold</b>
        </p>
      </div>
      <div class="block"> </div>
      <div class="block"> </div>
    </div>
    <div id="main">
  </body>
```



Source

Style

Layout

Events

DOM

Done



0.161s



iterating through the DOM tree

```
var children = document.childNodes;
for(var i=0; i < children.length; i++) {
    alert(children[i])
}
// --> [alert] [Object DocumentType]
// --> [alert] [Object HTMLHtmlElement]
```

```
var children = document.body.childNodes;
for(var i=0; i < children.length; i++) {
    alert(children[i])
}
// --> [alert] [Object Text]
// --> [alert] [Object HTMLDivElement]
// --> [alert] [Object Text]
// etc
```

DOM element attributes

```
<div id="main" class="box">  
  <div class="post">  
    <h2>Title</h2>  
    <p>  
      Lorem ipsum dolor sit amet, consetetur sadipscing elitr,  
      sed diam nonumy eirmod tempor invidunt ut labore et dolore  
      magna aliquyam erat, sed diam voluptua.  
    </p>  
  </div>  
</div>
```

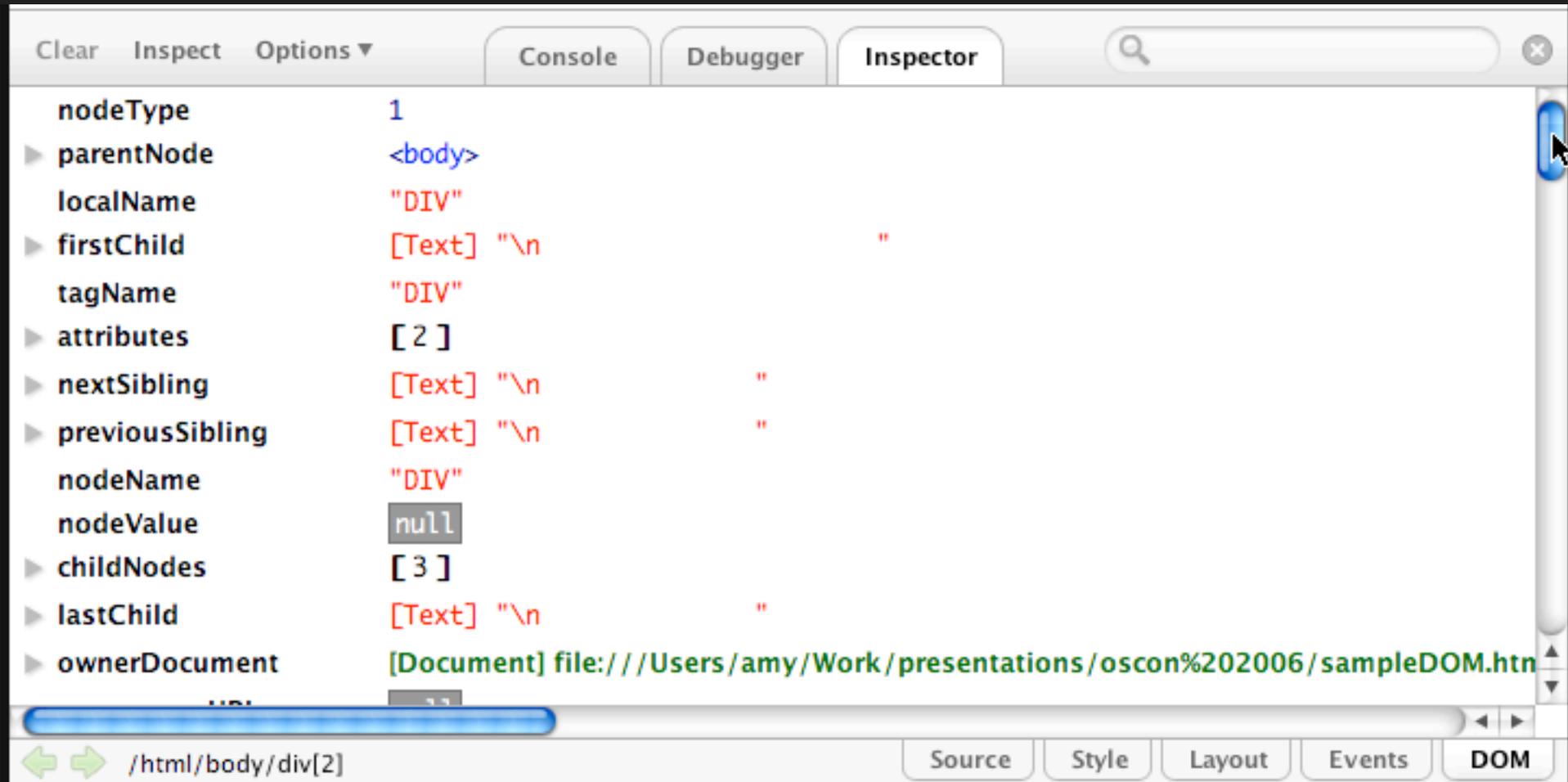
DOM element attributes



The image shows a browser's DOM Inspector window. At the top, there are tabs for 'Clear', 'Inspect', 'Options', 'Console', 'Debugger', and 'Inspector'. A search bar is located to the right of the 'Inspector' tab. The main area displays a tree view of the document's DOM. The root element is `<html lang="en" xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">`. It has two children: `<head>` and `<body>`. The `<body>` element has three children: `<div id="sidebar">`, `<div class="box" id="main">` (which is highlighted in blue), and `<script> var children = document.childNodes; for...</script>`. The `<div class="box" id="main">` element has one child: `<div class="post">`. This `<div class="post">` element contains three children: `<h2>Title</h2>`, `<p> Lorem ipsum dolor sit amet, consetetur s...</p>`, and `<p> Duis autem vel eum iriure dolor in hendr...</p>`. The bottom of the window shows a breadcrumb path `/html/body/div[2]` and a set of tabs for 'Source', 'Style', 'Layout', 'Events', and 'DOM'.

```
<html lang="en" xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">
  <head>
  <body>
    <div id="sidebar">
    <div class="box" id="main">
      <div class="post">
        <h2>Title</h2>
        <p> Lorem ipsum dolor sit amet, consetetur s...</p>
        <p> Duis autem vel eum iriure dolor in hendr...</p>
      </div>
    </div>
    <script> var children = document.childNodes; for...</script>
  </body>
</html>
```

DOM element attributes



The image shows a screenshot of a web browser's DOM Inspector. The interface includes a search bar at the top right, tabs for 'Console', 'Debugger', and 'Inspector', and a search icon. The main area displays the DOM tree for a selected element, showing its properties and values. The selected element is a DIV with the following attributes:

- nodeType**: 1
- parentNode**: <body>
- localName**: "DIV"
- firstChild**: [Text] "\n"
- tagName**: "DIV"
- attributes**: [2]
- nextSibling**: [Text] "\n"
- previousSibling**: [Text] "\n"
- nodeName**: "DIV"
- nodeValue**: null
- childNodes**: [3]
- lastChild**: [Text] "\n"
- ownerDocument**: [Document] file:///Users/amy/Work/presentations/oscon%202006/sampleDOM.htm

The bottom of the window shows a breadcrumb path: /html/body/div[2]. At the bottom right, there are tabs for 'Source', 'Style', 'Layout', 'Events', and 'DOM', with 'DOM' currently selected.

DOM element attributes

The image shows a browser's DOM Inspector window. At the top, there are tabs for 'Clear', 'Inspect', and 'Options'. Below these are three main sections: 'Console', 'Debugger', and 'Inspector'. The 'Inspector' section is active and displays a list of attributes and their values for a selected element. The attributes listed are: namespaceURI (null), prefix (null), id ('main'), title (''), lang (''), dir (''), className ('box'), align (''), offsetTop (52), offsetLeft (8), offsetWidth (700), offsetHeight (124), offsetParent (<body>), and innerHTML ('\\n'). The innerHTML value is expanded to show the HTML content: '<div class="post">\\n'. At the bottom of the window, there are navigation arrows and a path: '/html/body/div[2]'. On the right side, there are tabs for 'Source', 'Style', 'Layout', 'Events', and 'DOM', with 'DOM' being the active tab.

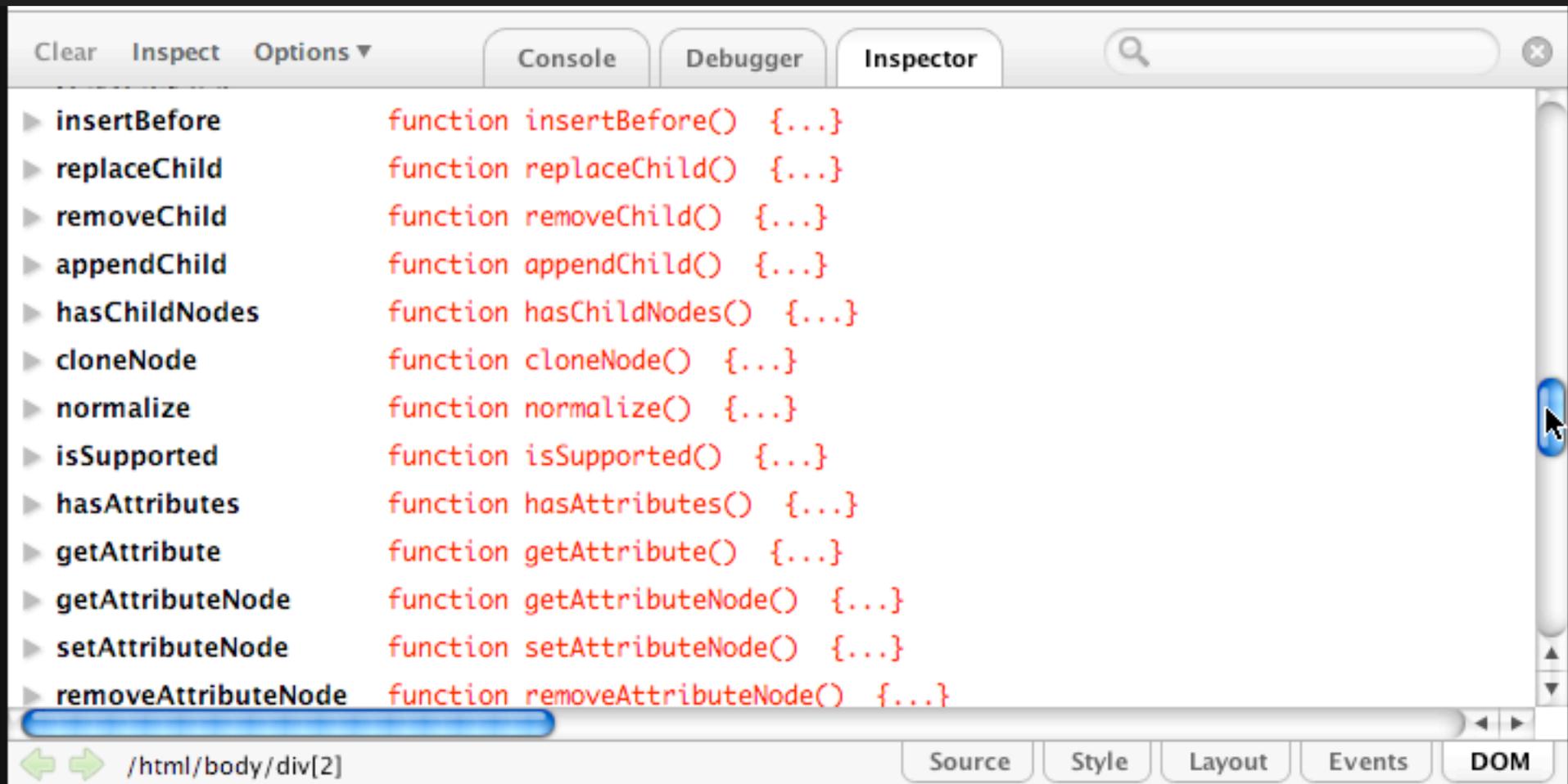
Attribute	Value
namespaceURI	null
prefix	null
id	"main"
title	""
lang	""
dir	""
className	"box"
align	""
offsetTop	52
offsetLeft	8
offsetWidth	700
offsetHeight	124
offsetParent	<body>
innerHTML	"\\n"

innerHTML: <div class="post">\\n

Path: /html/body/div[2]

Inspector Tabs: Source, Style, Layout, Events, DOM

DOM element attributes & methods



The image shows a screenshot of a web browser's developer console, specifically the 'Inspector' tab. The console displays a list of DOM methods and their corresponding function signatures. The methods are listed on the left, and their signatures are shown on the right. The methods are: insertBefore, replaceChild, removeChild, appendChild, hasChildNodes, cloneNode, normalize, isSupported, hasAttributes, getAttribute, getAttributeNode, setAttributeNode, and removeAttributeNode. Each method is followed by a function signature in red text, such as 'function insertBefore() {...}'. The console also shows the breadcrumb path '/html/body/div[2]' and navigation buttons for Source, Style, Layout, Events, and DOM.

Method	Signature
insertBefore	function insertBefore() {...}
replaceChild	function replaceChild() {...}
removeChild	function removeChild() {...}
appendChild	function appendChild() {...}
hasChildNodes	function hasChildNodes() {...}
cloneNode	function cloneNode() {...}
normalize	function normalize() {...}
isSupported	function isSupported() {...}
hasAttributes	function hasAttributes() {...}
getAttribute	function getAttribute() {...}
getAttributeNode	function getAttributeNode() {...}
setAttributeNode	function setAttributeNode() {...}
removeAttributeNode	function removeAttributeNode() {...}

DOM element attributes & methods



The image shows a screenshot of a web browser's DOM Inspector. The interface includes a search bar at the top right, tabs for 'Console', 'Debugger', and 'Inspector', and a search icon. The main area displays a list of DOM methods for an element, with a blue highlight on the 'removeEventListener' method. The methods are listed in two groups: 'getElementsByTagName' and 'getElementsByTagNameNS'. The 'removeEventListener' method is highlighted with a blue bar. The breadcrumb at the bottom shows the path '/html/body/div[2]' and tabs for 'Source', 'Style', 'Layout', 'Events', and 'DOM'.

```
Clear Inspect Options ▾ Console Debugger Inspector
```

getElementsByTagName

- ▶ `getAttributeNS` function `getAttributeNS()` {...}
- ▶ `setAttributeNS` function `setAttributeNS()` {...}
- ▶ `removeAttributeNS` function `removeAttributeNS()` {...}
- ▶ `getAttributeNodeNS` function `getAttributeNodeNS()` {...}
- ▶ `setAttributeNodeNS` function `setAttributeNodeNS()` {...}
- ▶ function `getElementsByTagNameNS()` {...}

getElementsByTagNameNS

- ▶ `hasAttribute` function `hasAttribute()` {...}
- ▶ `hasAttributeNS` function `hasAttributeNS()` {...}
- ▶ `blur` function `blur()` {...}
- ▶ `focus` function `focus()` {...}
- ▶ `removeEventListener` function `removeEventListener()` {...}

← → /html/body/div[2] Source Style Layout Events **DOM**

node.childNodes

The screenshot shows a browser's DOM Inspector window. The top bar includes 'Clear', 'Inspect', and 'Options' menus, along with tabs for 'Console', 'Debugger', and 'Inspector'. A search bar is located on the right. The main area displays the DOM tree with 'childNodes' expanded, showing an array of 3 elements. The first element is selected, and its properties and value are shown on the right. The value is a string containing a newline character and a closing tag for a 'post' class. The bottom bar shows the current path as '/html/body/div[2]' and tabs for 'Source', 'Style', 'Layout', 'Events', and 'DOM'.

```
childNodes [3]
  0 [Text] "\n"
  1 <div class="post">
    nodeType 1
    parentNode <div id="main" class="box">
    localName "DIV"
    firstChild [Text] "\n"
    tagName "DIV"
    attributes [1]
    nextSibling [Text] "\n"
    previousSibling [Text] "\n"
    nodeName "DIV"
    nodeValue null
```

/html/body/div[2] Source Style Layout Events DOM

node.innerHTML

The screenshot shows a web browser's developer console with the 'Inspector' tab active. A mouse cursor is hovering over the 'innerHTML' property of a selected DOM node. The console displays the following HTML structure:

```
"\n      <div class="post">\n\n        <div class="post">\n          <h2>Title</h2>\n          <p>\n            Lorem ipsum dolor sit amet, consetetur sadipscing el  
sed diam nonumy eirmod tempor invidunt ut labore et dolore  
magna aliquyam erat, sed diam voluptua.\n          </p>\n          <p>\n            Duis autem vel eum iriure dolor in hendrerit in vulp  
velit esse molestie consequat vel illum dolore eu feugiat.\n          </p>\n        </div>\n      </div>\n    </h2>
```

Below the HTML code, the console shows the following properties:

```
scrollTop 0  
scrollLeft 0
```

At the bottom of the console, the breadcrumb path is shown as `/html/body/div[2]`, and the 'DOM' tab is selected among other options like 'Source', 'Style', 'Layout', and 'Events'.

finding DOM elements by id

```
▶ <div id="sidebar">
▼ <div class="box" id="main">
  ▼ <div class="post">
    <h2>Title</h2>
    <p> Lorem ipsum dolor sit amet, consetetur s...</p>
    <p> Duis autem vel eum iriure dolor in hendr...</p>
  </div>
</div>
```

```
var node = document.getElementById("main")
// --> [object HTMLDivElement]
```

```
node.id
// --> "id"
```

```
node.tagName
// --> "DIV"
```

```
node.className
// --> "box"
```

}

finding DOM elements by tagName

```
▼ <body>  
  ▼ <div id="sidebar">  
    ▶ <div class="block">  
      <div class="block"> </div>  
      <div class="block"> </div>  
    </div>
```

```
var parent = document.getElementById("sidebar")
```

```
// --> [object HTMLDivElement]
```

```
var sidebarBlocks = parent.getElementsByTagName("div")
```

```
// --> [ div, div, div ]
```

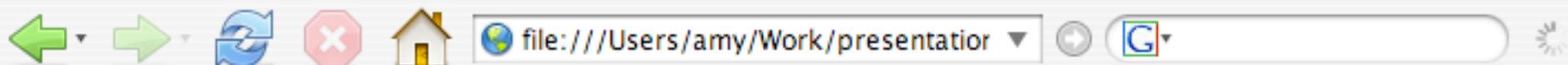
creating new DOM nodes

```
▼ <body>  
  ▼ <div id="sidebar">  
    ▶ <div class="block">  
      <div class="block"> </div>  
      <div class="block"> </div>  
    </div>
```

```
var parent = document.getElementById("sidebar")  
// --> [object HTMLDivElement]
```

```
var sidebarBlocks = parent.getElementsByTagName("div")  
// --> [ div, div, div ]
```

```
for(i = 0; i < sidebarBlocks.length; i++) {  
  var newNode = document.createElement('p')  
  newNode.innerHTML = "I am #" + i  
  sidebarBlocks[i].appendChild(newNode)  
}
```



Here's some text. Here's some **bold** text.

I am #0

I am #1

I am #2

Clear Inspect Options ▾

Console

Debugger

Inspector



```
>>> var parent = document.getElementById("sidebar")
>>> var sidebarBlocks = parent.getElementsByTagName("div")
>>> for(i = 0; i < sidebarBlocks.length; i++) {   var newNode = document.createElement('p'); newNode.
<p>
```

```
>>> |
```

Done

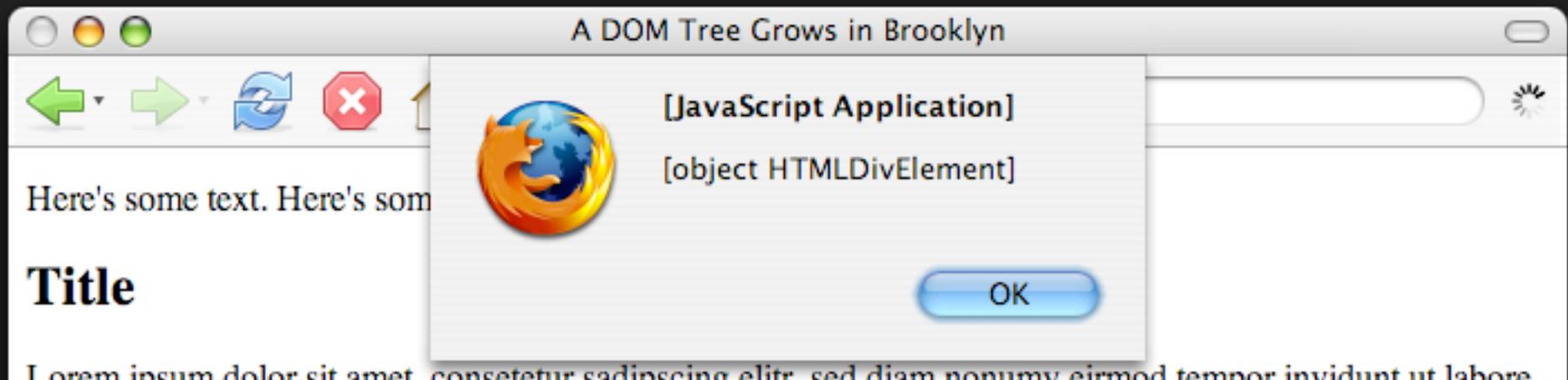


0.174s



embedding javascript

```
<p>
  Duis autem vel eum iriure dolor in hendrerit in vulputate
  velit esse molestie consequat vel illum dolore eu feugiat.
</p>
</div>
</div>
<script>
  alert(document.getElementById("sidebar"))
</script>
</body>
</html>
```



Prototype

```
Object.extend(RealClass, AbstractClass)
Object.extend(RealClass, {
  anotherMethod: function() {}
})
```

```
// before:
```

```
document.getElementById('content')
```

```
// after:
```

```
$('content')
```

```
// <span class="title">title!</span>
```

```
// <span class="title">title 2!</span>
```

```
var titles = $$('span.title')
```

```
"<b>I'm a tag!</b>".stripTags()  
// I'm a tag!
```

```
"<b>I'm a tag!</b>".escapeHTML()  
// &lt;b&gt;I'm a tag!&lt;/b&gt;
```

```
[5,6,7, 'peach', 9].without('peach', 7)  
// [5,6,9]
```

```
var n = 10;  
n.times(function(index){  
    alert(index);  
});
```

```
['one', 'two', 'three'].each (
  function(num, index) {
    msg = num+" in position "+index
  }
)
```

```
Event.observe('dropbox', 'mouseover',  
              function() {}, true);
```

```
document.getElementsByClassName(  
  'draggables')
```

```
new Ajax.Updater('ajax_result',  
                '/posts/1/comment');
```

```
new Ajax.Updater('ajax_result',  
    '/posts/1/comment',  
    onLoading: function() {  
        Element.show('spinner')  
    },  
    onComplete: function() {  
        Element.hide('spinner')  
    }  
);
```

Development & Debugging



FireFox

your long-term best bud



FireBug

your new soulmate

Firefox - Rediscover the Web

http://www.mozilla.com/firefox/

mozilla Products Add-Ons Support Developers

Home » Products » Firefox

Firefox[®] 1.5

The award-winning, free Web browser is better than ever. Browse the Web with confidence - Firefox protects you from viruses, spyware and pop-ups. Enjoy improvements to performance, ease of use and privacy. It's easy to import your favorites and settings and get started. Download Firefox now and get the most out of the Web.

 **Download Firefox**
1.5.0.4 for Mac OS X, English (16.0MB)

[System Requirements](#) - [Release Notes](#) - [Other Systems & Languages](#)

Internet Utilities
Remove your home on the web with editors, blogging clients and more.

Search Engines
Internet Utilities
Mozilla Firefox
Firefox Start

Clear Inspect Options Console Inspector

```
The award-winning, free Web browser is better ...</p>
<script type="text/javascript"> <!-- // Configure the Firefox downl...</script>
<div class="home-download">
<div class="download-other">
```

Done /html/body/div/div/p Source Style Layout Events DOM 0.586s

inspecting DOM elements: source

Firefox - Rediscover the Web

http://www.mozilla.com/firefox/

mozilla Products Add-Ons Support Developers

Home » Products » Firefox

Firefox[®] 1.5

The award-winning, free Web browser is better than ever. Browse the Web with confidence - Firefox protects you from viruses, spyware and pop-ups. Enjoy improvements to performance, ease of use and privacy. It's easy to import your favorites and settings and get started. Download Firefox now and get the most out of the Web.

 **Download Firefox**
1.5.0.4 for Mac OS X, English (16.0MB)

[System Requirements](#) - [Release Notes](#) - [Other Systems & Languages](#)

Clear **Inspect** Options ▾ Console Inspector

▼ margin

margin-top	0px	#main-feature p.product-i...	content.css (line 109)
margin-right	0px	#main-feature p.product-i...	content.css (line 109)
margin-bottom	10px	#main-feature p.product-i...	content.css (line 109)
margin-left	0px	#main-feature p.product-i...	content.css (line 109)

▼ font

color	#414D66	#main-feature p.product-i...	content.css (line 109)
-------	---------	------------------------------	------------------------

▼ text

line-height	18.8px	#main-feature p.product-i...	content.css (line 109)
-------------	--------	------------------------------	------------------------

Done /html/body/div/div/p Source Style Layout Events DOM 0.586s

inspecting DOM elements: style

Firefox - Rediscover the Web

mozilla Products Add-Ons Support Developers

Home » Products » Firefox

Firefox[®] 1.5

The award-winning, free Web browser is better than ever. Browse the Web with confidence - Firefox protects you from viruses, spyware and pop-ups. Enjoy improvements to performance, ease of use and privacy. It's easy to import your favorites and settings and get started. Download Firefox now and get the most out of the Web.

 **Download Firefox**
1.5.0.4 for Mac OS X, English (16.0MB)

[System Requirements](#) - [Release Notes](#) - [Other Systems & Languages](#)

Clear **Inspect** Options ▾ Console Inspector

mousemove	target = , clientX = 284, clientY = 307
mousemove	target = , clientX = 284, clientY = 306
mousemove	target = , clientX = 283, clientY = 306
mousemove	target = , clientX = 283, clientY = 304
mousemove	target = , clientX = 283, clientY = 303
mousemove	target = , clientX = 282, clientY = 303
mousemove	target = , clientX = 282, clientY = 302
mousemove	target = , clientX = 282, clientY = 302

Source Style Layout Events DOM

/html/body/div/div/div/a/span

http://www.mozilla.com/products/download.html?product=firefox-1.5.0.4&os=osx&lang=en-US 0.586s

inspecting DOM elements: live events

Firefox - Rediscover the Web

http://www.mozilla.com/firefox/

mozilla Products Add-Ons Support Developers

Home » Products » Firefox

Firefox[®] 1.5

The award-winning, free Web browser is better than ever. Browse the Web with confidence - Firefox protects you from viruses, spyware and pop-ups. Enjoy improvements to performance, ease of use and privacy. It's easy to import your favorites and settings and get started. Download Firefox now and get the most out of the Web.

 **Download Firefox**
1.5.0.4 for Mac OS X, English (16.0MB)

[System Requirements](#) - [Release Notes](#) - [Other Systems & Languages](#)



Clear Inspect **Options** Console Inspector

- Show JavaScript Errors
- Show JavaScript Warnings
- Show CSS Errors
- Show XML Errors
- Show Errors From Websites
- Show Errors From Chrome
- Show Console Messages
- Show Errors From Other Domains
- Throttle Messages
- Show XMLHttpRequests

! assignment to javascript:try{ S... (line 1)

! assignment to javascript:try{ S... (line 1)

0.586s 2 Errors

console: errors and filtering things to show

More speaking announcementy things

http://www.slash7.com/articles/2006/07/22

Presentation: Getting Started with Ajax on Rails
Presentation: Web 2.0 from the Ground Up
Presentation: When Interface Design Attacks! (part 1)
Presentation: When Interface Design Attacks! (part 2)

RSS **Atom**

[RSS feed for this post](#) [trackback uri](#)

Your name [\(leave url/email »\)](#)

Your message

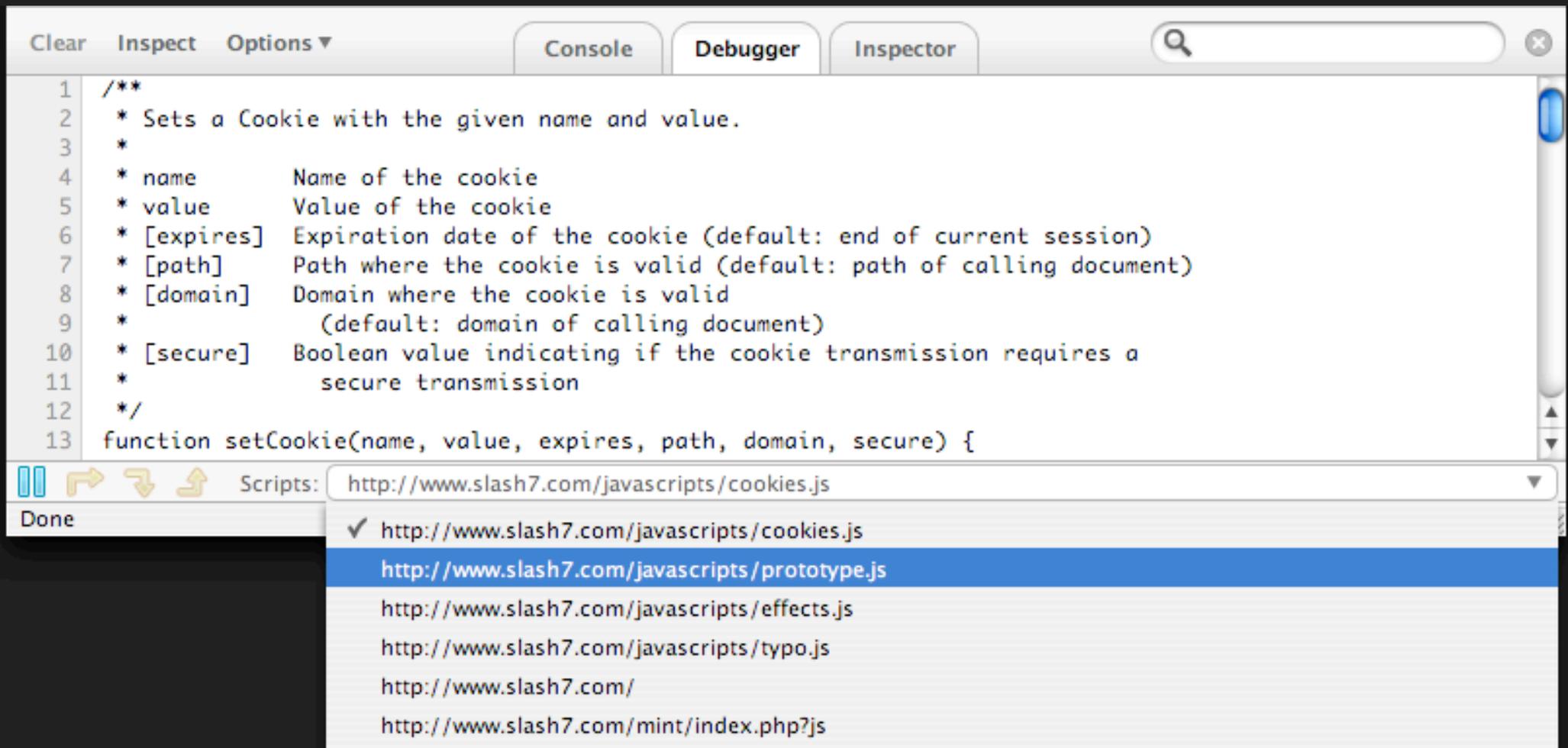
Clear Inspect Options ▾ Console Inspector

POST http://www.slash7.com/articles/comment/10619

Post Response Headers

```
<li id="comment-10702" style="display:none" onmouseover="if (getCookie('is_admin') == 'yes') { Element.s
  <div id="admin_comment_10702" style="display: none"><a class="admintools" href="#" onclick="if (confirm
  <a name="comment-10702"></a>
  <a class="admintools" href="#" onclick="if (confirm
  <a name="comment-10702"></a>
  <img alt="Avatar" class="gravatar" src="http://www.gravatar.com/avatar.php?gravatar_id=d41d8cd98f00b204
  <cite><strong>Amy Hoy</strong> </cite> said 1 day later:<br />
  <p>Posting comments to your own blog in a screenshot? The ultimate in self-promotional wankery</p>
</li>
```

console: logging / inspecting AJAX requests



the step-thru debugger: new in FireBug .4

breakpoints

step-through debugging

flexible console logging

built-in profiling & testing functionality

live DOM inspector / editor

~~the ability to send email~~

Javascript Shell

no snazzy logo

www.squarefree.com/shell/shell.html

Other Tools & Libraries

JSUnit

MochiKit

jQuery

moo.ajax

TrimPath

moo.fx

Behavior & event:Selectors

Scriptaculous (fx)

moo.fx

Links for You

<http://www.slash7.com/>

look for the list of presentations under the
“Goodies” sidebar