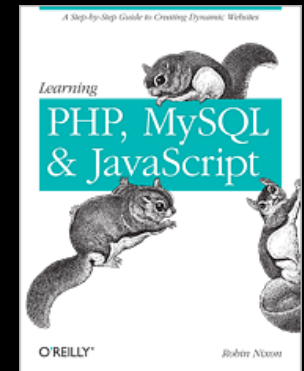


# Exploring JavaScript

Chapter 14

Dr. Charles Severance

To be used in association with the book:  
PHP, MySQL, and JavaScript by Robin Nixon



open.michigan

Unless otherwise noted, the content of this course material is licensed under a Creative Commons Attribution 3.0 License.

<http://creativecommons.org/licenses/by/3.0/>.

Copyright 2011, Charles Severance



# About JavaScript

- Introduced in Netscape in 1995
- Developed by Brandon Eich (now the CTO of Mozilla Foundation)
- Named to make use of Java market buzz
- Standardized today as ECMAScript



[http://en.wikipedia.org/wiki/Brendan\\_Eich](http://en.wikipedia.org/wiki/Brendan_Eich)

```
<html>
<head>
<title>Hello World</title>
</head>
<body>
<p>One Paragraph</p>
<script type="text/javascript">
  document.write("<p>Hello World</p>")
</script>
<noscript>
Your browser doesn't support or has disabled
JavaScript.
</noscript>
<p>Second Paragraph</p>
</body>
</html>
```

One Paragraph

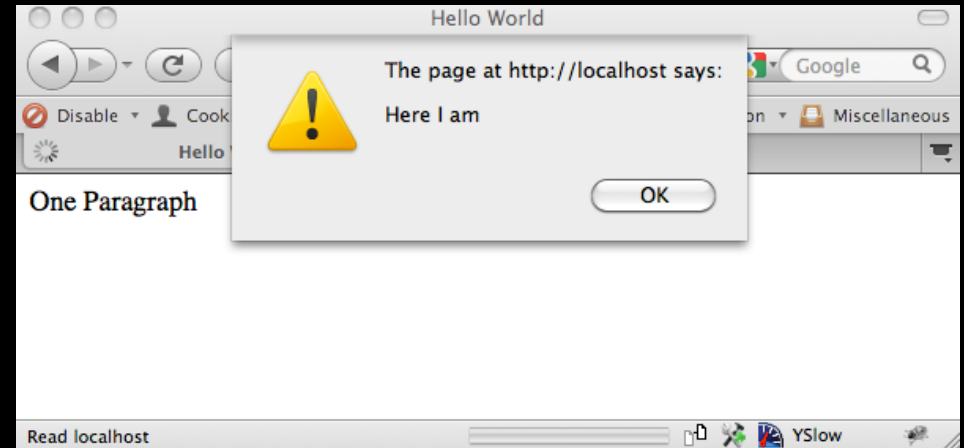
Hello World

Second Paragraph

# Low-Level Print

- When in doubt, you can always add an `alert()` to your JavaScript
- The `alert()` function takes a string as a parameter and pauses the JavaScript execution until you press "OK"

```
<html>
<head>
<title>Hello World</title>
</head>
<body>
<p>One Paragraph</p>
<script type="text/javascript">
  alert("Here I am");
  document.write("<p>Hello World</p>")
</script>
<noscript>
Your browser doesn't support or has disabled
JavaScript.
</noscript>
<p>Second Paragraph</p>
</body>
</html>
```

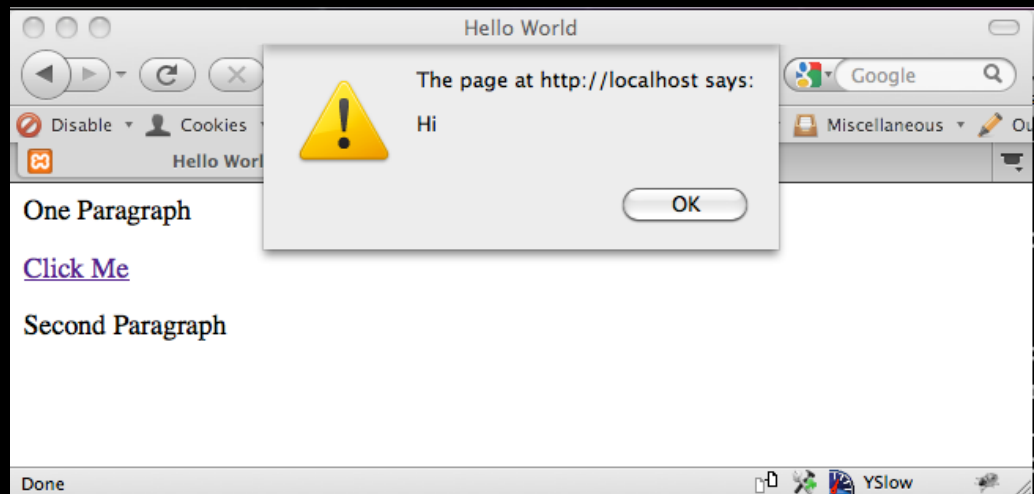


# Including JavaScript

- Three Patterns
  - As part of an event in an HTML tag
  - Inline within the document
  - From a downloaded file

```
<html>
<head>
<title>Hello World</title>
</head>
<body>
<p>One Paragraph</p>
<p><a href="js-01.htm"
  onclick="alert('Hi'); return false;">Click Me</a></p>
<p>Third Paragraph</p>
</body>
</html>
```

JavaScript on a tag



```
<html>
<head>
<title>Hello World</title>
</head>
<body>
<p>One Paragraph</p>
<script type="text/javascript">
  <!--
  document.write("<p>Hello World</p>")
  // -->
</script>
<p>Second Paragraph</p>
</body>
</html>
```

One Paragraph

Hello World

Second Paragraph

JavaScript inline that  
will pass validation

```
<html>
<head>
<title>Hello World</title>
</head>
<body>
<p>One Paragraph</p>
<script type="text/javascript" src="script.js">
</script>
<p>Third Paragraph</p>
</body>
</html>
```

One Paragraph

Hello World

Second Paragraph

JavaScript in a separate  
file

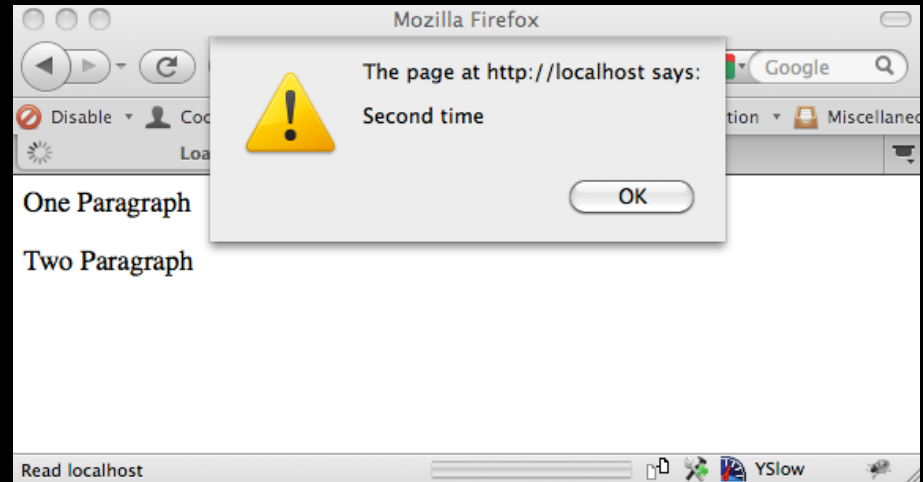
**script.js:**

```
document.write("<p>Hello World</p>");
```

# Syntax Errors

- Like any language, we can make syntax errors
- By default, browsers silently eat any kind of JavaScript error
- But the code stops running in that file or script section and

```
<p>One Paragraph</p>
<script type="text/javascript">
  alert("I am broken");
  alert("I am good");
</script>
<p>Two Paragraph</p>
<script type="text/javascript">
  alert("Second time");
</script>
<p>Three Paragraph</p>
```



# Seeing the Error

- Since the end-user really cannot take any action to fix the JavaScript coming as part of a web page, the browser eats the errors.
- As a developer, we need to look for the errors - sometimes it takes a minute to even remember to check for a JS error

Web Inspector — http://localhost/si572/php-14/js-05.htm

Elements Resources Network Scripts Search Console

All Errors Warnings Logs

**SyntaxError: Unexpected EOF** js-05.htm:3

>

Web Inspector navigation icons: back, forward, refresh, home, search, zoom in, zoom out, print, close.

Mozilla Firefox

http://localhost/si572/php-14/js-05.htm

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source

http://localhost...hp-14/js-05.htm

One Paragraph

Two Paragraph

Three Paragraph

Console HTML CSS Script DOM Net YSlow

Clear Persist Profile All Errors Warnings Info Debug Info

**unterminated string literal** alert("I am broken'); js-05.htm (line 3)

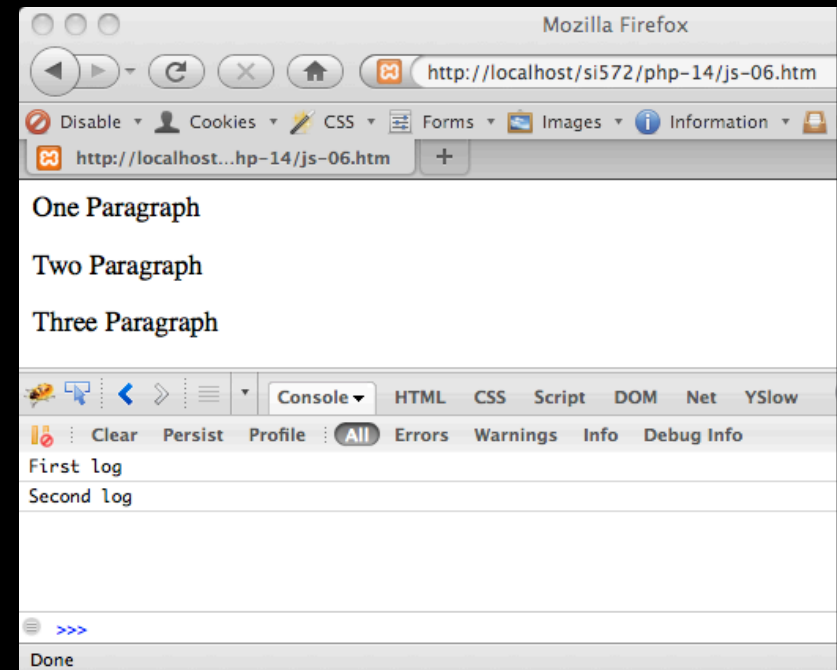
Done YSlow 3.175s 1 Error

Firefox browser interface showing navigation, developer tools, and status bar.

# Console Logging

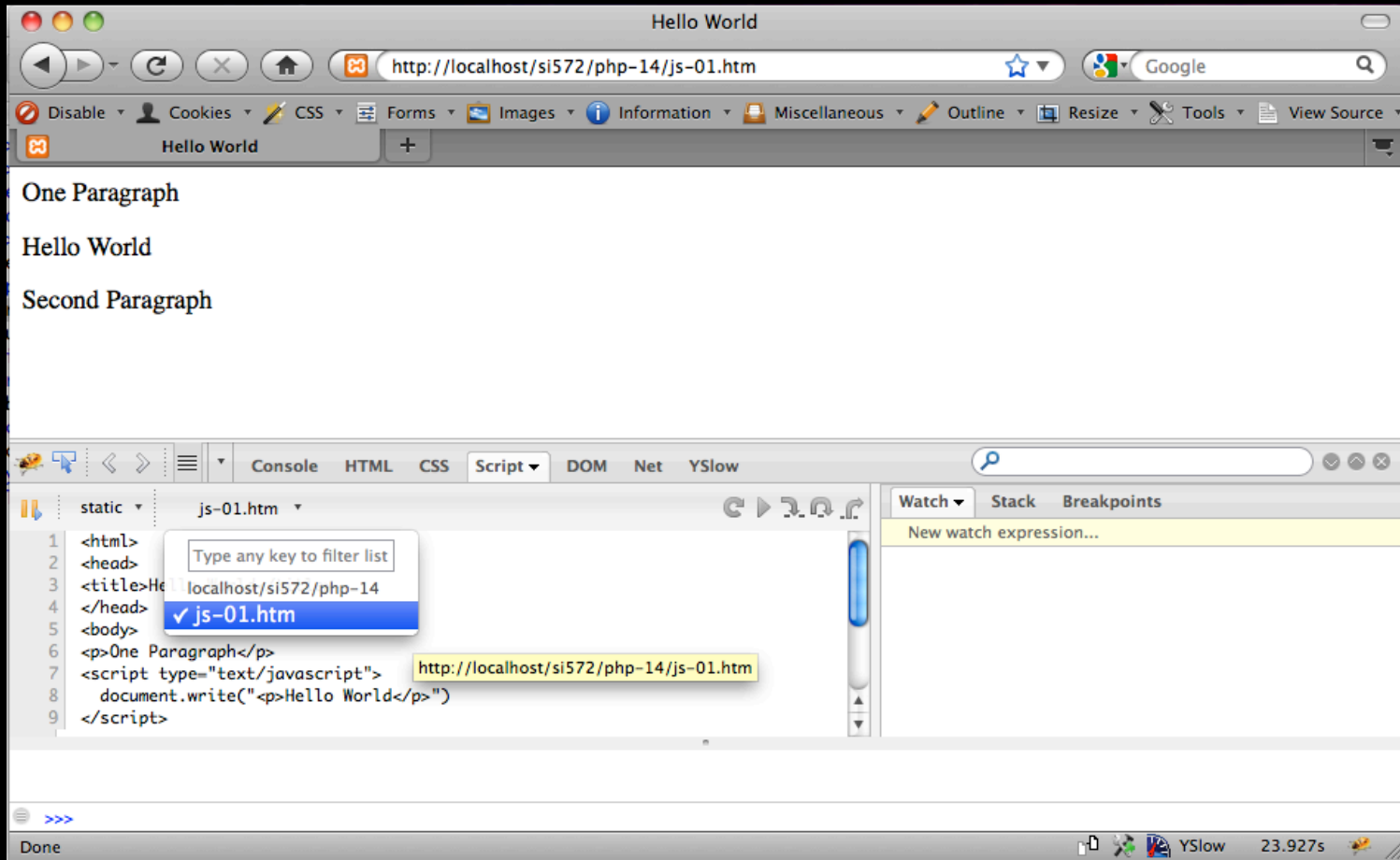
- Debugging using `alert()` can get tiring - sometimes you want to record what happens in case something goes wrong
- <http://getfirebug.com/logging>
- `console.log("String")` - and many more functions
- Note: Requires recent browsers

```
<p>One Paragraph</p>
<script type="text/javascript">
  console.log("First log");
</script>
<p>Two Paragraph</p>
<script type="text/javascript">
  console.log("Second log");
</script>
<p>Three Paragraph</p>
```



# Using the Debugger

- Add the FireBug FireFox Extension
- Enable FireBug, reload
- Set Breakpoint, and reload again



The image shows a web browser window with the title "Hello World". The address bar displays the URL `http://localhost/si572/php-14/js-01.htm`. The browser's menu bar includes options like "Disable", "Cookies", "CSS", "Forms", "Images", "Information", "Miscellaneous", "Outline", "Resize", "Tools", and "View Source". The page content consists of three paragraphs: "One Paragraph", "Hello World", and "Second Paragraph".

Below the page content is the browser's developer tools interface. The "Script" tab is active, showing the source code of the file `static/js-01.htm`. The code is as follows:

```
1 <html>
2 <head>
3 <title>Hello World</title>
4 </head>
5 <body>
6 <p>One Paragraph</p>
7 <script type="text/javascript">
8   document.write("<p>Hello World</p>")
9 </script>
```

A red dot on line 8 indicates a script error. The "Watch" panel on the right is empty, showing "New watch expression...". The status bar at the bottom of the developer tools shows "Done" and a performance indicator of "YSlow 23.927s".

The screenshot shows a web browser window titled "Hello World" with the address bar displaying "http://localhost/si572/php-14/js-01.htm". The page content is "One Paragraph". The browser's developer tools are open, showing the Script panel with the following code:

```
4 </head>
5 <body>
6 <p>One Paragraph</p>
7 <script type="text/javascript">
8   document.write("<p>Hello World</p>")
9 </script>
10 <noscript>
11 Your browser doesn't support or has disabled JavaScript.
12 </noscript>
```

The Watch panel on the right shows the following state:

- new watch expression...
- this: Window js-01.htm
- Components: nsXPCComponents { interfaces=nsXPCComponents\_I, interfacesByID=nsXPCComponer
- applicationCache: 0 items in offline cache
- closed: false
- console: Object { firebug="1.7.3" }

The status bar at the bottom indicates "Read localhost" and "YSlow".

The image shows a web browser window titled "Hello World" with the address bar displaying "http://localhost/si572/php-14/js-01.htm". The page content consists of three paragraphs: "One Paragraph", "Hello World", and "Second Paragraph". The browser's developer tools are open at the bottom, showing the "Script" tab with the following code:

```
1 <html>
2 <head>
3 <title>Hello World</title>
4 </head>
5 <body>
6 <p>One Paragraph</p>
7 <script type="text/javascript">
8   document.write("<p>Hello World</p>")
9 </script>
```

The status bar at the bottom indicates "Done" and "YSlow 36.82s".

# JavaScript Language

# Comments in JavaScript = Awesome

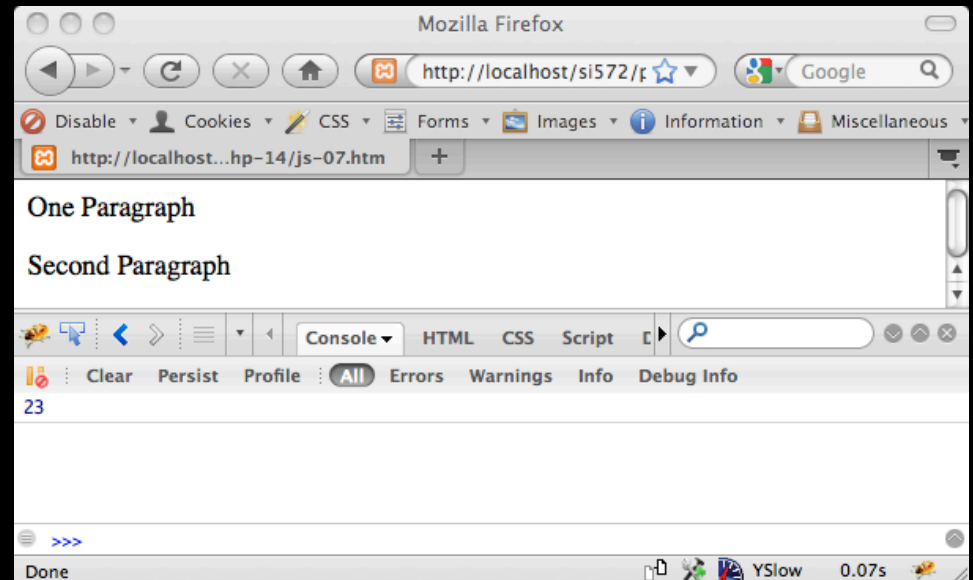
```
// This is a comment
```

```
/* This is a section of  
multiline comments that will  
not be interpreted */
```

# Statements

- White space and newlines do not matter
- Statements end with semicolon ;
- There are cases where you can the semicolon off - but don't bother exploiting the feature - just add semicolons like C, Java, PHP, C++, etc..

```
<p>One Paragraph</p>
<script type="text/javascript">
  x = 3 +
    5 * 4; console.log(
x);
</script>
<p>Second Paragraph</p>
```



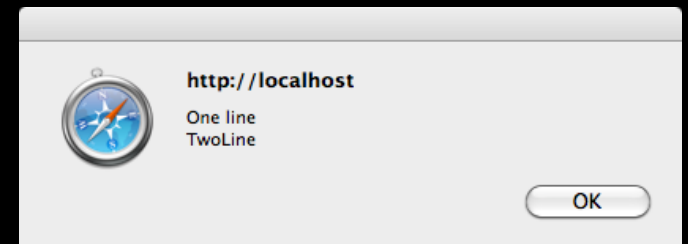
# Variable Names

- Valid Characters: a-z, A-Z, 0-9, \_ and \$
- Must not start with a number
- Names are case sensitive
- Starting with a dollar sign is considered "tacky"

# String Constants

- Double or Single Quotes - Single quotes are used typically in JavaScript and we let HTML use double quotes to keep out minds a little sane.
- **Escaping** - done using the back-slash character

```
<script type="text/javascript">  
alert('One line\nTwoLine');  
</script>
```



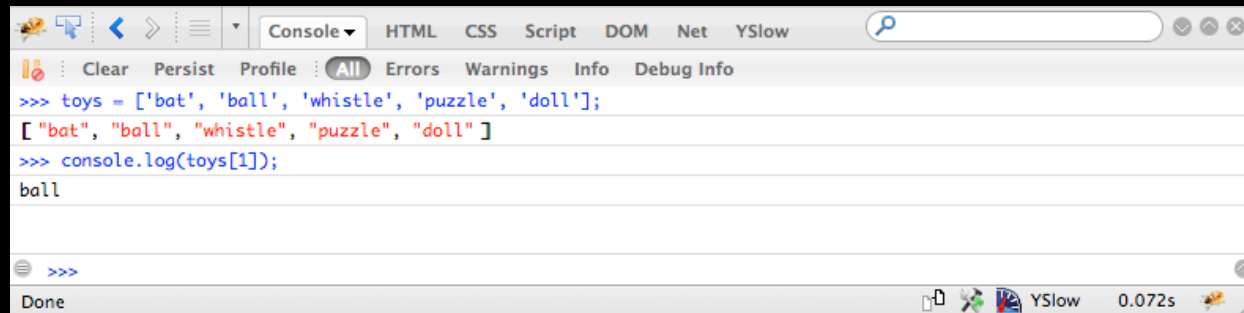
# Numeric Constants

- As you would expect

# JavaScript Arrays

- Arrays are zero-based - more like Python lists

```
toys = ['bat', 'ball', 'whistle', 'puzzle', 'doll'];  
console.log(toys[1]);
```



```
Console | HTML | CSS | Script | DOM | Net | YSlow  
Clear | Persist | Profile | All | Errors | Warnings | Info | Debug Info  
>>> toys = ['bat', 'ball', 'whistle', 'puzzle', 'doll'];  
[ "bat", "ball", "whistle", "puzzle", "doll" ]  
>>> console.log(toys[1]);  
ball  
>>>  
Done | YSlow | 0.072s
```

# Operators

*Table 14-2. Arithmetic operators*

Operator	Description	Example
+	Addition	$j + 12$
-	Subtraction	$j - 22$
*	Multiplication	$j * 7$
/	Division	$j / 3.14$
%	Modulus (division remainder)	$j \% 6$
++	Increment	$++j$
--	Decrement	$--j$

# Assignment Operators

Table 14-3. Assignment operators

Operator	Example	Equivalent to
=	<code>j = 99</code>	<code>j = 99</code>
+=	<code>j += 2</code>	<code>j = j + 2</code>
+=	<code>j += 'string'</code>	<code>j = j + 'string'</code>
-=	<code>j -= 12</code>	<code>j = j - 12</code>
*=	<code>j *= 2</code>	<code>j = j * 2</code>
/=	<code>j /= 6</code>	<code>j = j / 6</code>
%=	<code>j %= 7</code>	<code>j = j % 7</code>

# Comparison Operators

Table 14-4. Comparison operators

Operator	Description	Example
==	Is <b>equal</b> to	<code>j == 42</code>
!=	Is <b>not equal</b> to	<code>j != 17</code>
>	Is <b>greater than</b>	<code>j &gt; 0</code>
<	Is <b>less than</b>	<code>j &lt; 100</code>
>=	Is <b>greater than or equal</b> to	<code>j &gt;= 23</code>
<=	Is <b>less than or equal</b> to	<code>j &lt;= 13</code>
===	Is <b>equal</b> to (and of the same type)	<code>j === 56</code>
!==	Is <b>not equal</b> to (and of the same type)	<code>j !== '1'</code>

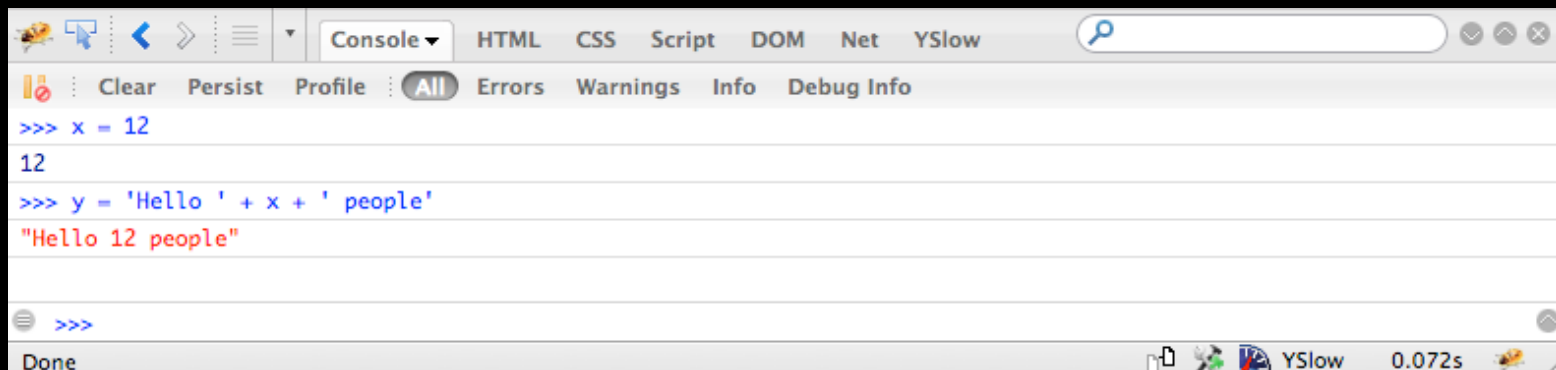
# Logical Operators

*Table 14-5. Logical operators*

Operator	Description	Example
&&	And	<code>j == 1 &amp;&amp; k == 2</code>
	Or	<code>j &lt; 100    j &gt; 0</code>
!	Not	<code>! (j == k)</code>

# String Concatenation

- JavaScript string concatenation is like Python and Java and uses "+" versus PHP which uses "."
- Like the PHP "." operator, it automatically converts non-string values to strings as needed



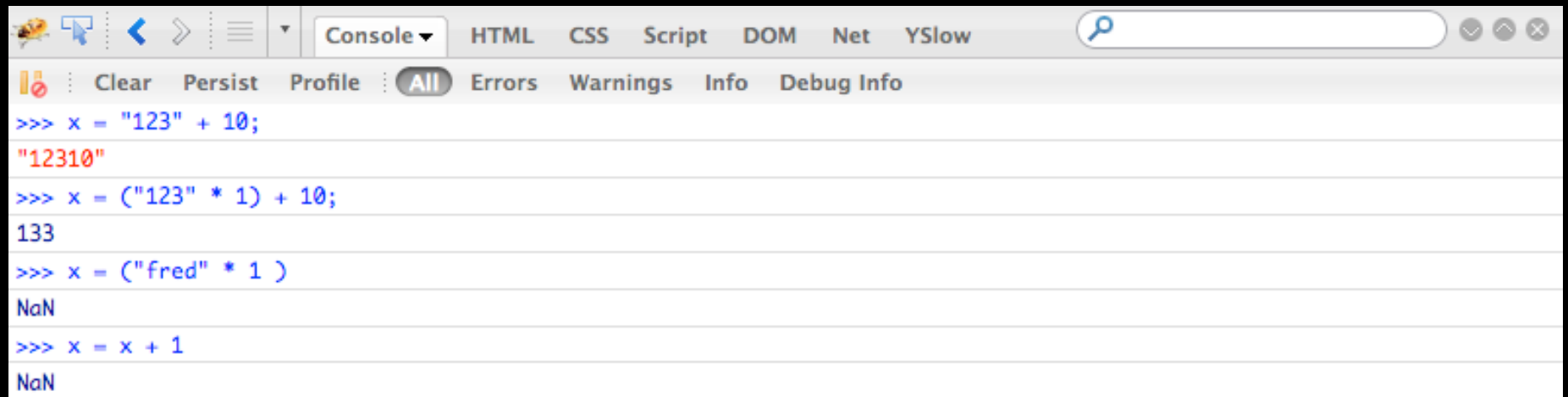
```
>>> x = 12
12
>>> y = 'Hello ' + x + ' people'
"Hello 12 people"
>>>
```

The screenshot shows a browser's developer console with the following content:

- Buttons: Clear, Persist, Profile, All, Errors, Warnings, Info, Debug Info
- Code: `>>> x = 12`
- Output: `12`
- Code: `>>> y = 'Hello ' + x + ' people'`
- Output: `"Hello 12 people"`
- Code: `>>>`
- Status: Done
- Performance: YSlow 0.072s

# Variable Typing

- JavaScript is a loosely typed language and does automatic type conversion when doing expressions

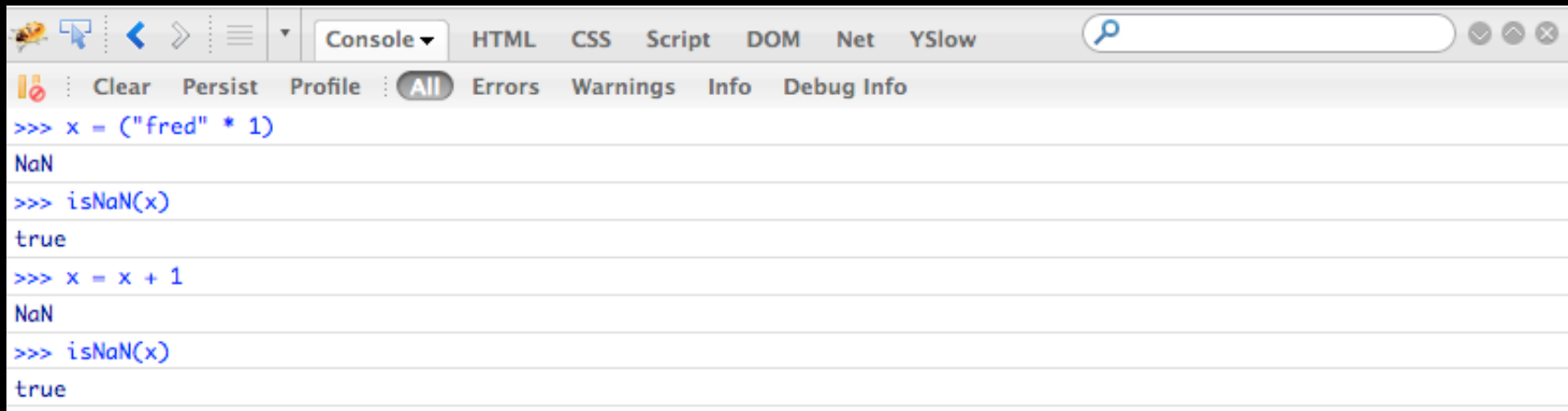


The screenshot shows a browser's developer console with the following content:

```
>>> x = "123" + 10;  
"12310"  
>>> x = ("123" * 1) + 10;  
133  
>>> x = ("fred" * 1 )  
NaN  
>>> x = x + 1  
NaN
```

# Variable Conversion

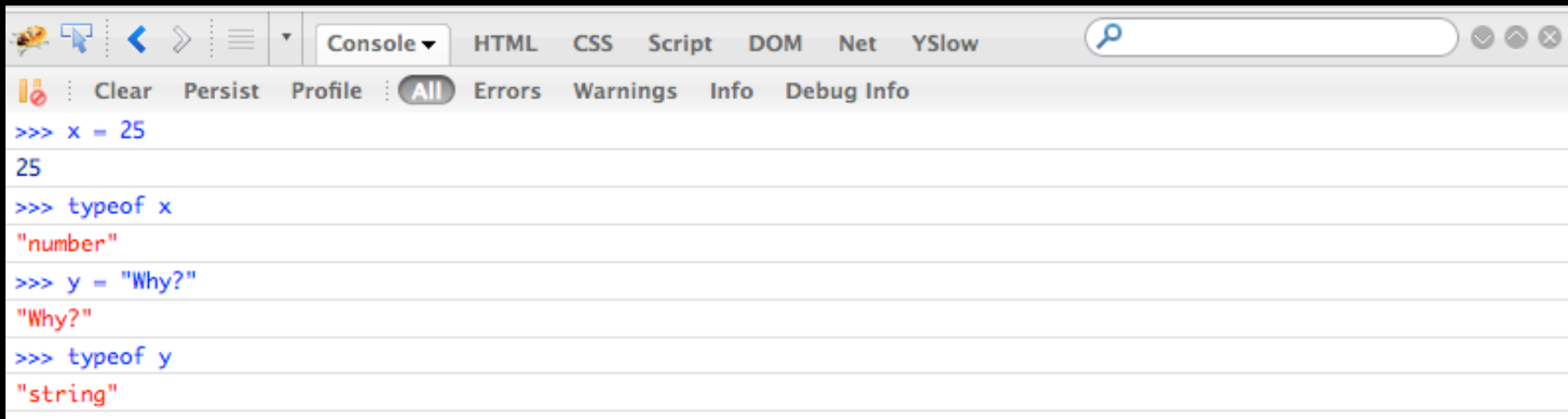
- If a string cannot be converted to a number you end up with "Not a Number" or "NaN" - it is a value but all operations with NaN as a operand end up with NaN (i.e. it is sticky)



```
>>> x = ("fred" * 1)
NaN
>>> isNaN(x)
true
>>> x = x + 1
NaN
>>> isNaN(x)
true
```

# Determining Type

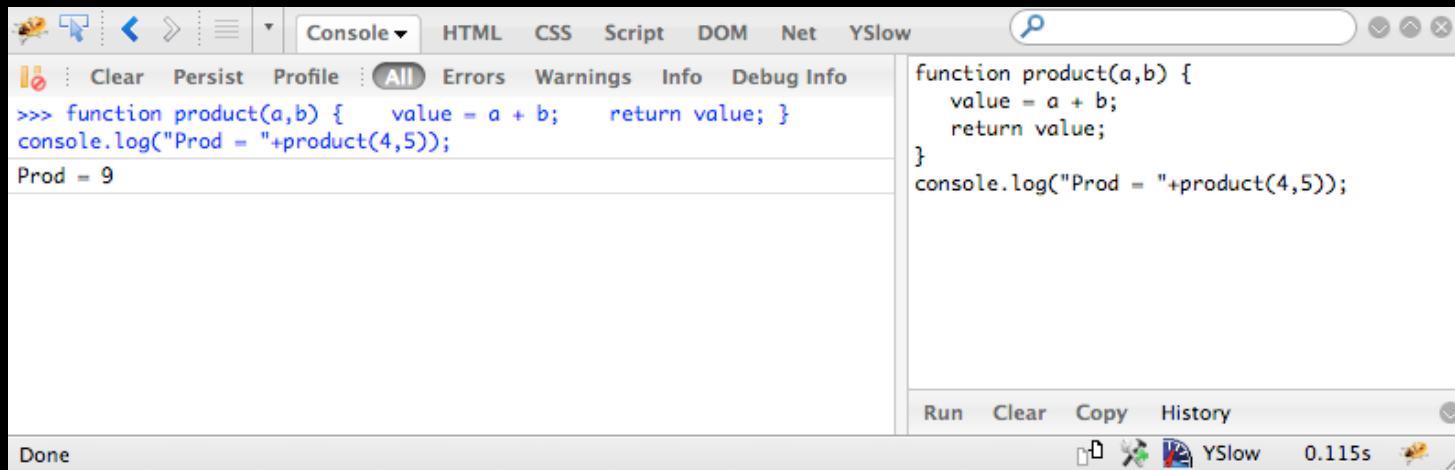
- JavaScript provides a unary typeof operator that returns the type of a variable or constant as a string



```
>>> x = 25
25
>>> typeof x
"number"
>>> y = "Why?"
"Why?"
>>> typeof y
"string"
```

# Functions

- Functions use a typical syntax and are indicated using the **function** keyword
- The **return** keyword functions as expected



The screenshot shows a browser's developer console with the following content:

```
function product(a,b) {  
  value = a + b;  
  return value;  
}  
console.log("Prod = "+product(4,5));
```

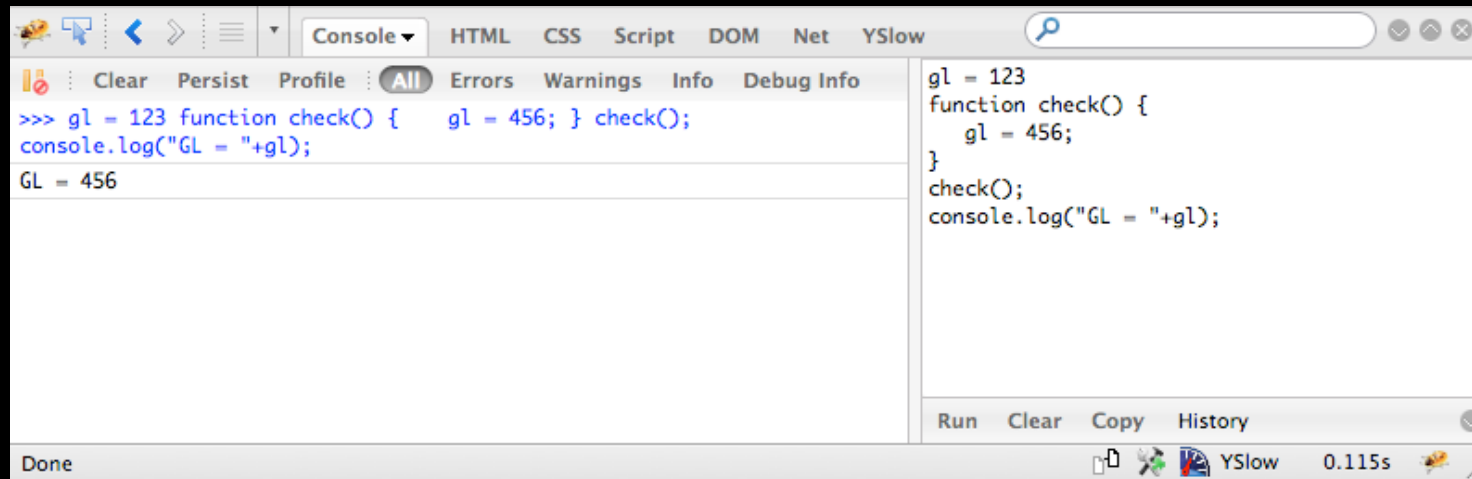
The console output shows:

```
Prod = 9
```

The console interface includes a search bar, a dropdown menu for 'Console', and tabs for 'HTML', 'CSS', 'Script', 'DOM', 'Net', and 'YSlow'. The 'Script' tab is active, showing the function definition. The console output is visible below the code. The status bar at the bottom indicates 'Done' and 'YSlow 0.115s'.

# Scope - Global (default)

- Variables that are defined outside a function, that are referenced inside of a function have global scope
- This is a little different than what we are used to



The screenshot shows a browser's developer console with the following content:

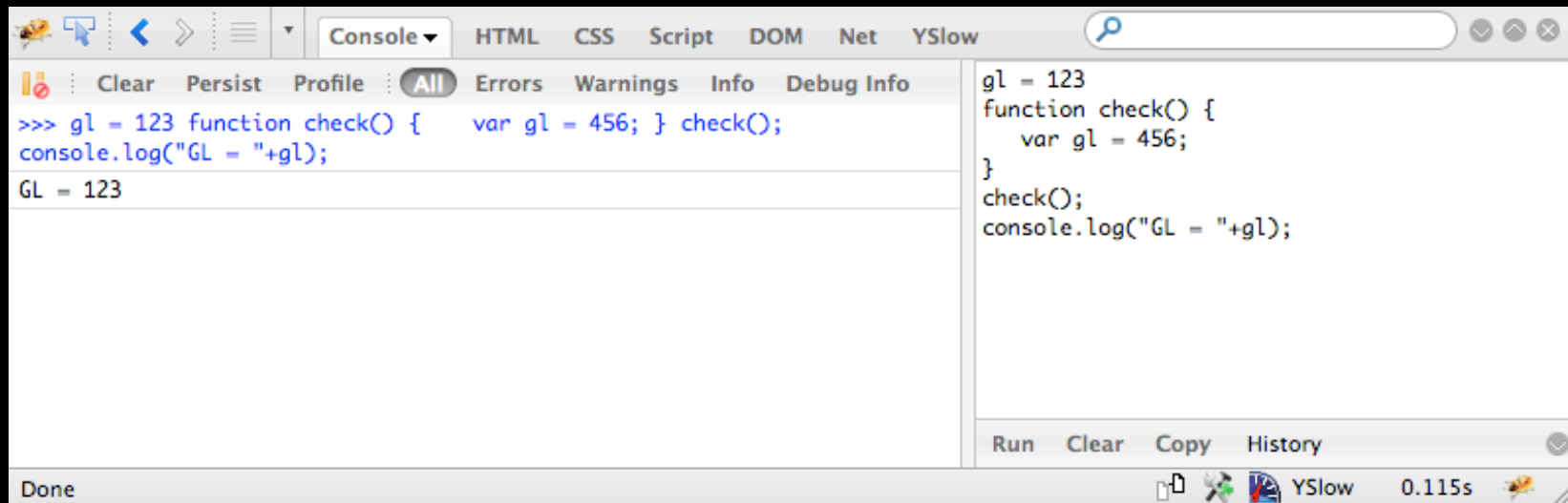
```
gl = 123
function check() {
  gl = 456;
}
check();
console.log("GL = "+gl);
```

The console output shows: `GL = 456`

The console interface includes tabs for Console, HTML, CSS, Script, DOM, Net, and YSlow. The console is currently set to 'All' and shows the execution of the provided code. The status bar at the bottom indicates 'Done' and a YSlow performance score of 0.115s.

# Making a Variable Local

- In a function the parameters (formal arguments) are local and any variable we mark with the **var** keyword are local too



The screenshot shows a browser's developer console with the following content:

```
>>> gl = 123 function check() { var gl = 456; } check();
console.log("GL = "+gl);
GL = 123
```

The right pane of the console shows the function definition and the execution result:

```
gl = 123
function check() {
  var gl = 456;
}
check();
console.log("GL = "+gl);
```

The bottom status bar indicates "Done" and "YSlow 0.115s".

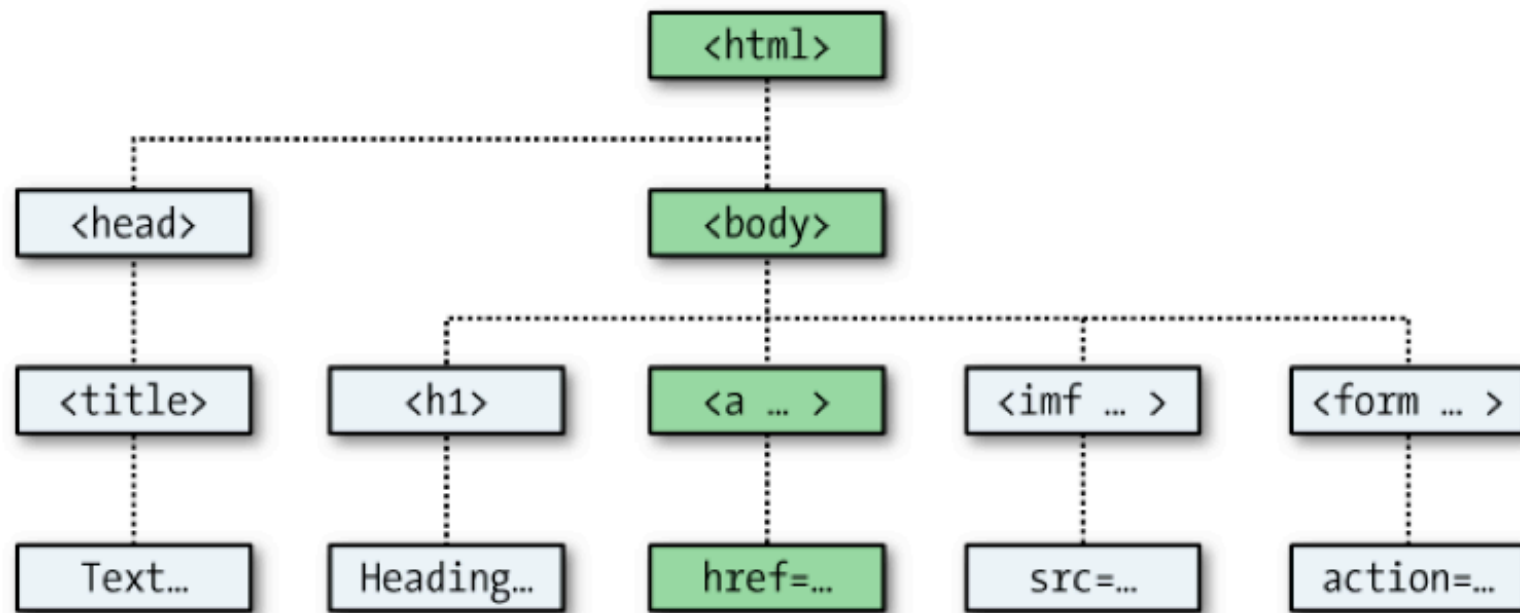
# DOM

## Document Object Model

# Document Object Model

- JavaScript can manipulate the current HTML document
- The “Document Object Model” tells us the syntax to use to access various “bits” of the current screen to read and/or manipulate
- You can even find pieces of the model by **id** attribute and change them
- We can read and write the DOM from JavaScript

[http://en.wikipedia.org/wiki/Document\\_Object\\_Model](http://en.wikipedia.org/wiki/Document_Object_Model)



# DOM's are not Identical

- Not all browsers represent their page exactly the same.
- This makes it a challenge to keep all of your JavaScript working on all browsers
- Also means you need to test your code over and over on each browser
- Aargh..



## IE8: Clicking on Portfolio Layouts causes the page to load, then refresh and then no options display

Edit

Assign

Comment

More Actions ▾

Start Progress

Resolve Issue

Workflow ▾

Views ▾

### ▼ Description

Tested in both FF and IE, this problem only affects IE.

Clicking on Portfolio Layouts causes the page to load, then refresh and then no options display. Am attaching a video that shows what happens.

### ▼ Attachments



PortfolioLayouts\_IE8.swf

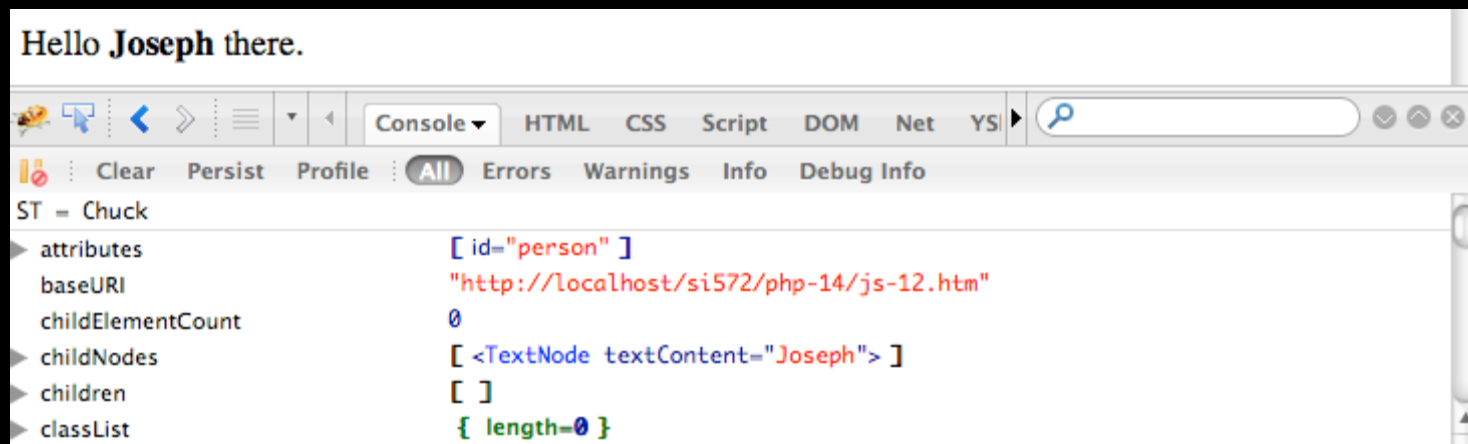
684 kB

23-Sep-2011 06:54

# Using getElementById()

- In order not to have to traverse each unique DOM, we use a function call that all browsers support that allows us to bypass the DOM structure and jump to a particular tag within the DOM and manipulate that tag

```
<p>
Hello <b><span id="person">Chuck</span></b> there.
</p>
<script type="text/javascript">
st = document.getElementById('person').innerHTML;
console.log("ST = "+st);
document.getElementById('person').innerHTML = 'Joseph';
console.dir(document.getElementById('person'));
</script>
```



<http://www.dr-chuck.com/javascript/four.htm>

```
<p>
<a href="#"
  onclick="document.getElementById('stuff').innerHTML = 'BACK';">BACK</a>
<a href="#"
  onclick="document.getElementById('stuff').innerHTML = 'FORTH';">FORTH</a>
</p>
<p>
Hello <b><span id="stuff">Stuff</span></b> there.
</p>
```

## Updating the Browser Document

This is one reason why you can only have one id per document.

[BACK FORTH](#)

Hello **Stuff** there.

[BACK FORTH](#)

Hello **BACK** there.

[BACK FORTH](#)

Hello **FORTH** there.

# JQuery to the rescue

- While the DOM's are not particularly portable, and direct DOM manipulation is a little clunky, there are a number of JavaScript frameworks that handle the myriad of subtle differences between browsers
- <http://jquery.org/>
- With JQuery, instead of manipulating the DOM, we use JQuery functions and everything works much better..

Questions...