

JavaScript

Dr. Charles Severance

www.wa4e.com

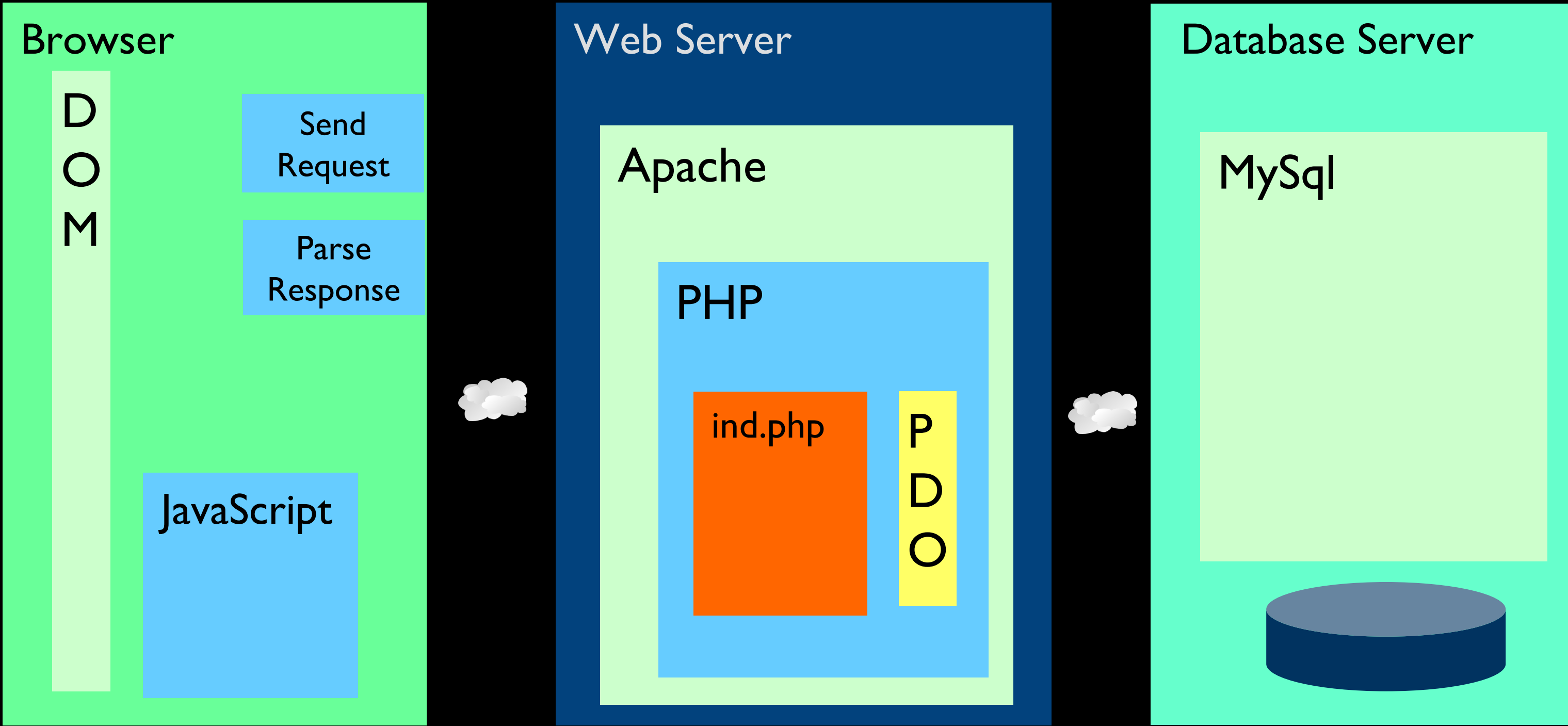
<http://www.wa4e.com/code/javascript>

<http://www.wa4e.com/code/javascript.zip>





Time



<http://www.wa4e.com/code/rrc/>



About JavaScript

- Introduced in Netscape in 1995
- Developed by Brandon Eich
- Named to make use of Java market buzz
- Standardized today as ECMAScript



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

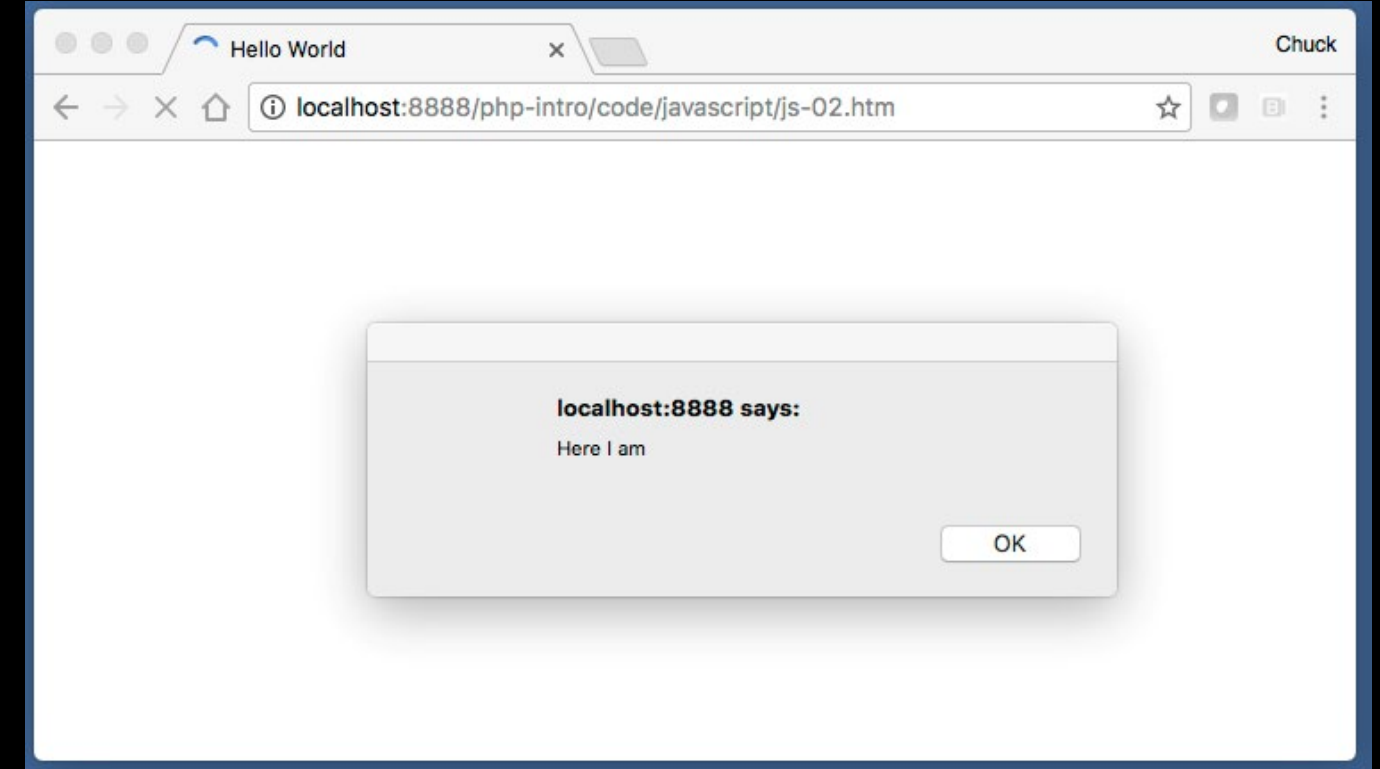
Hello World

Second Paragraph

Low-Level Debugging

- 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>
```



js-02.htm

Including JavaScript

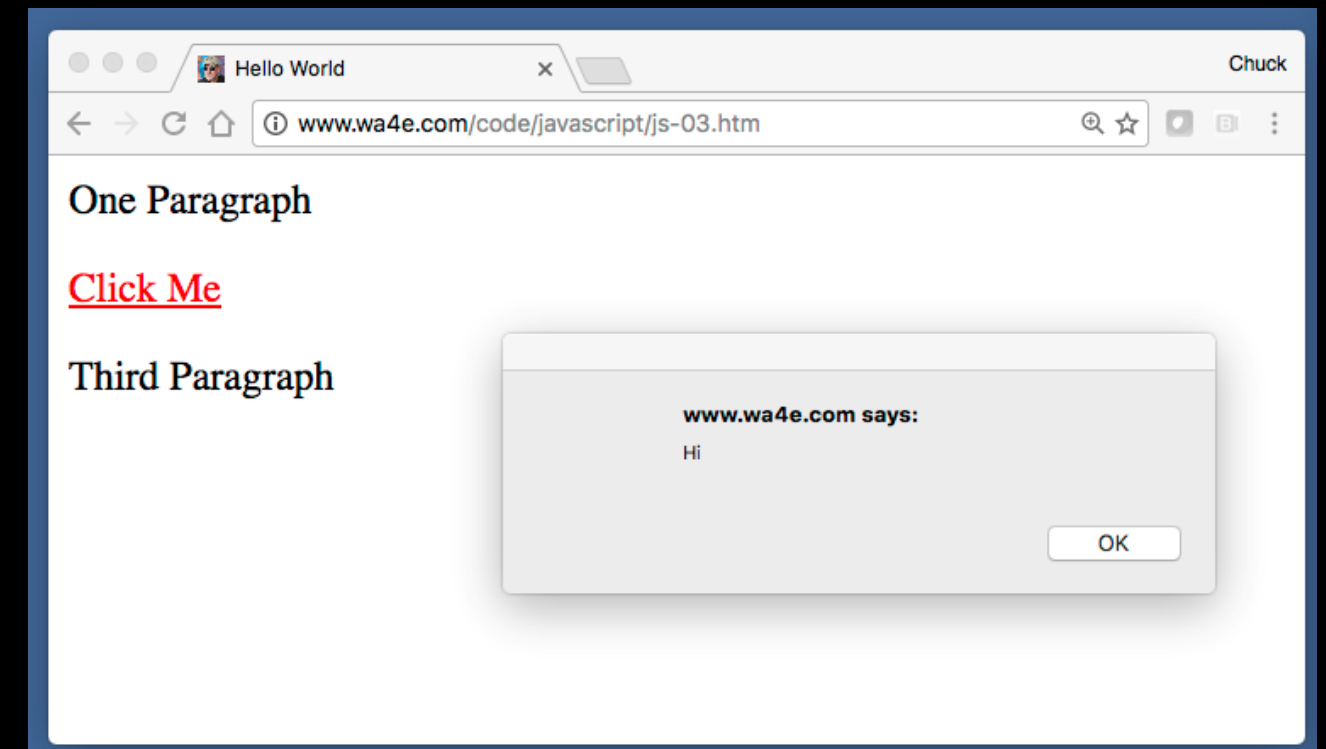
Three Patterns:

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

js-03.htm

```
<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" 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>");
```

js-04.htm

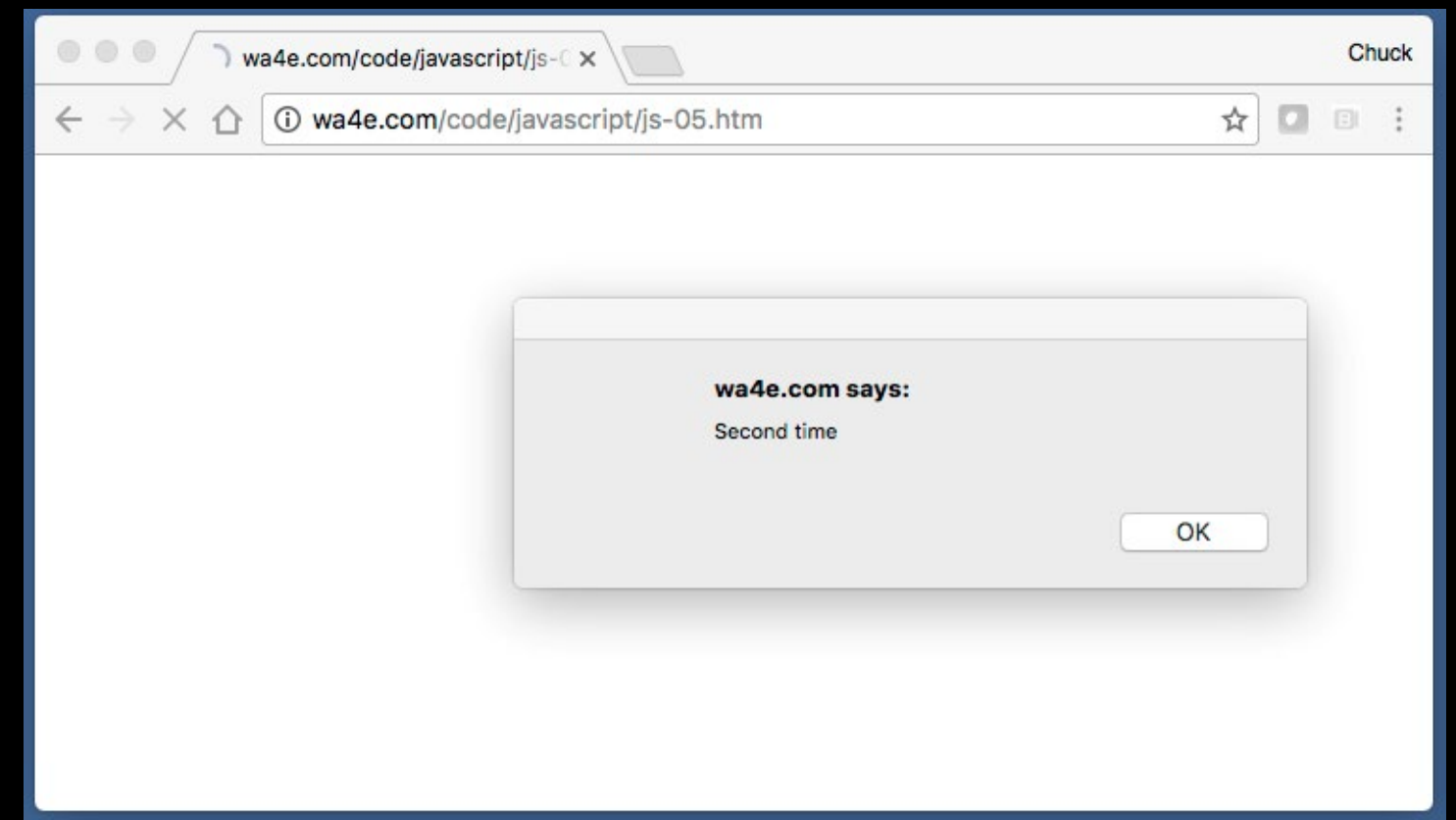
Basic JavaScript

Syntax Errors

- As in 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

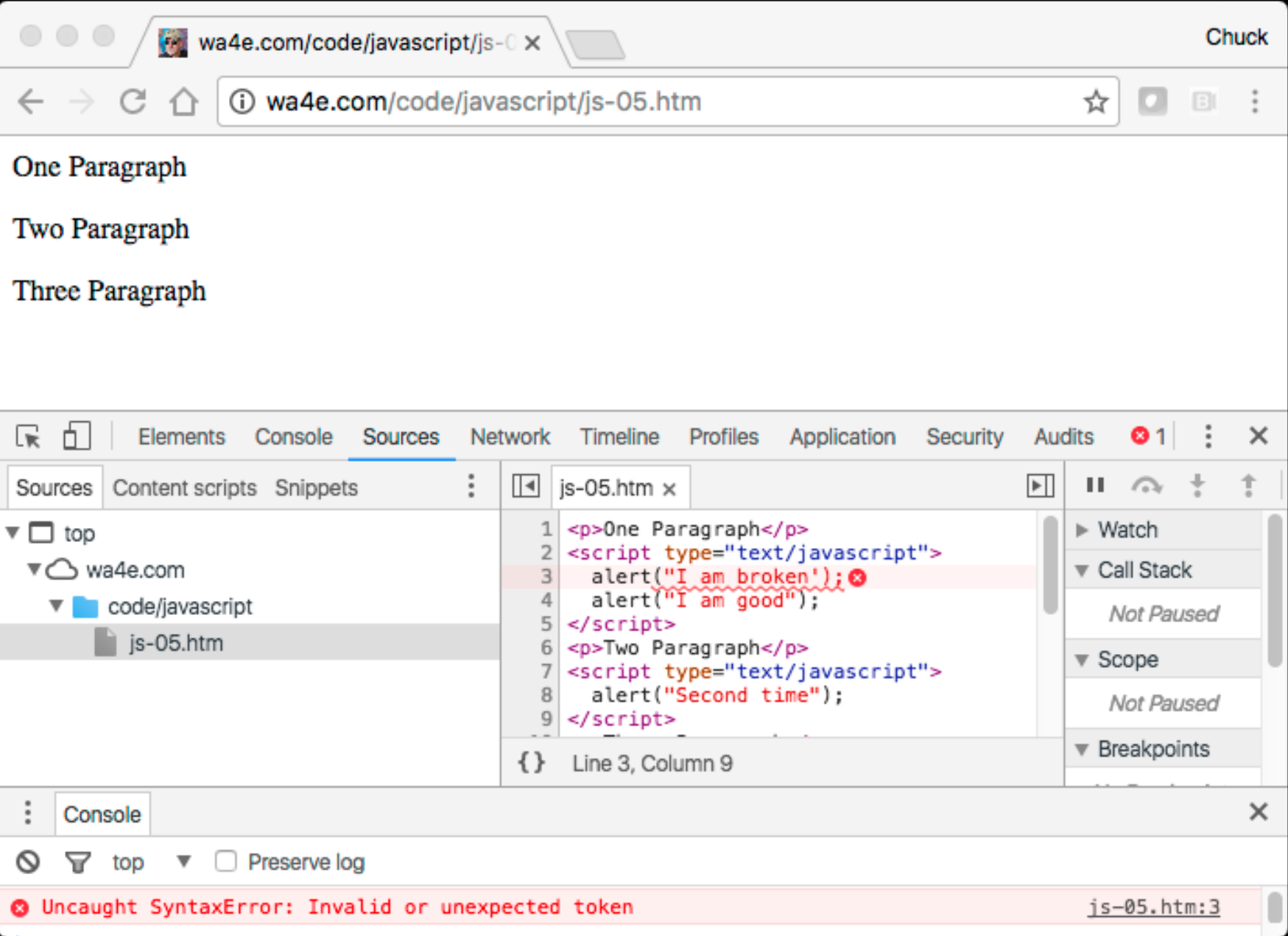
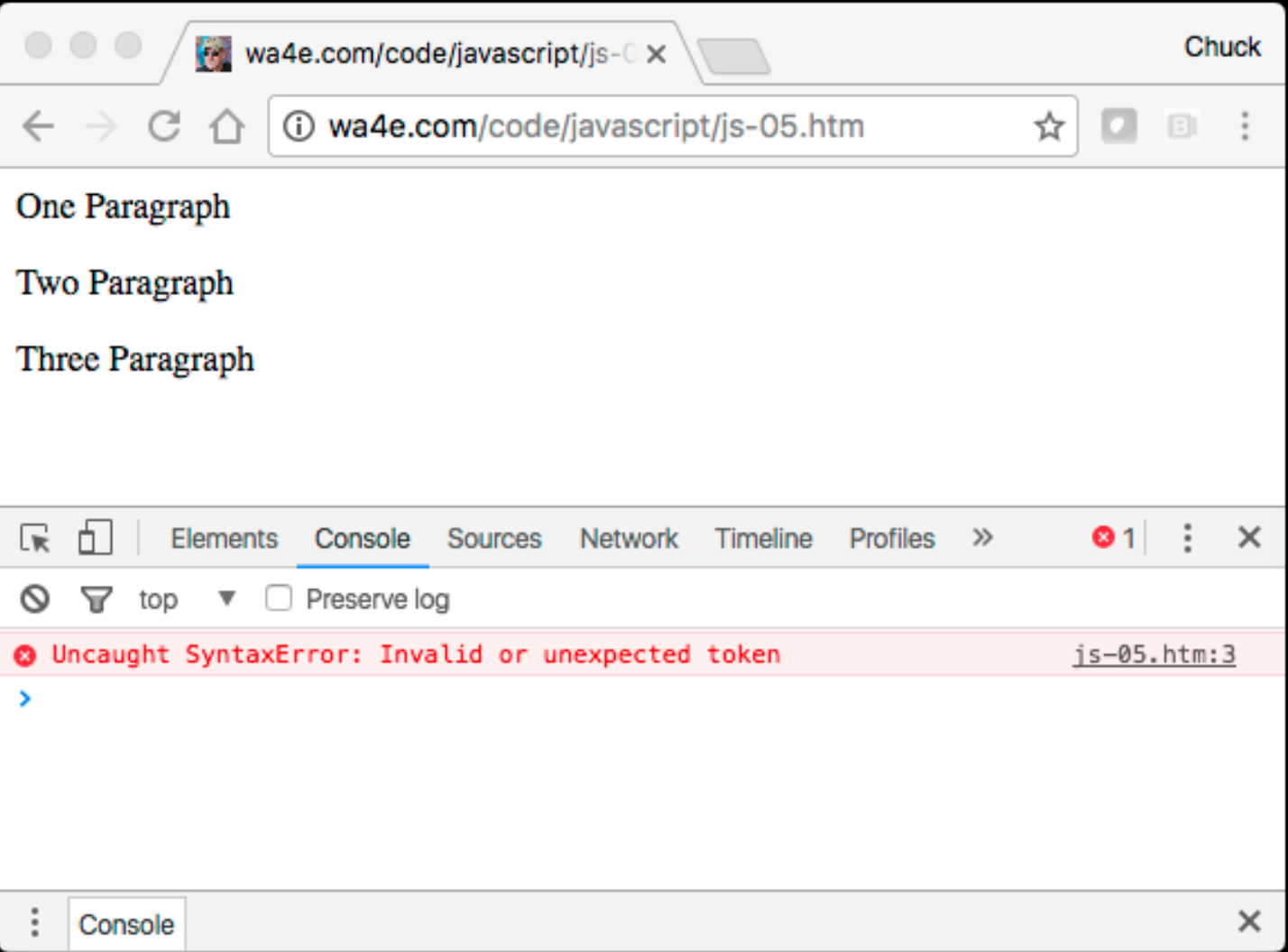
js-05.htm

```
<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 developers, we need to look for the errors - sometimes it takes a minute to even remember to check for a JS error.



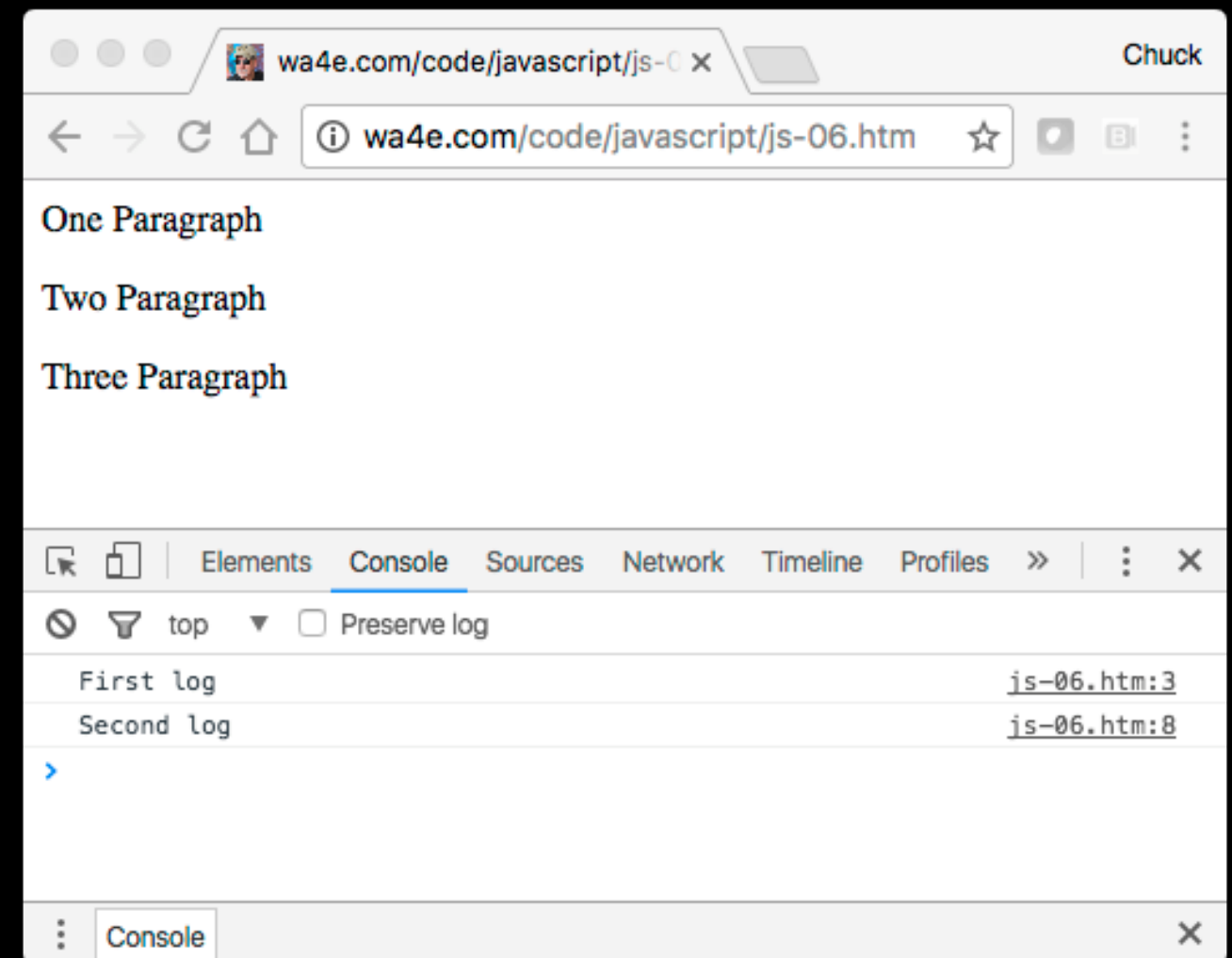
Console Logging

- Debugging using `alert()` can get tiring - sometimes you want to record what happens in case something goes wrong
- `console.log("String")` - and many more functions

Note: Requires recent browsers

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

js-06.htm



Console is Not Always There

Some browsers only define the `console` if a debugger is running.

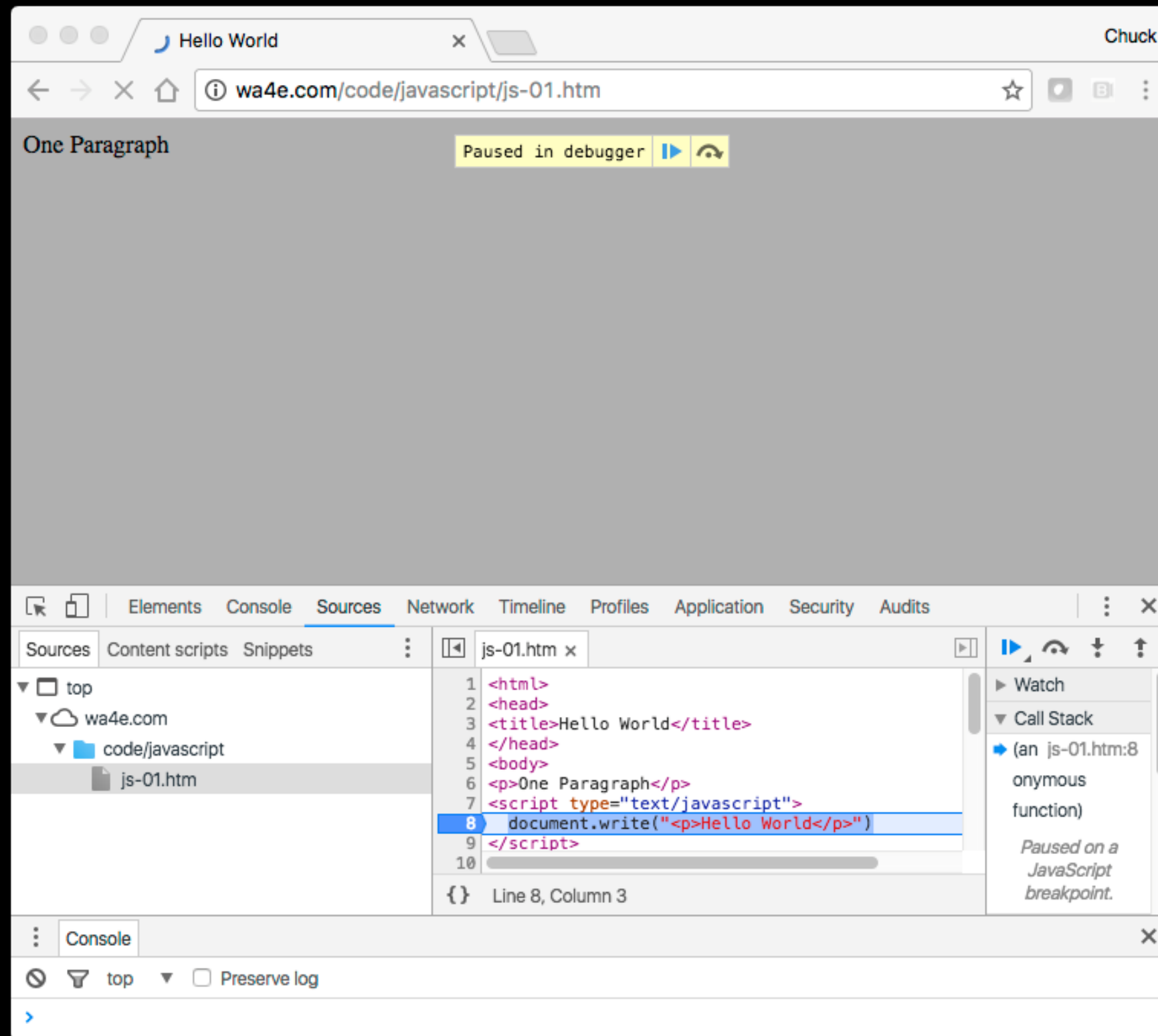
```
window.console && console.log(...)
```

```
if (typeof console == "undefined") {  
    this.console = {log: function() {} }  
}
```

<http://stackoverflow.com/questions/3326650/console-is-undefined-error-for-internet-explorer>

Using the Debugger (Chrome)

- Get into a source view.
- Click on a line of JavaScript to set a breakpoint.
- Reload the page.



JavaScript – Core Language Features

Comments in JavaScript = Awesome

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

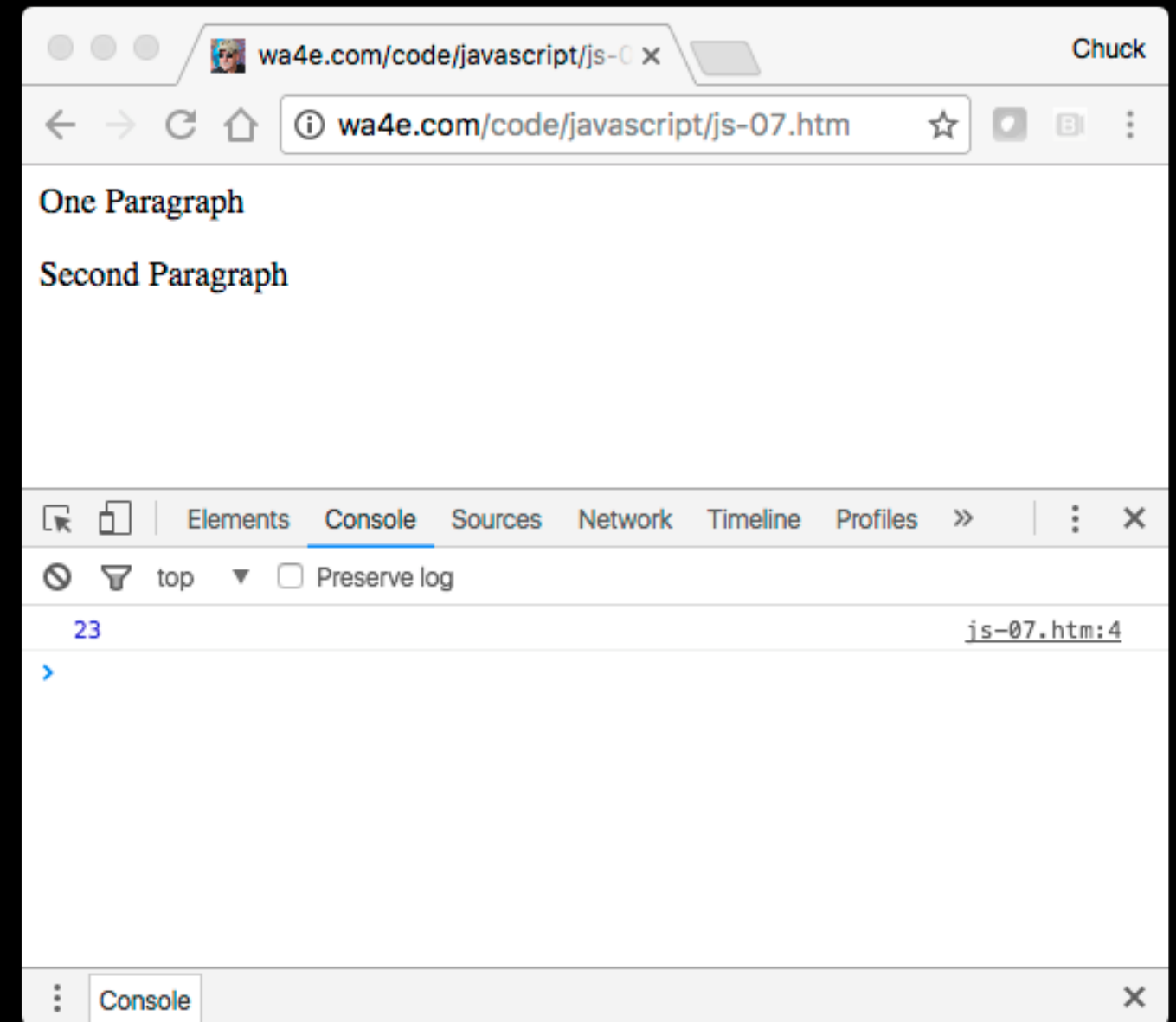
```
/* This is a section of  
multiline comments that will  
not be interpreted */
```

Statements

- White space and newlines do not matter.
- Statements end with a semicolon ;
- There are cases where you can leave the semicolon off, but don't bother exploiting this feature - just add semicolons like in 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>
```

js-07.htm



Variable Names

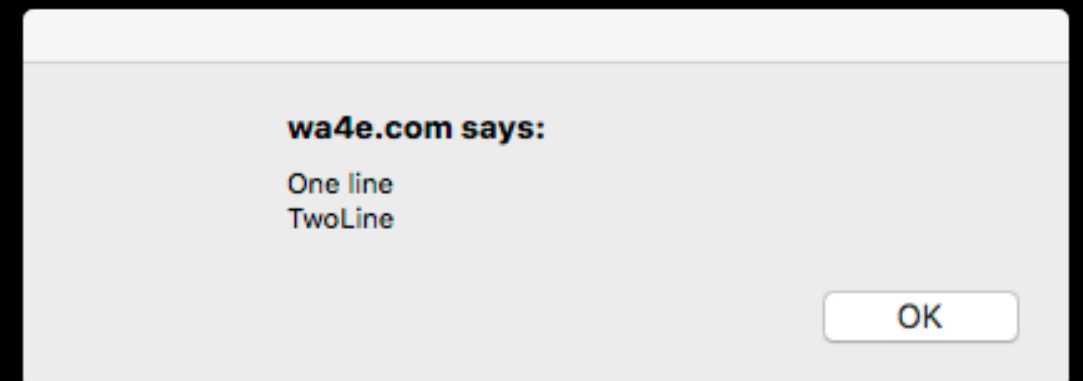
- Valid Characters: a-z, A-Z, 0-9, _ and \$
- Must not start with a number
- Names are case sensitive
- Starting with a dollar sign is considered “tacky”

String Constants

[js-08.htm](#)

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

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

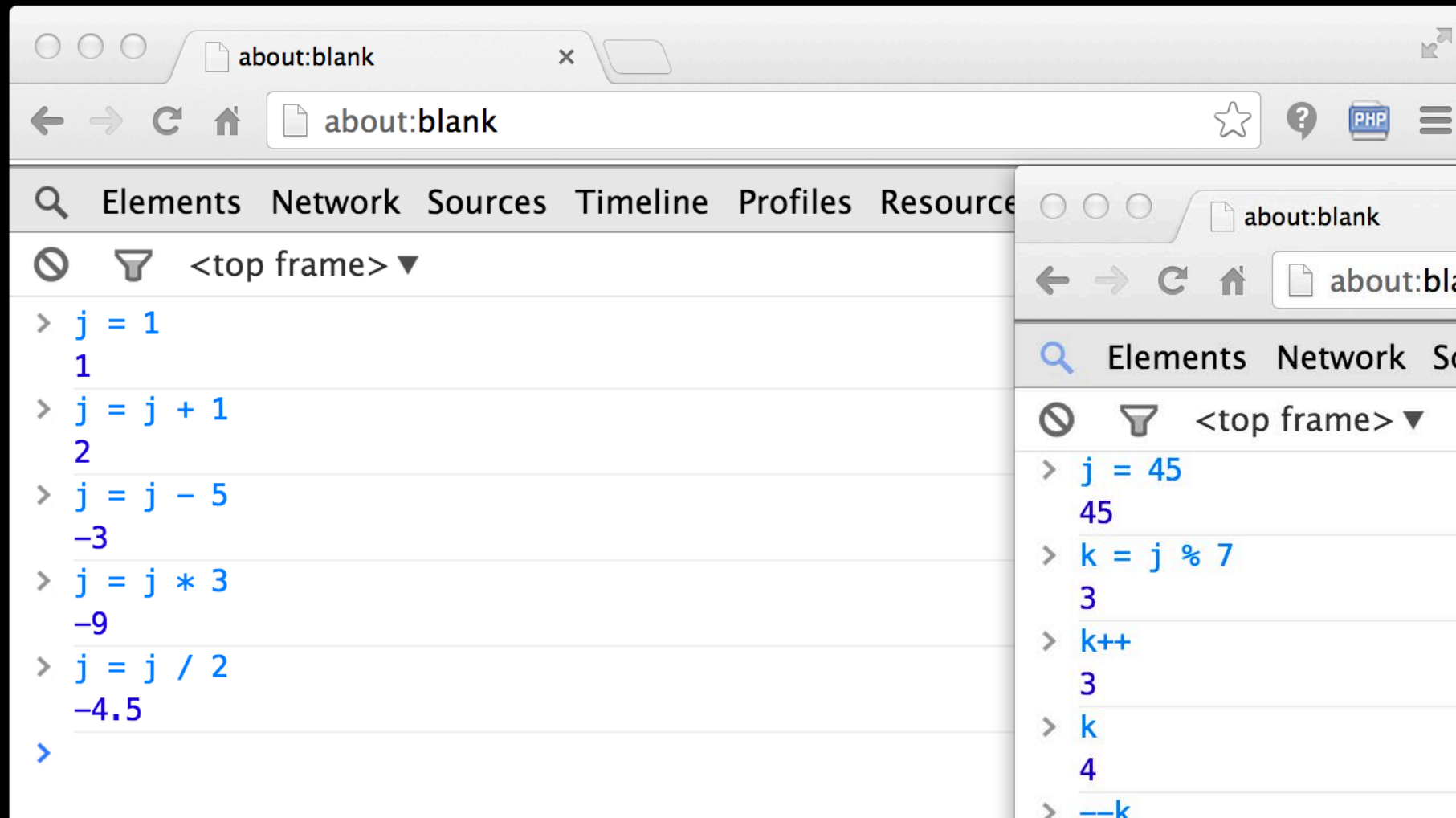


Numeric Constants

As you would expect...

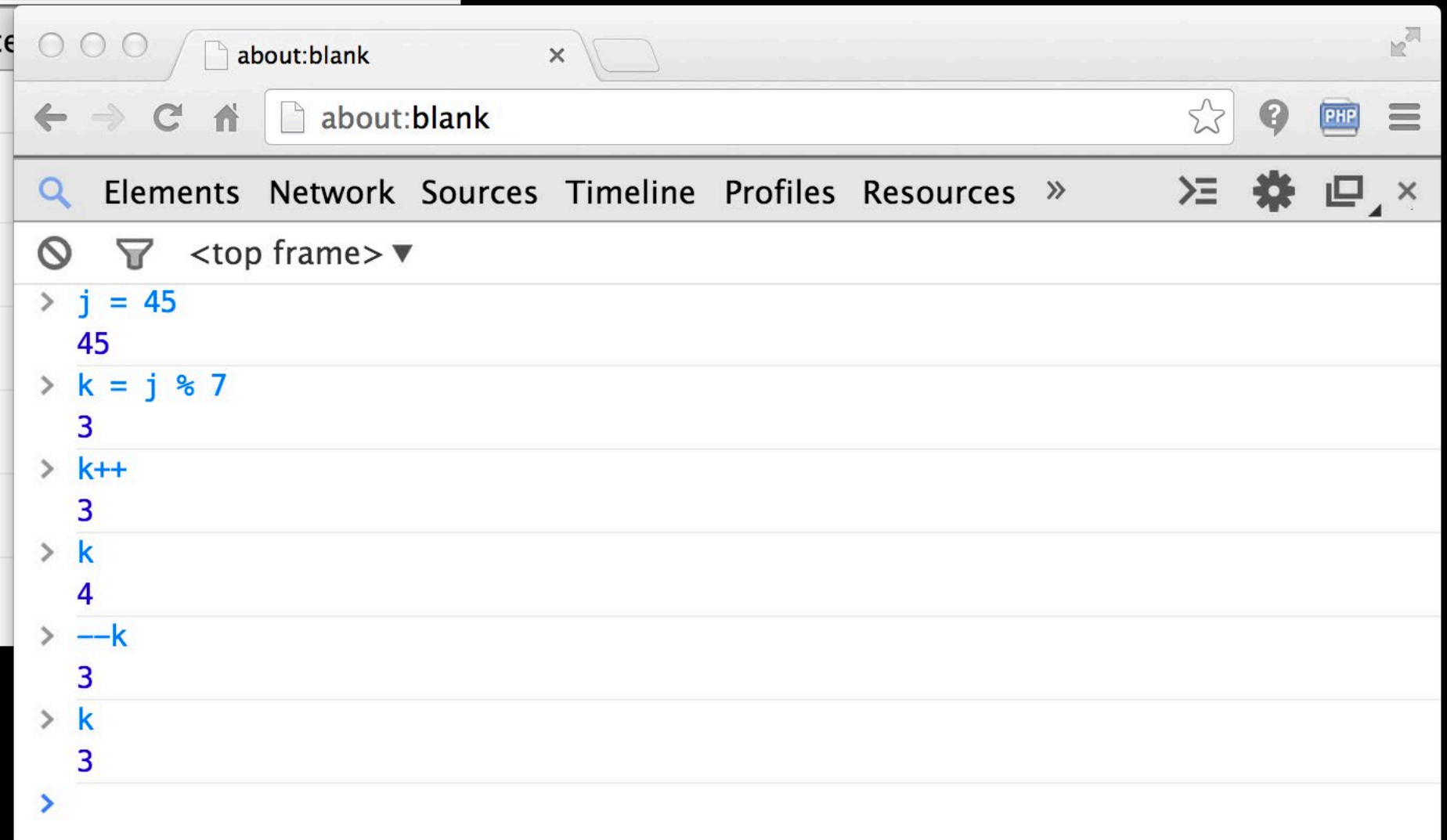
JavaScript – Variables and Expressions

Operators



A screenshot of a web browser's developer console. The browser window shows 'about:blank'. The developer tools are open, and the console shows a series of JavaScript commands and their results:

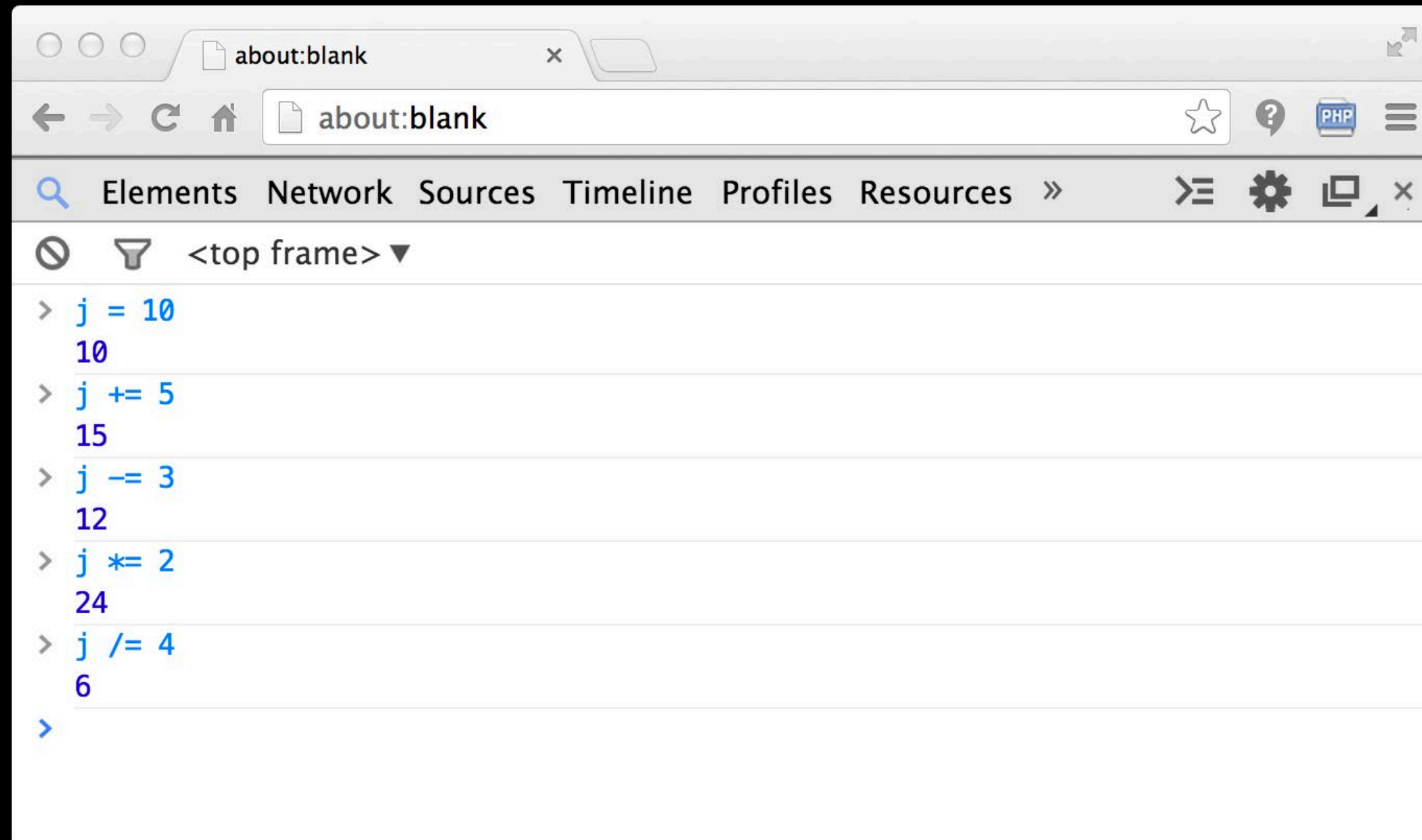
```
> j = 1
1
> j = j + 1
2
> j = j - 5
-3
> j = j * 3
-9
> j = j / 2
-4.5
>
```



A screenshot of a web browser's developer console. The browser window shows 'about:blank'. The developer tools are open, and the console shows a series of JavaScript commands and their results:

```
> j = 45
45
> k = j % 7
3
> k++
3
> k
4
> --k
3
> k
3
>
```

Assignment (side effect) Operators



The screenshot shows a web browser window with the address bar set to 'about:blank'. The developer tools are open, displaying the 'Sources' panel. The console shows a series of JavaScript commands and their results:

```
> j = 10
10
> j += 5
15
> j -= 3
12
> j *= 2
24
> j /= 4
6
>
```

Comparison Operators

The image displays two side-by-side browser developer console screenshots, both showing the 'about:blank' page. The left console shows a series of assignments and comparisons for a variable 'j' with the value 10. The right console shows assignments and comparisons for a variable 'j' with the value false.

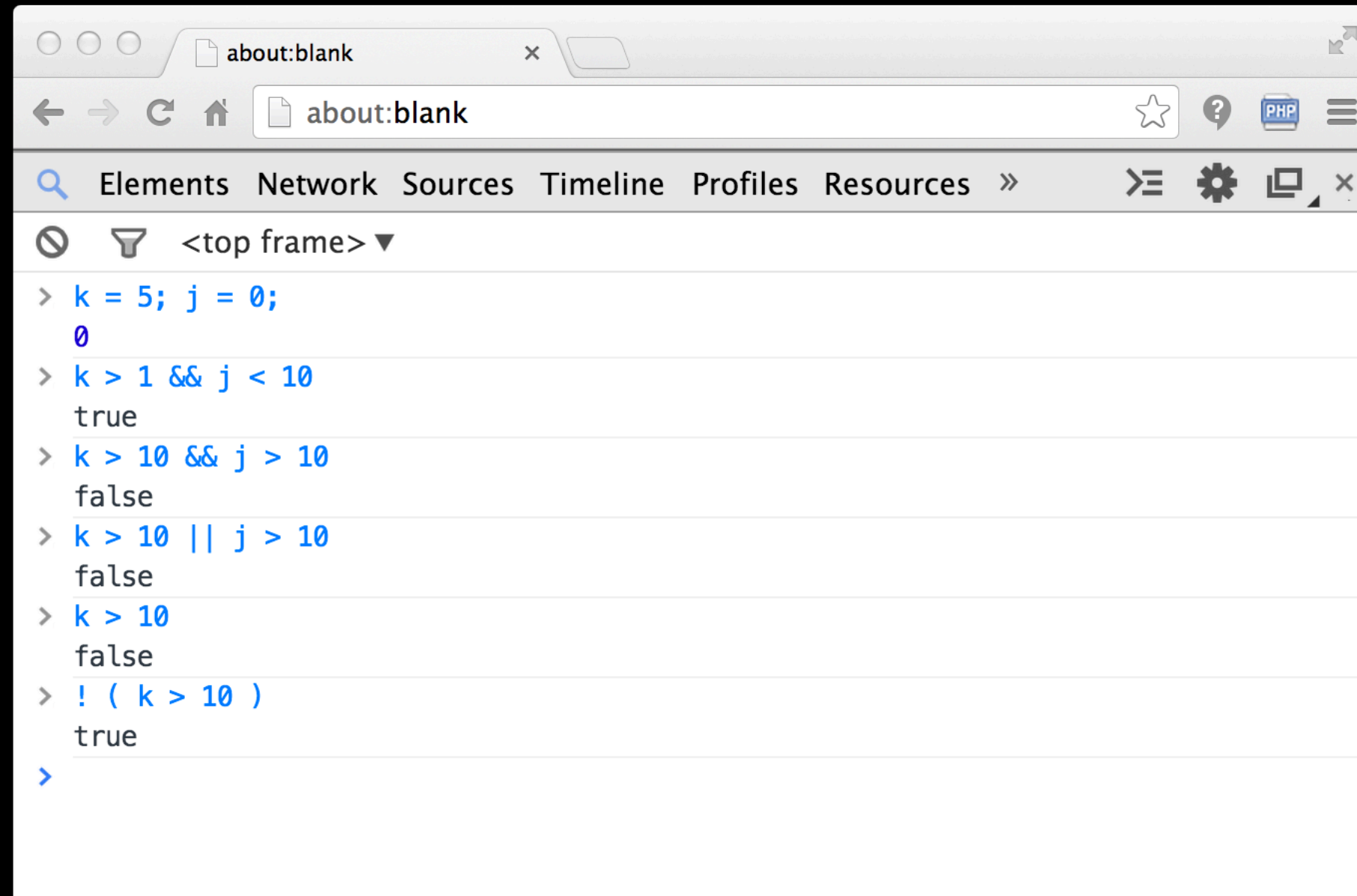
Left Console:

```
> j = 10
10
> j == 10
true
> j != 17
true
> j < 43
true
> j > 42
false
> j <= 10
true
>
```

Right Console:

```
> j = false
false
> j == 0
true
> j === 0
false
> j === false
true
> j !== false
false
> j !== true
true
> |
```

Logical Operators

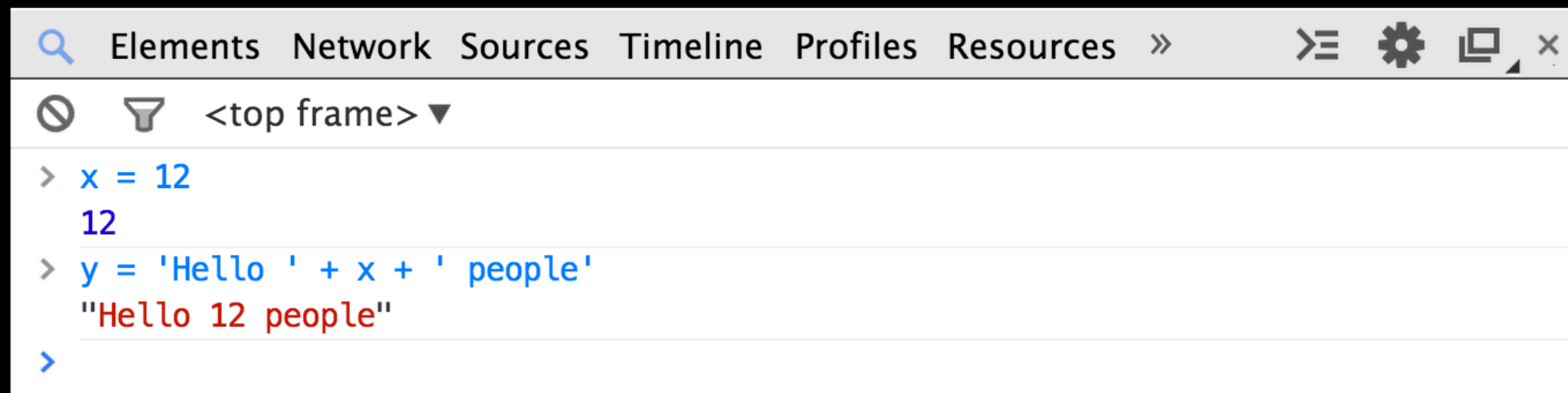


The screenshot shows a web browser window with a single tab titled 'about:blank'. The developer tools are open, displaying the 'Elements' panel. The console shows the following JavaScript code and its results:

```
> k = 5; j = 0;
0
> k > 1 && j < 10
true
> k > 10 && j > 10
false
> k > 10 || j > 10
false
> k > 10
false
> ! ( k > 10 )
true
>
```

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.



The screenshot shows a web browser's developer console with the following tabs: Elements, Network, Sources, Timeline, Profiles, and Resources. The console is set to the 'Sources' tab and shows the execution of two lines of JavaScript code in the '<top frame>' context. The first line is `x = 12`, which returns the value `12`. The second line is `y = 'Hello ' + x + ' people'`, which returns the string `"Hello 12 people"`. The console output demonstrates how the numeric value of `x` is automatically converted to a string for concatenation.

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


Variable Typing

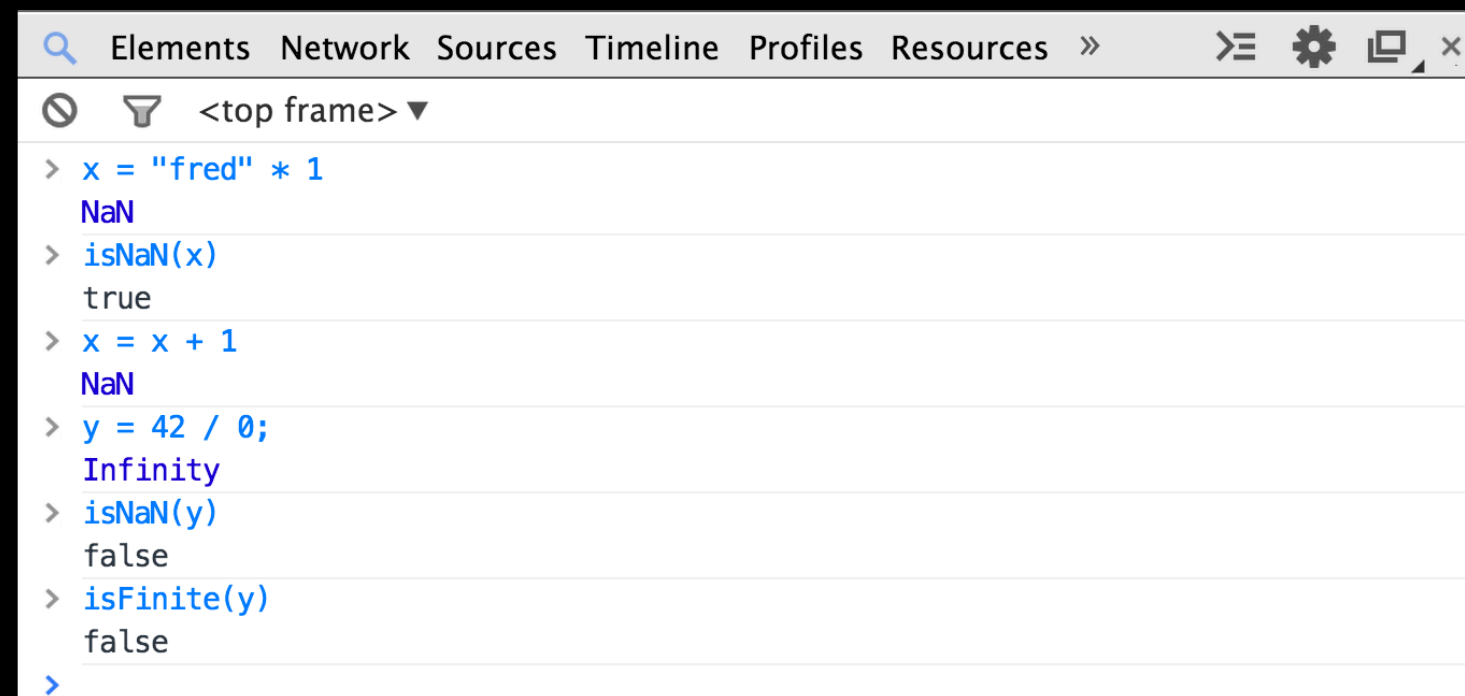
JavaScript is a loosely typed language and does automatic type conversion when evaluating expressions.



```
Elements Network Sources Timeline Profiles Resources »
<top frame> ▼
> x = "123" + 10;
  "12310"
> x = ("123" * 1 ) + 10;
  133
> x = ("fred" * 1)
  NaN
> x = x + 1
  NaN
>
```

Variable Conversion

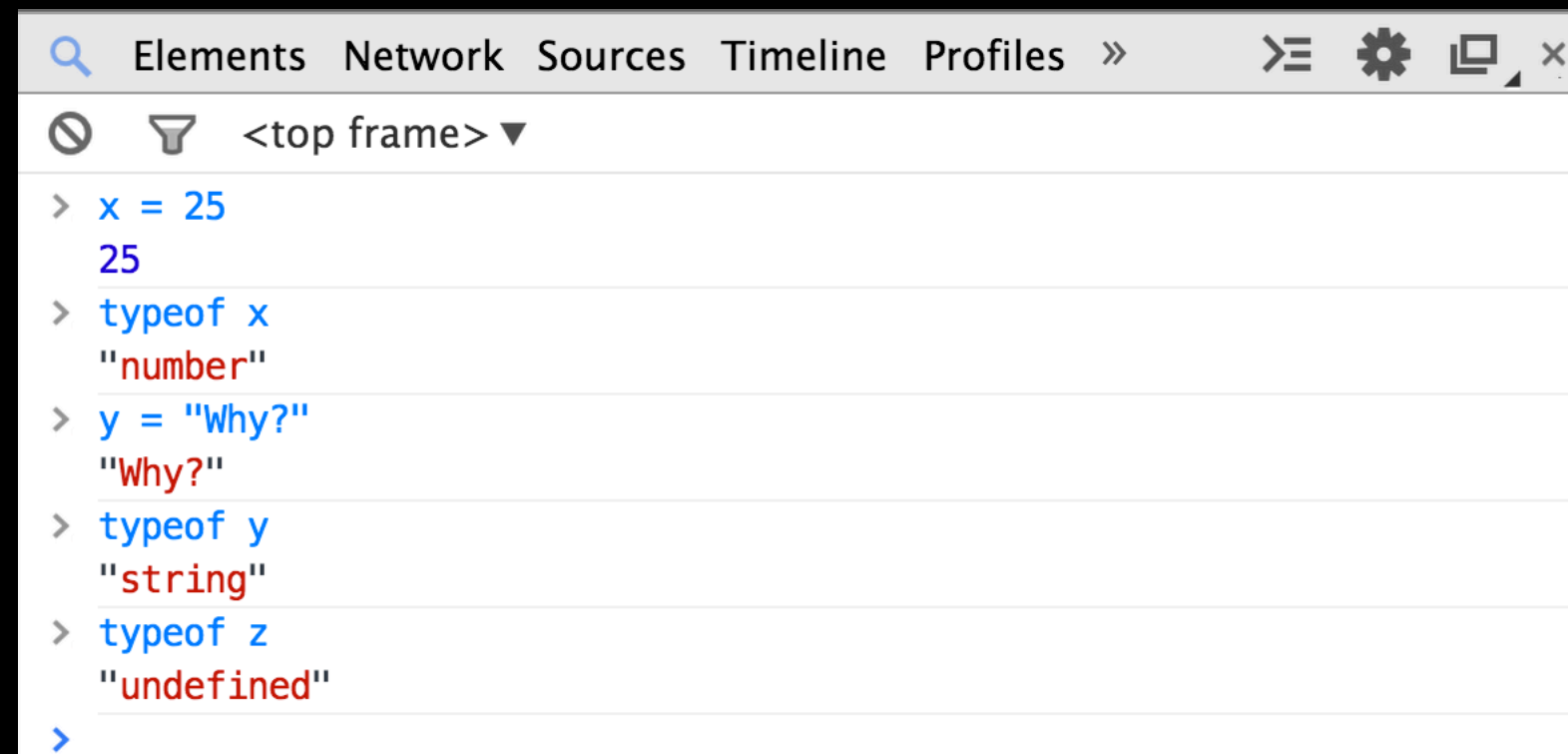
If a string cannot be converted to a number, you end up with “Not a Number” or “NaN”. It is a value, but it is sticky - all operations with NaN as a operand end up with NaN.



```
Elements Network Sources Timeline Profiles Resources »
<top frame> ▼
> x = "fred" * 1
  NaN
> isNaN(x)
  true
> x = x + 1
  NaN
> y = 42 / 0;
  Infinity
> isNaN(y)
  false
> isFinite(y)
  false
>
```

Determining Type

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



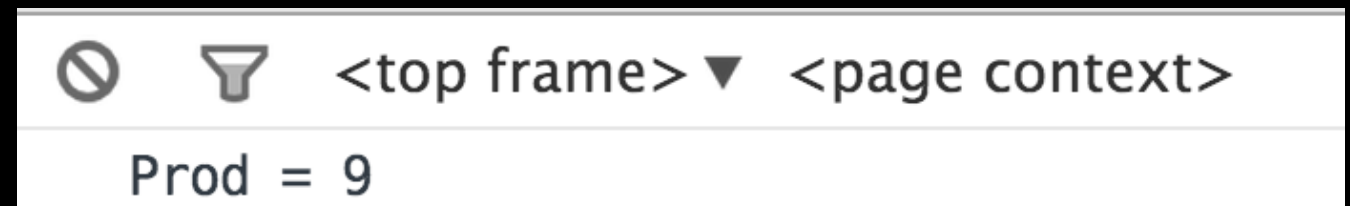
```
Elements Network Sources Timeline Profiles >> >=> ⚙️ 🖨️ ✕  
<top frame> ▼  
> x = 25  
25  
> typeof x  
"number"  
> y = "Why?"  
"Why?"  
> typeof y  
"string"  
> typeof z  
"undefined"  
>
```

Functions

js-09.htm

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

```
<script type="text/javascript">
function product(a,b) {
    value = a + b;
    return value;
}
console.log("Prod = "+product(4,5));
</script>
```



Scope - Global (default)

[js-10.htm](#)

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

```
<script type="text/javascript">
gl = 123;
function check() {
    gl = 456;
}
check();
window.console && console.log("GL = "+gl);
</script>
```



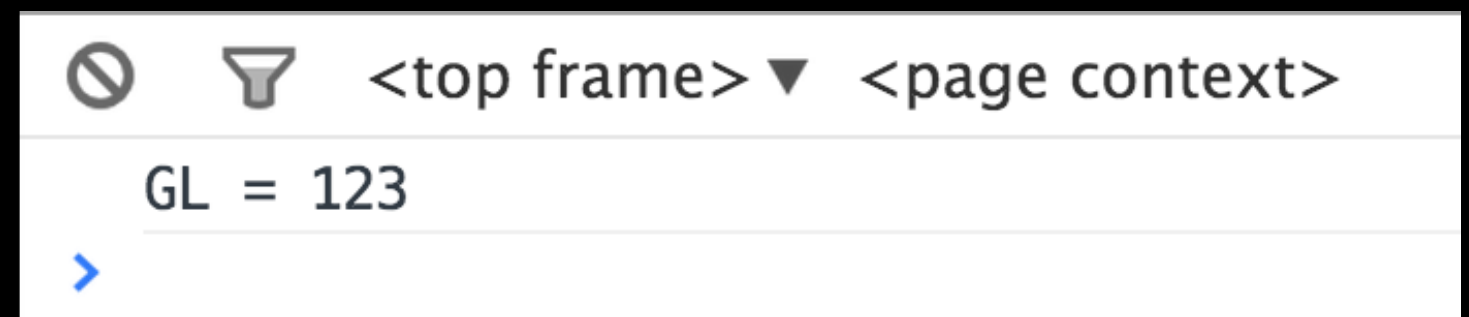
<top frame> ▼ <page context>
GL = 456

Making a Variable Local

[js-11.htm](#)

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

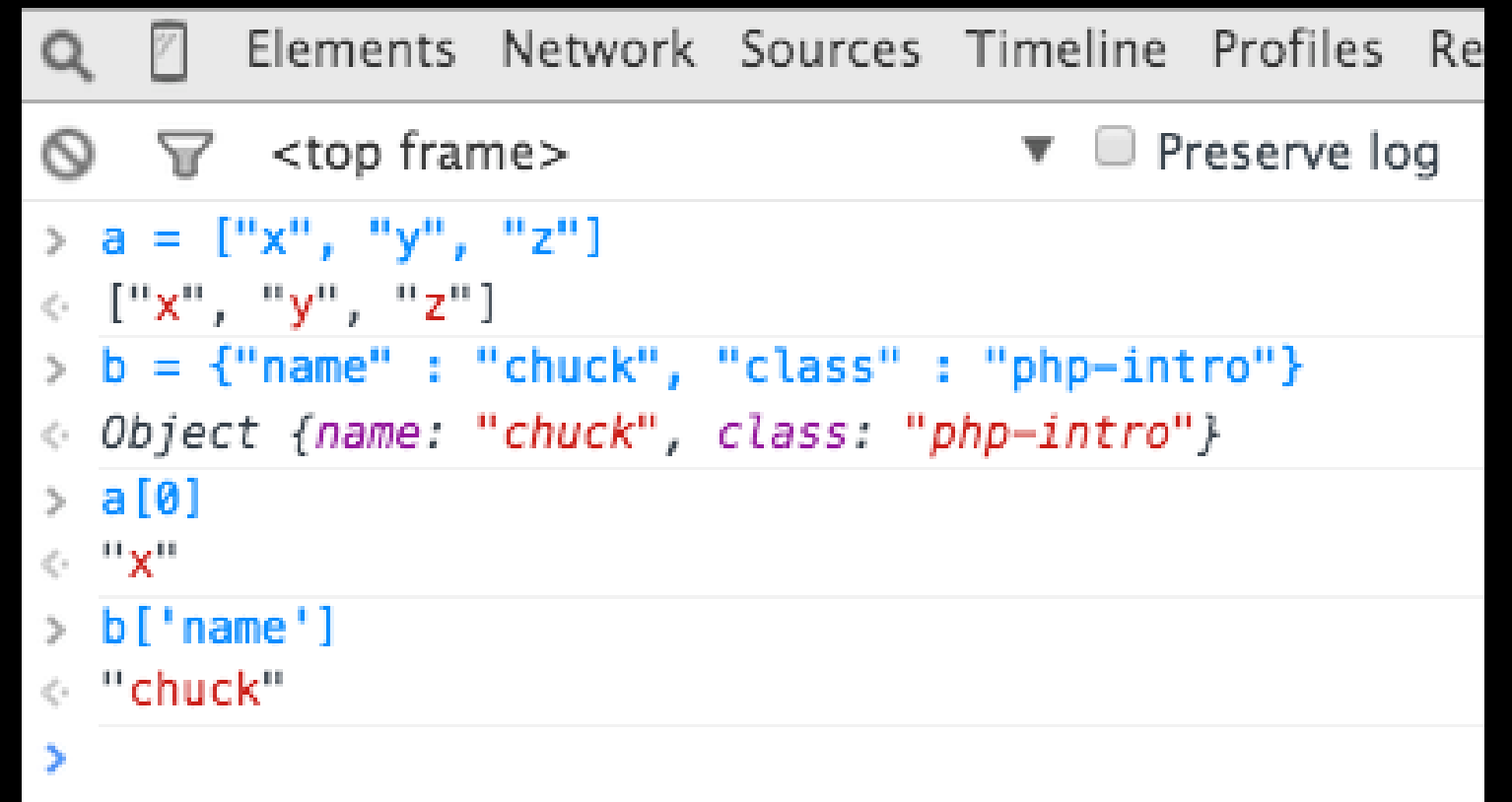
```
<script type="text/javascript">
gl = 123;
function check() {
    var gl = 456;
}
check();
window.console && console.log("GL = "+gl);
</script>
```



JavaScript – Arrays and Control Structures

Arrays in JavaScript

JavaScript supports both linear arrays and associative structures, but the associative structures are actually objects.



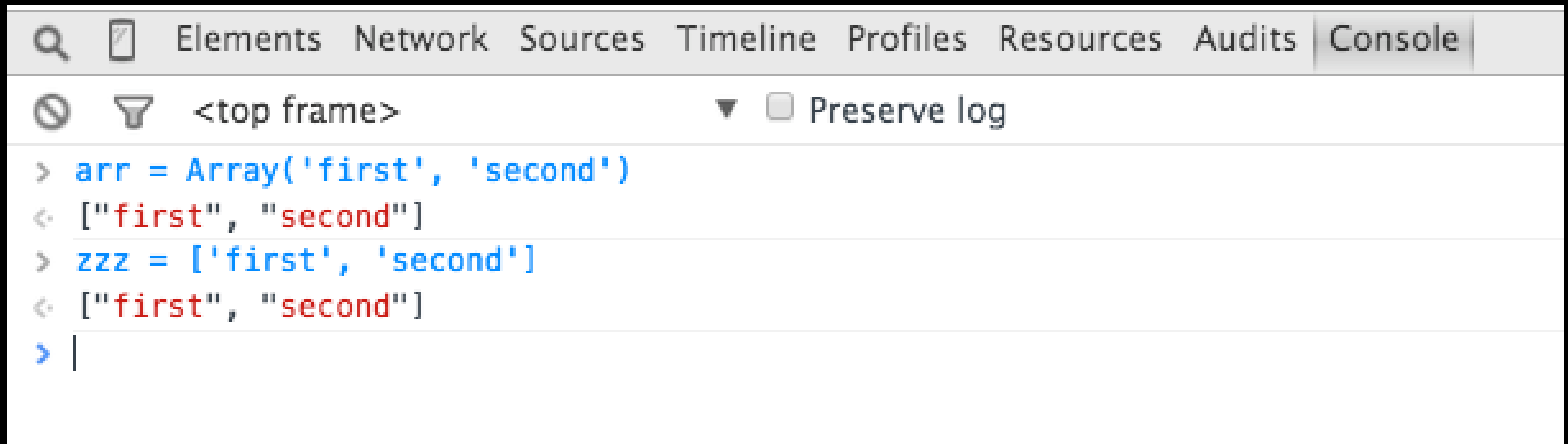
```
Elements Network Sources Timeline Profiles Re
<top frame> Preserve log
> a = ["x", "y", "z"]
< ["x", "y", "z"]
> b = {"name" : "chuck", "class" : "php-intro"}
< Object {name: "chuck", class: "php-intro"}
> a[0]
< "x"
> b['name']
< "chuck"
>
```


Linear Arrays

```
Elements Network Sources
<top frame>
> arr = Array()
< []
> arr.push('first')
< 1
> arr.push('second')
< 2
> arr
< ["first", "second"]
> |
```

```
Elements Network Sources
<top frame>
> arr = Array()
< []
> arr[0] = 'first'
< "first"
> arr[1] = 'second'
< "second"
> arr
< ["first", "second"]
>
```

Array Constructor / Constants



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays the following code and output:

```
> arr = Array('first', 'second')
< ["first", "second"]
> zzz = ['first', 'second']
< ["first", "second"]
> |
```

The console interface includes a search bar, a filter icon, and a 'Preserve log' checkbox. The output is color-coded: blue for code, red for strings, and black for array structures.

Control Structures

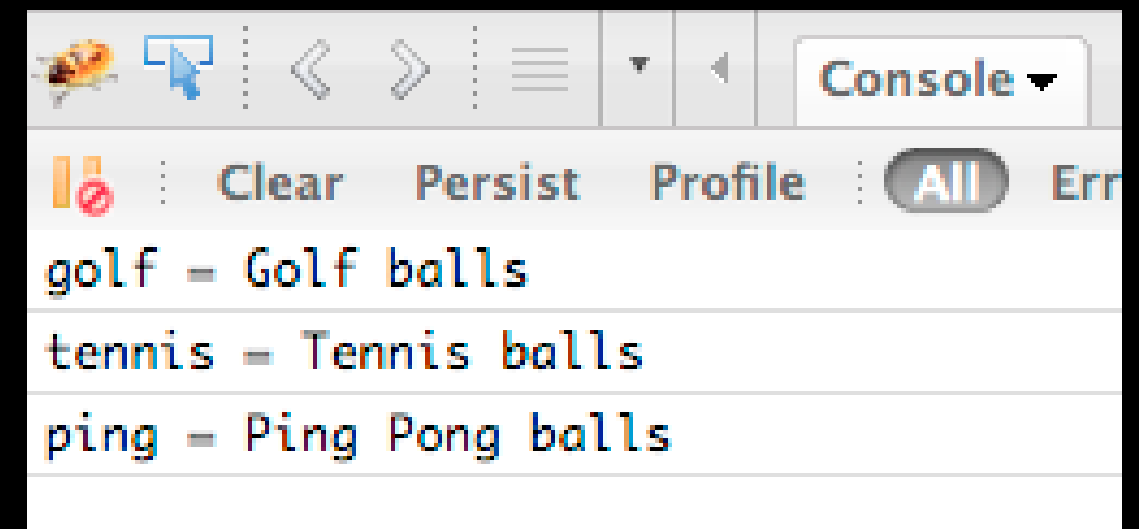
- We use curly braces for control structure.
- Whitespace / line ends do not matter.
- **if** statements are like PHP.
- **while** loops are like PHP.
- Counted **for** loops are like PHP – **for(; ;) ...**
- In loops, **break** and **continue** are like PHP.

Definite Loops (for)

[js-l2.htm](#)

```
balls = {"golf": "Golf balls",  
        "tennis": "Tennis balls",  
        "ping": "Ping Pong balls"};
```

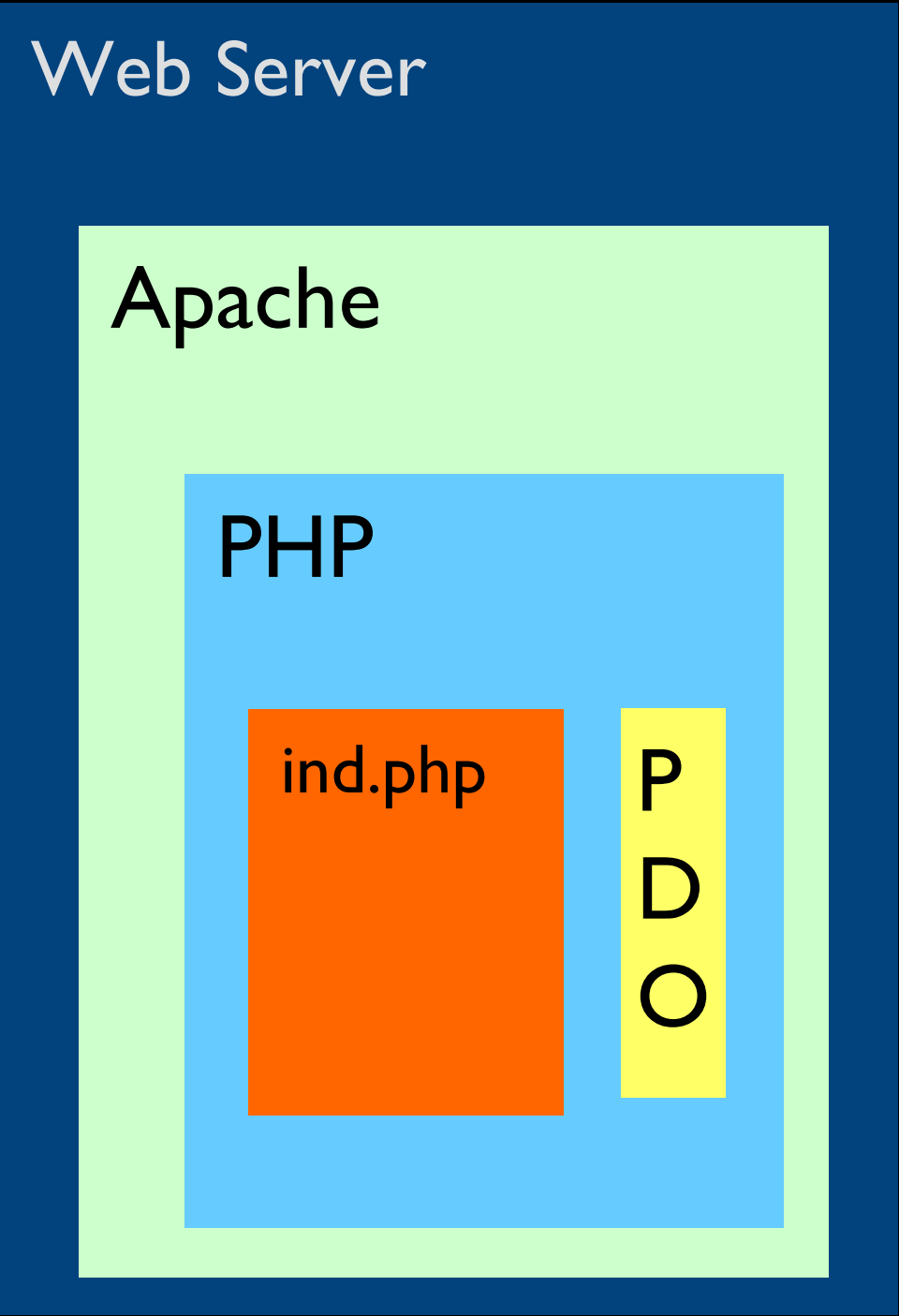
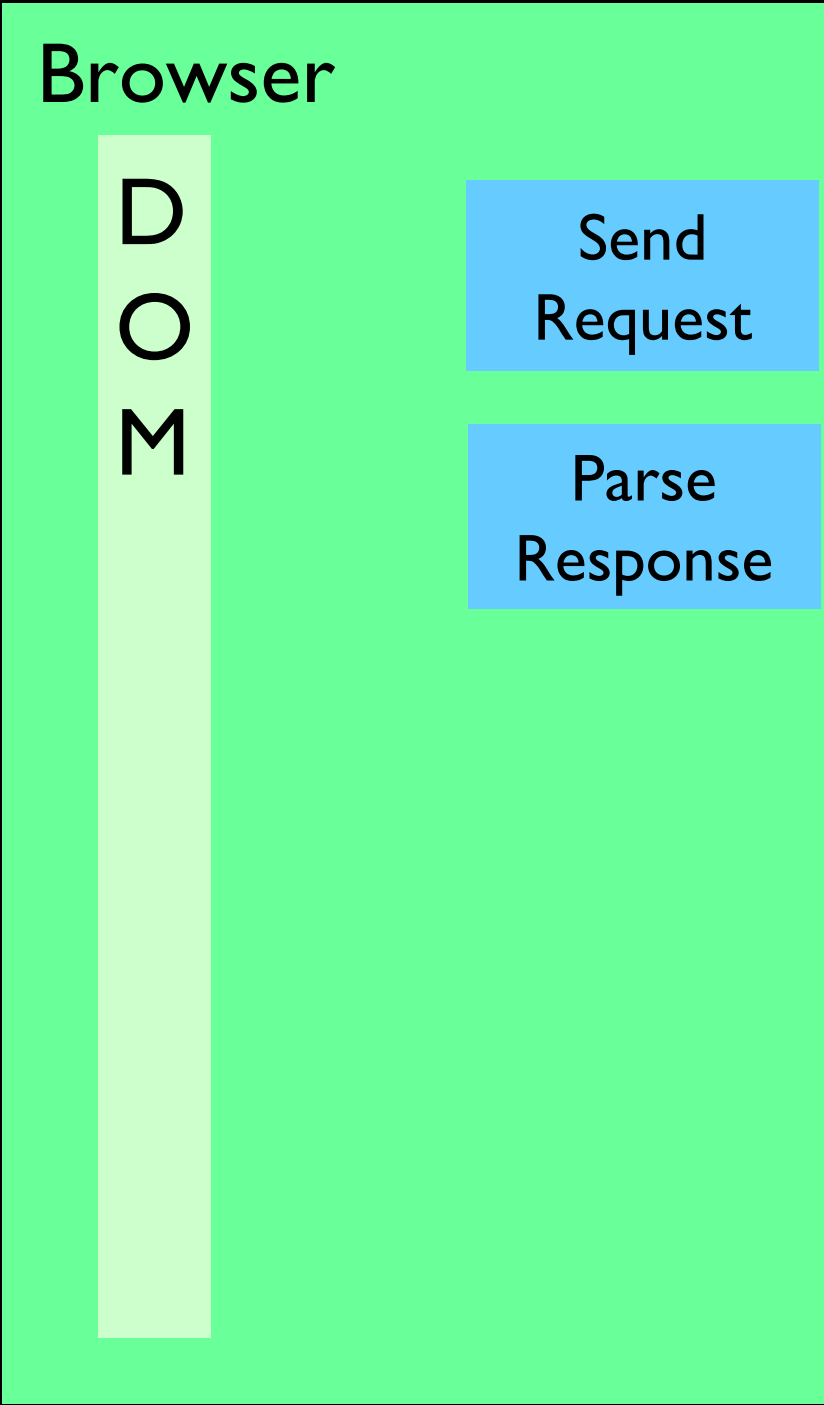
```
for (ball in balls) {  
    console.log(ball+' = '+balls[ball]);  
}
```



JavaScript - Document Object Model



Time



<http://www.wa4e.com/code/rrc/>

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

DOMs are Not Identical

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

[#SAK-21212] IE8: Clicking on Portfolio Layouts causes the page to load, then refresh and then no options display – Sakai

https://jira.sakaiproject.org/browse/SAK-21212

Sakai CLE / SAK-21212

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

EditAssignCommentMore Actions ▼Start ProgressResolve IssueWorkflow ▼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

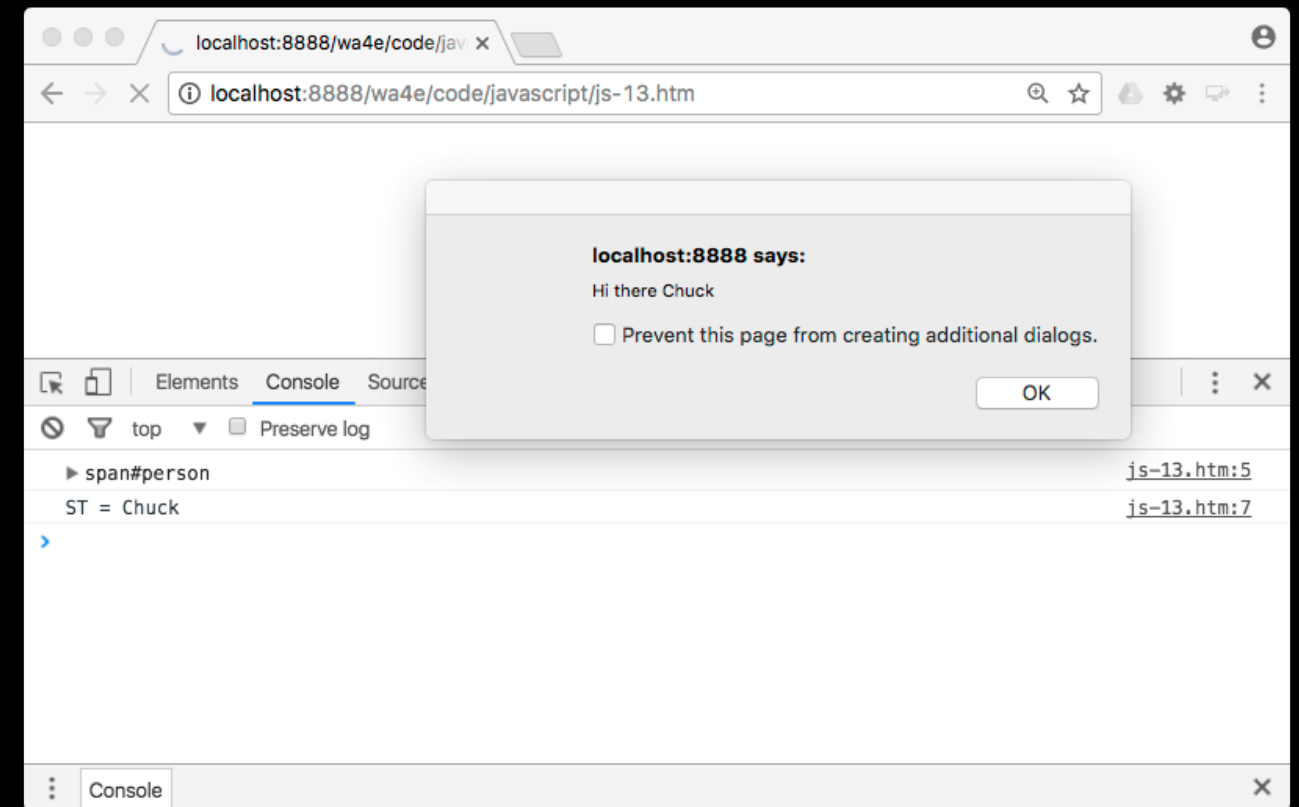
PortfolioLayouts_IE8.swf684 kB23-Sep-2011 06:54

Using getElementById()

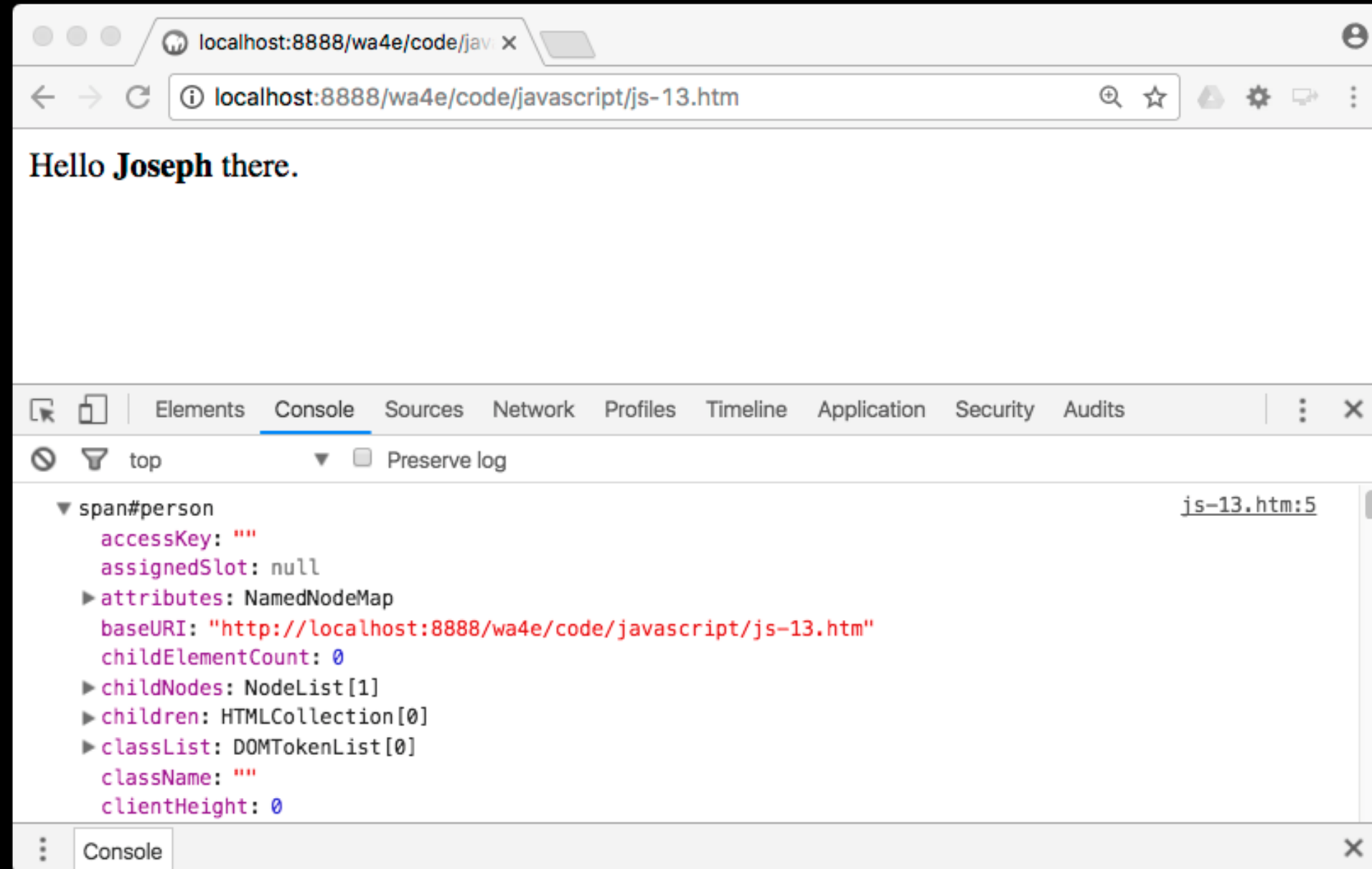
In order not to have to traverse each unique DOM, we use a function call that all browsers support. This allows us to bypass the DOM structure and jump to a particular tag within the DOM and manipulate that tag.

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

js-13.htm



js-13.htm



```
console.dir(document.getElementById('person'));
```

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

Updating the Browser Document

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

[Back](#) [Forth](#)

Hello **Stuff** there.

js-14.htm

[Back](#) [Forth](#)

Hello **BACK** there.

[Back](#) [Forth](#)

Hello **FORTH** there.

```
<p>
<a href="#" onclick="add();return false;">More</a>
</p>
<ul id="the-list">
<li>First Item</li>
</ul>
<script>
counter = 1;
console.log(document.getElementById('the-list'))
function add() {
    var x = document.createElement('li');
    x.className = "list-item";
    x.innerHTML = "The counter is "+counter;
    document.getElementById('the-list').appendChild(x);
    counter++;
}
</script>
```

Js-15.htm

More

- First Item
- The counter is 1
- The counter is 2
- The counter is 3

JQuery to the Rescue

- While the DOMs 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.
- With JQuery, instead of manipulating the DOM, we use JQuery functions and everything works much better...

<http://jquery.org>

Summary

- Using JavaScript
- Syntax errors
- Debugging
- Language features
- Global and local scope
- Arrays
- Control structures
- Document Object Model
- JQuery



Acknowledgements / Contributions

These slides are Copyright 2010- Charles R. Severance (www.dr-chuck.com) as part of www.wa4e.com and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

Insert new Contributors and Translators here including names and dates

Continue new Contributors and Translators here