Table of Contents

This slide deck consists of slides used in 5 lecture videos in Week 2. Below is a list of shortcut hyperlinks for you to jump into specific sections.

JavaScript

Dr. Charles Severance

www.dj4e.com

http://www.dj4e.com/code/javascript
http://www.dj4e.com/code/javascript.zip

About JavaScript

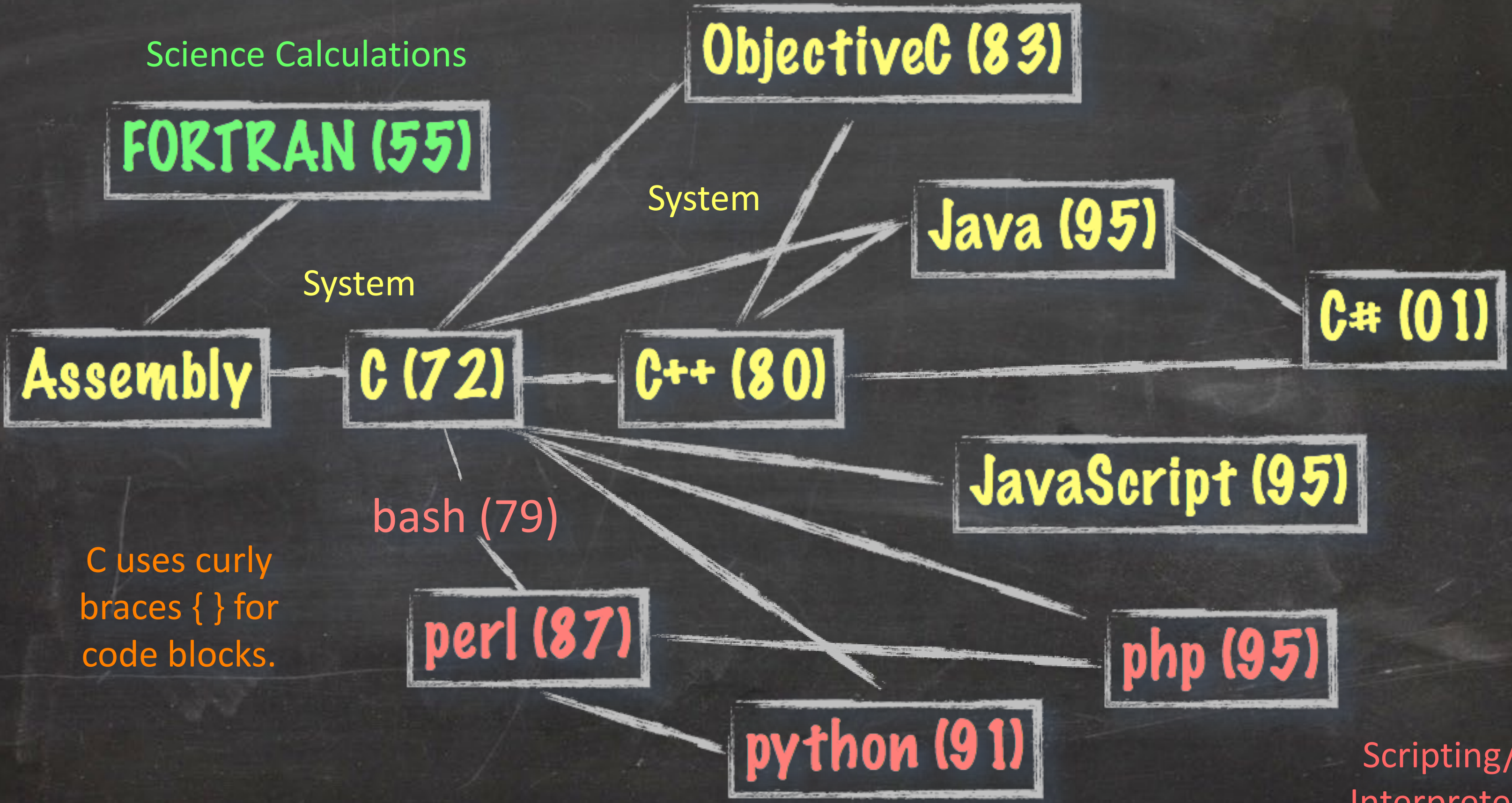
- In addition to HTML and CSS...
- Browsers have a powerful programming language called JavaScript that runs in the browser
- Actually not much like Java - more like Python with a C syntax
- Very powerful and flexible - we keep “discovering” new power

http://en.wikipedia.org/wiki/JavaScript

Inventing 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
https://www.youtube.com/watch?v=IPxQ9kEaF8c



C uses curly braces { } for code blocks.

Scripting/ Interpreted

http://en.wikipedia.org/wiki/History_of_programming_languages

Writing JavaScript

- Augment HTML using the Document Object Model (DOM) – "Vanilla JavaScript"
- Augment HTML using a library like JQuery
- Building an MVC Application in the Browser using Vue/React
- Building a server side application using Node / Express

Language Syntax (like C/Java)

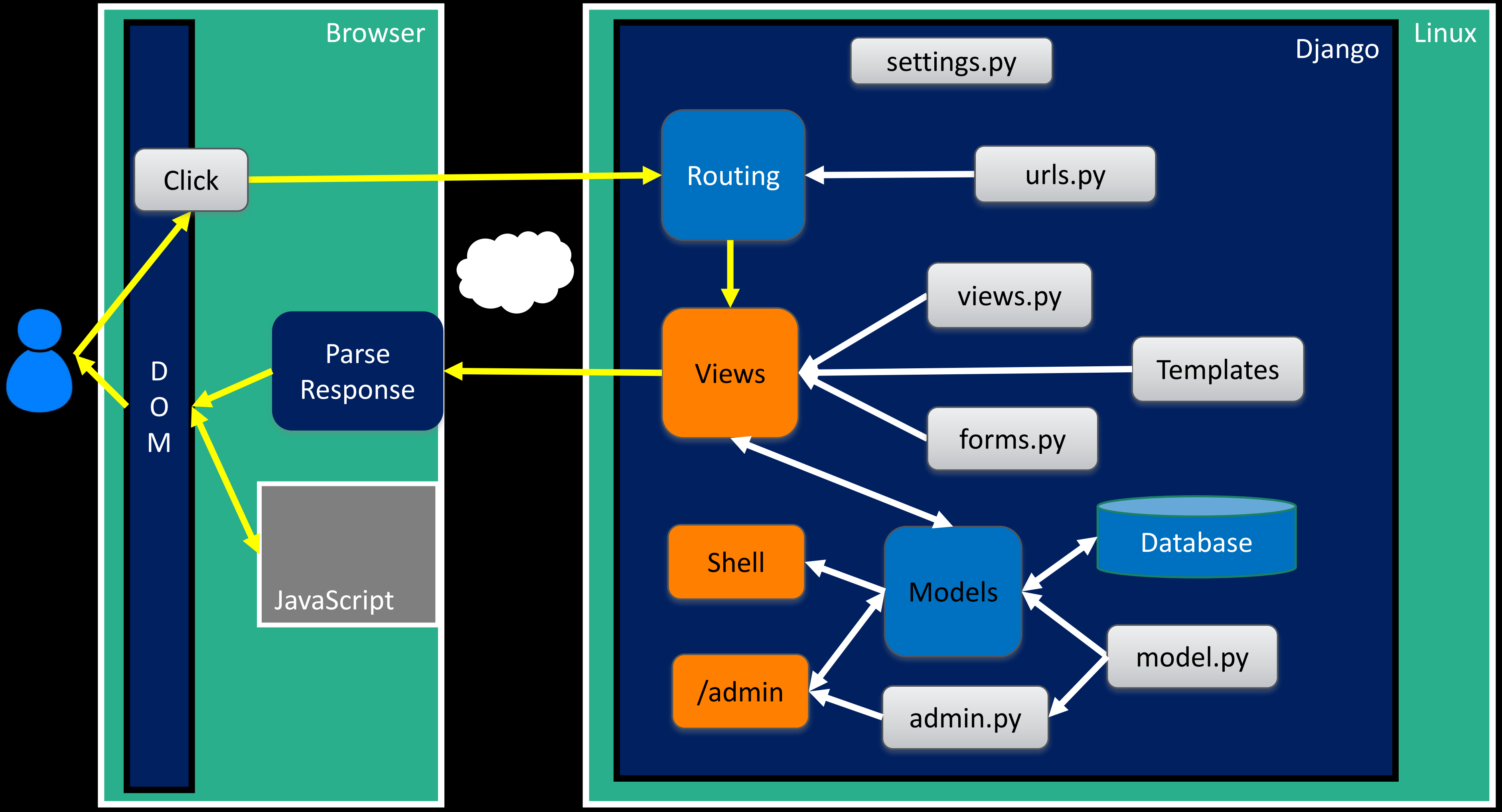
- Whitespace does not matter - spaces and new lines
- Begin and end of blocks are curly braces
- Statements must end in semicolons

```
function message()
{
   alert("This alert box was called with the onload event");
}
```

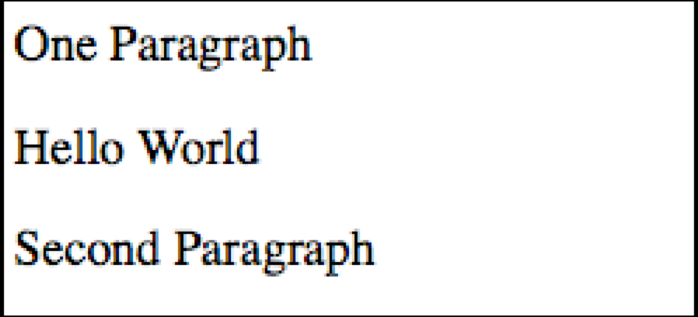
Working with JavaScript in the Browser

http://www.dj4e.com/code/javascript

http://www.dj4e.com/code/javascript.zip


Browser
D
O
M
Click
Parse Response
JavaScript
Linux
Django
settings.py
Routing
urls.py
views.py
Views
Templates
forms.py
Database
Shell
Models
/admin
model.py
admin.py

```
<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

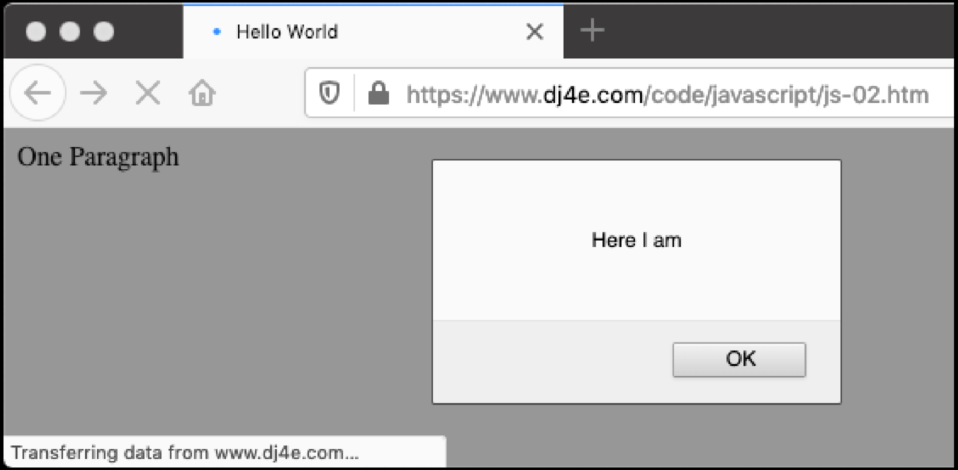
Second Paragraph

js-01.htm

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>
```



One Paragraph

Here I am

Transferring data from www.dj4e.com...

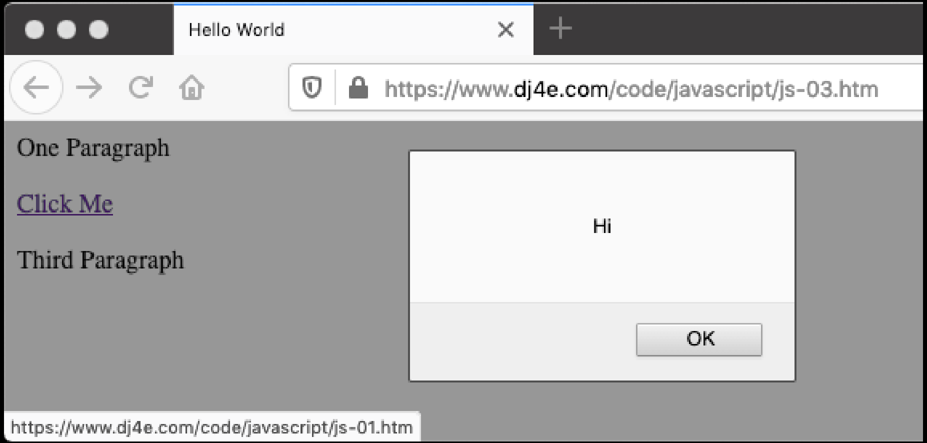
js-02.htm

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>
```



One Paragraph

Click Me

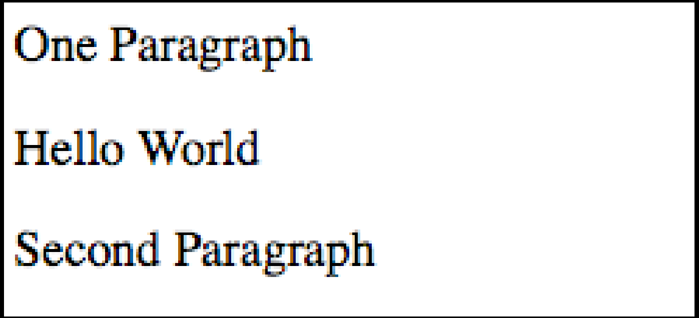
Third Paragraph

Hi

https://www.dj4e.com/code/javascript/js-01.htm

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>");
```

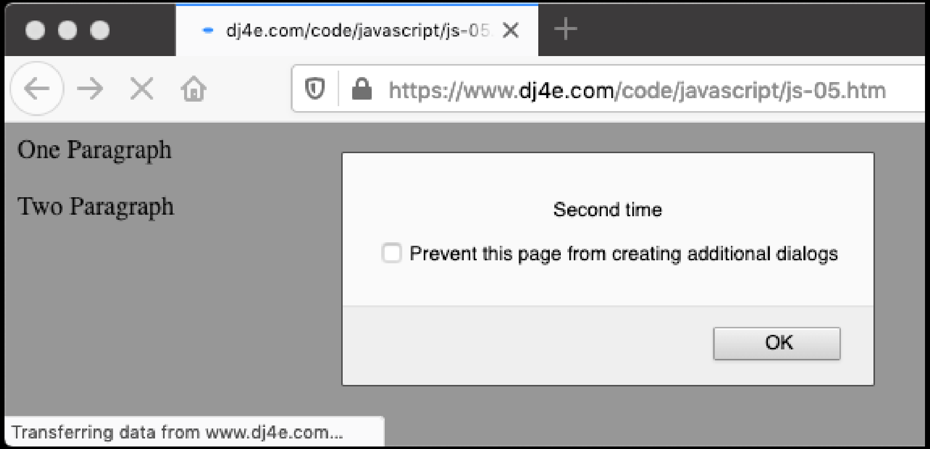
js-04.htm

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>
```


dj4e.com/code/javascript/js-05

https://www.dj4e.com/code/javascript/js-05.htm

One Paragraph

Two Paragraph

Second time

Prevent this page from creating additional dialogs

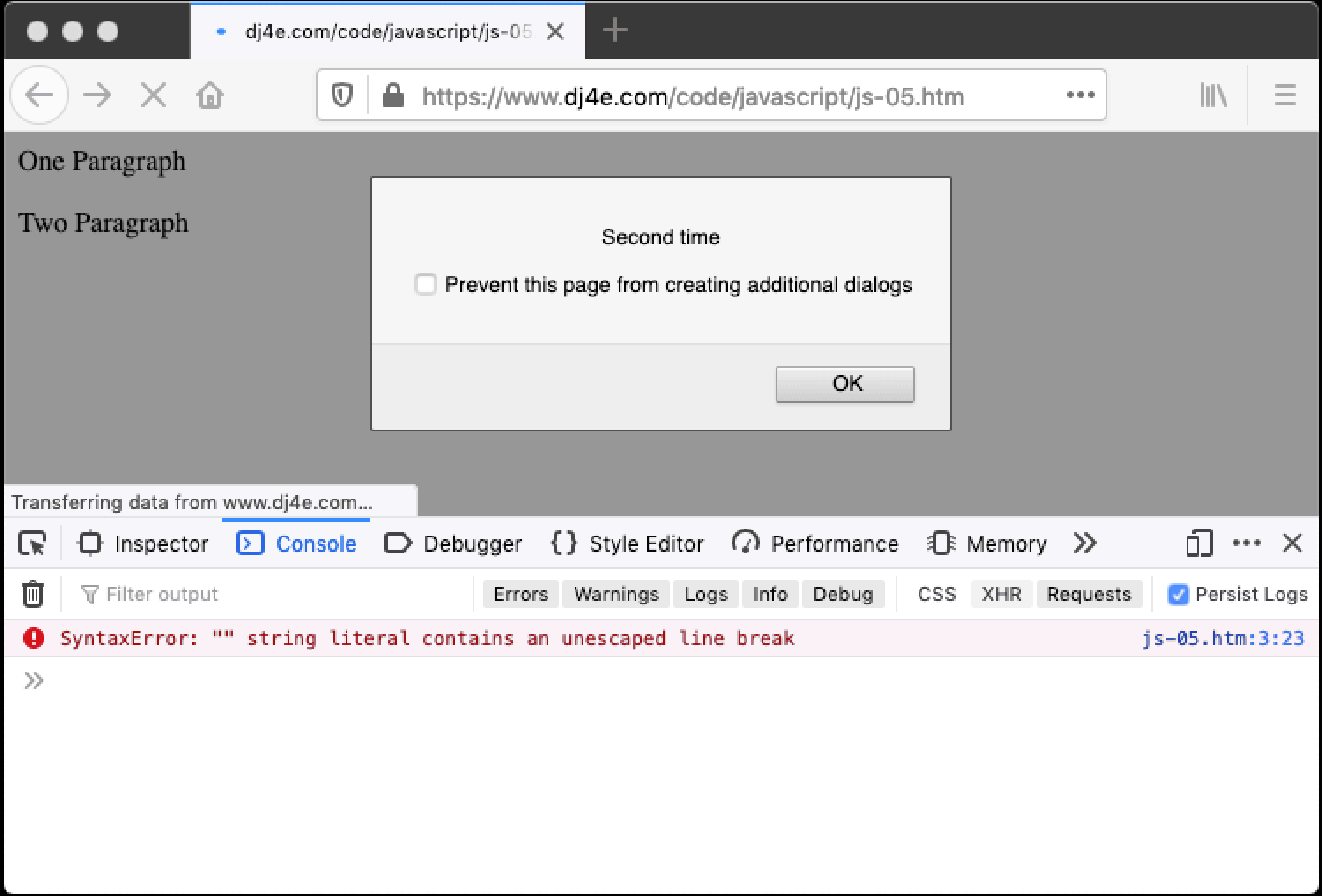
OK

Transferring data from www.dj4e.com...

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.

js-05.htm



One Paragraph

Two Paragraph

Second time

Prevent this page from creating additional dialogs

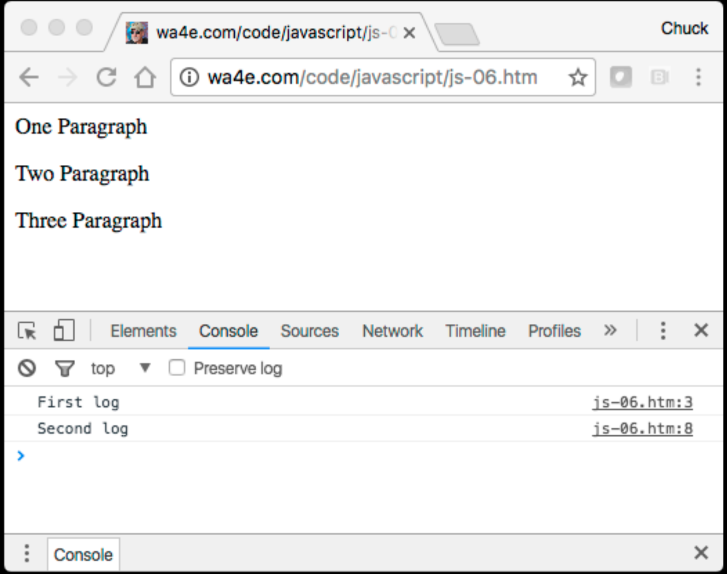
```
SyntaxError: "" string literal contains an unescaped line break     js-05.htm:3:23
```

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

js-06.htm

```
<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>
```



One Paragraph

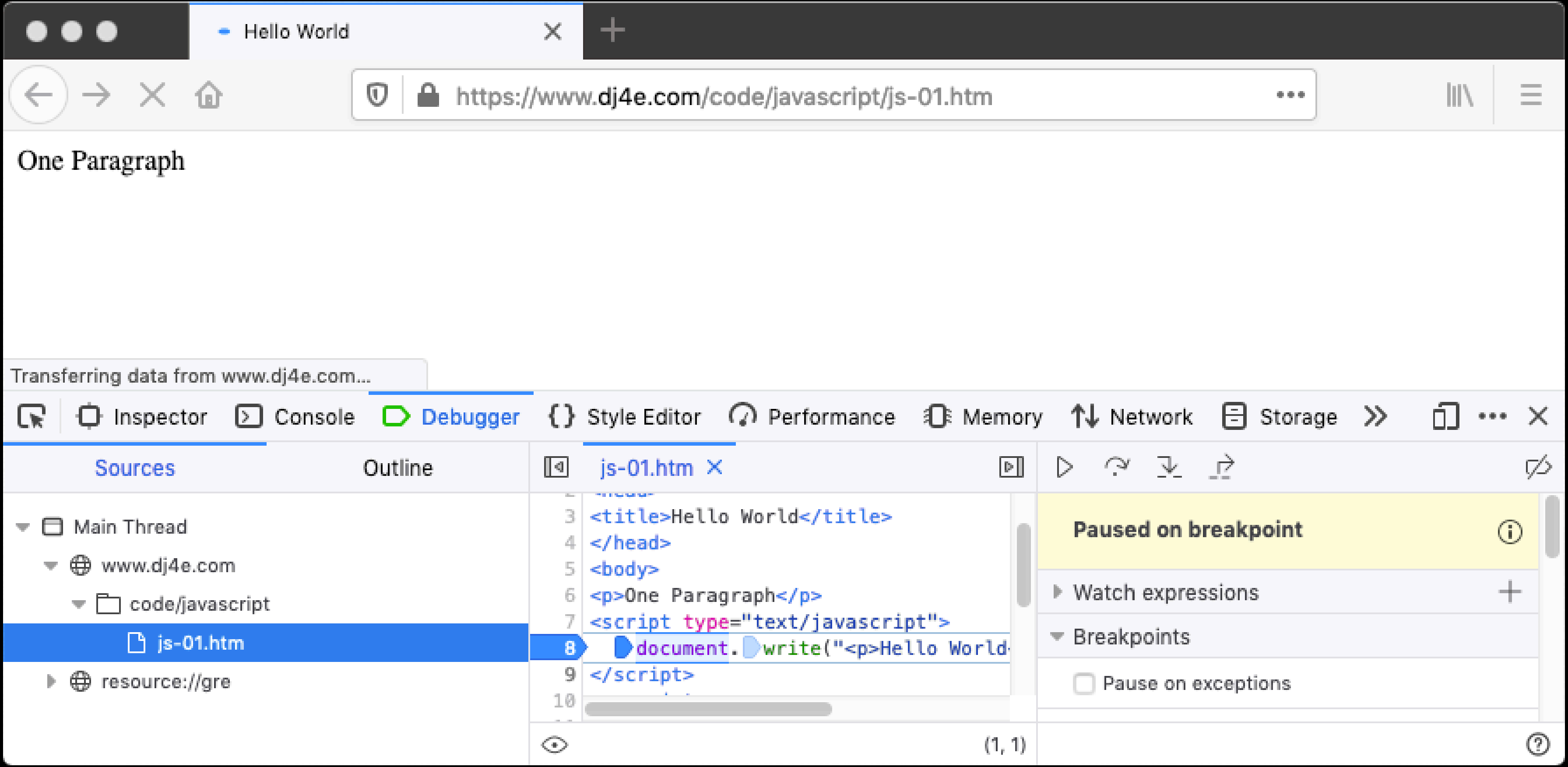
Two Paragraph

Three Paragraph

```
First log                                   js-06.htm:3
Second log                                  js-06.htm:8
```

Using the Debugger (Firefox)

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



One Paragraph

Transferring data from www.dj4e.com...

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

Paused on breakpoint

JavaScript Language

http://www.dj4e.com/code/javascript

http://www.dj4e.com/code/javascript.zip

Comments in JavaScript = Awesome

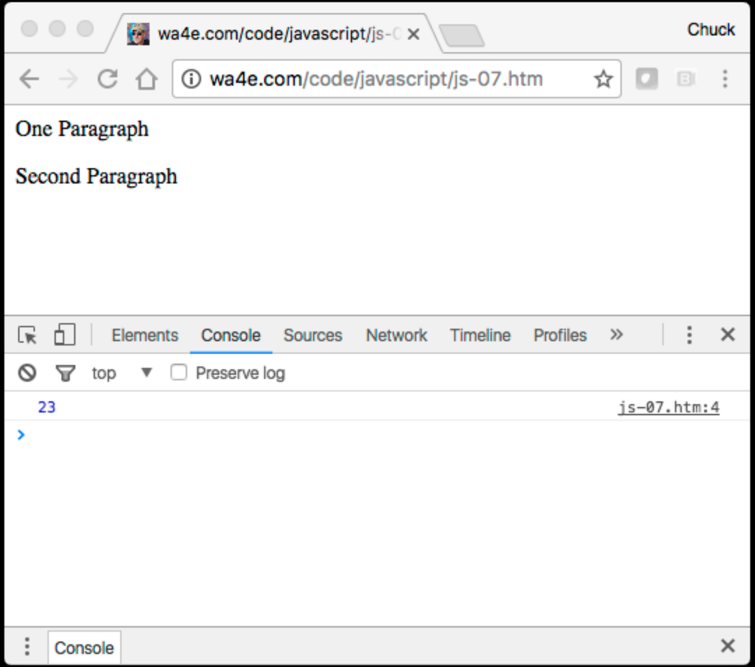
```
// This is a comment

/* 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.

js-07.htm

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


wa4e.com/code/javascript/js-07.htm

One Paragraph

Second Paragraph

```
23                                                    js-07.htm:4
>
```

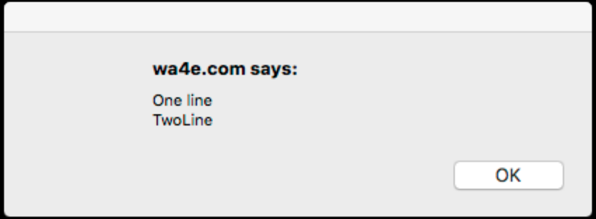
Variable Names

- 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 our minds a little sane.
- Character Escaping - done using the backslash character

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



```
wa4e.com says:
One line
TwoLine
```

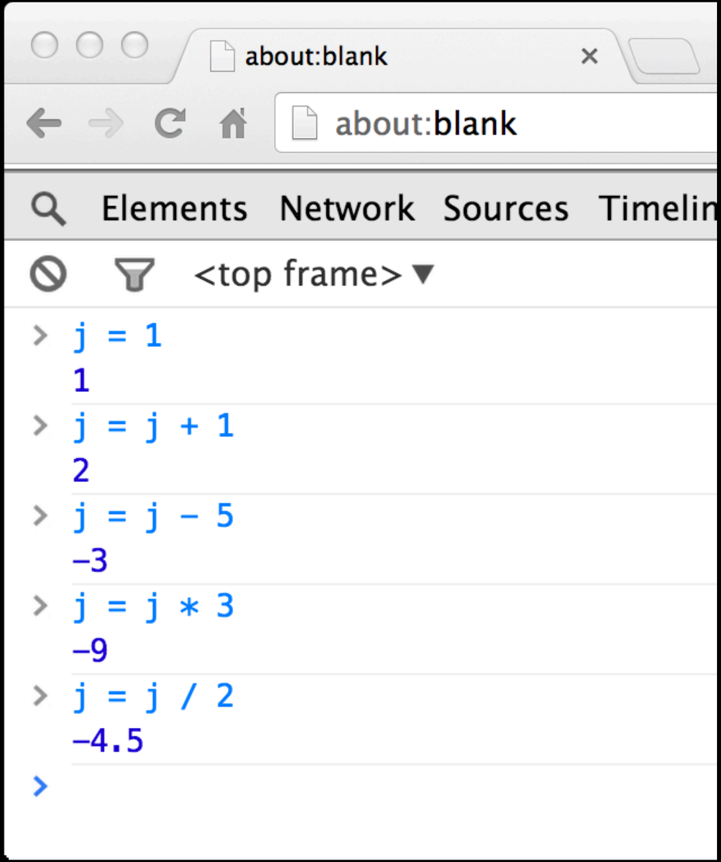
Numeric Constants

- Constant syntax is like most other languages
- Weirdness – One number type (no int or float)

```
>> x = 5/3;
1.6666666666666667
>> x = Math.trunc(x)
1
```

Operators

```
>> j = 1
1
>> j = j + 1
2
>> j = j - 5
-3
>> j = j * 5
-9
>> j = j / 5
-4.5
>> j = Math.trunc(j)
-4
```



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

More Operators

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

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

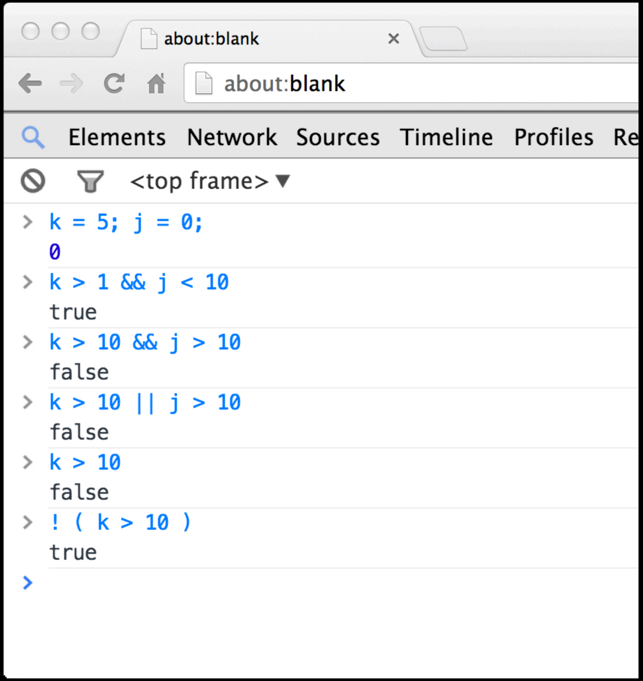
Comparison Operators

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

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

Logical Operators

```
>> 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
```



```
> 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

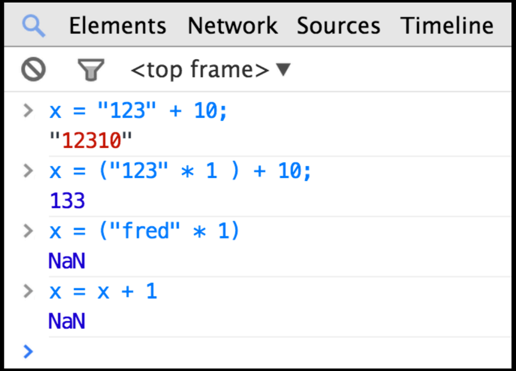
- JavaScript string concatenation is like Python except that it does implicit conversion from integer to string

```
>> 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. It does not trace back when arithmetic is confusing.

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



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

Variable Conversion

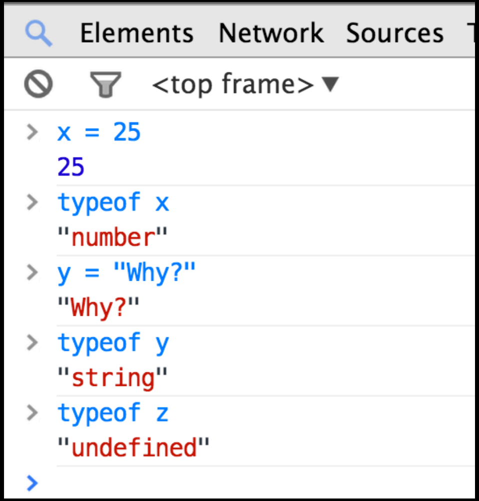
- If a string cannot be converted to a number, you end up with “Not a Number” or “NaN”. It is a value, but it is sticky - all operations with NaN as a operand end up with NaN.

```
>> x = "fred" + 1
NaN
>> isNaN(x)
true
>> x = x + 1
NaN
>> y = 42 / 0
Infinity
>> isNaN(y)
false
>> isInfinty(y)
false
```

Determining Type

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

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



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

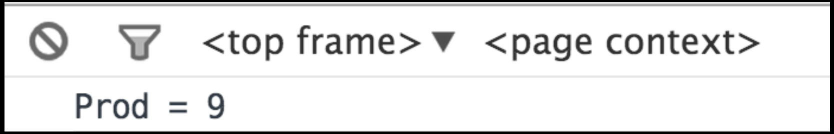
Functions and Arrays

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>
```



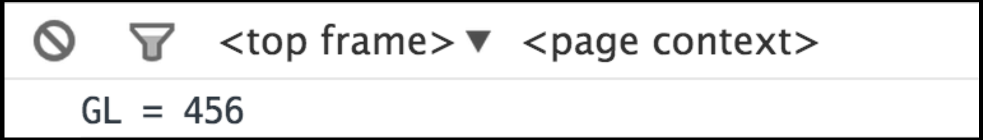
```
Prod = 9
```

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();
console.log("GL = "+gl);
</script>
```



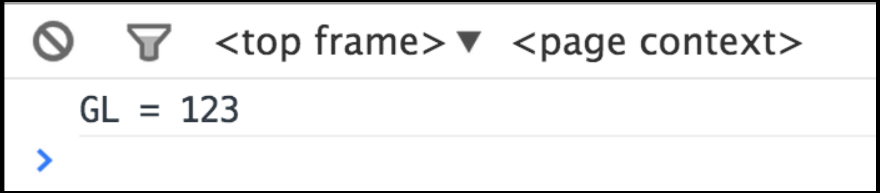
```
GL = 456
```

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();
console.log("GL = "+gl);
</script>
```



```
GL = 123
>
```

Arrays in JavaScript

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

```
>> a = ["x", "y", "z"]
["x", "y", "z"]
>> b = {"name":"chuck", "class":"dj4e"}
Object {"name":"chuck", "class":"dj4e"}
>> a[0]
"x"
>> b['name']
"chuck"
```

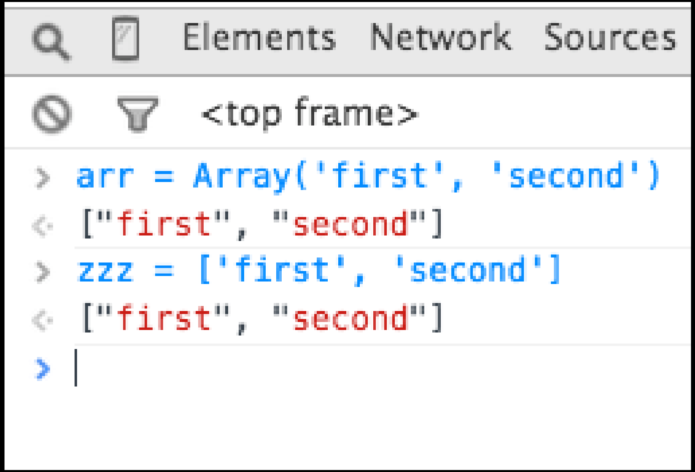
Linear Arrays

```
>> arr = Array()
[]
>> arr.push('first')
1
>> arr.push('second')
2
>> arr
["first", "second"]
```

```
>> arr = Array()
[]
>> arr[0] = 'first'
"first"
>> arr[1] = 'second'
"second"
>> arr
["first", "second"]
```

Array Constructor / Constants

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



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

Control Structures

Conditional - if

js-12.htm

- Logical operators (== != < > <= >= && || ! === !==)
- Curly braces

```
<script type="text/javascript">
    var ans = 42;
    if (ans == 42 ) {
        console.log("Hello world!");
    } else {
        console.log("Wrong answer");
    }
</script>
```

$\longrightarrow$

```
Hello World!
```

Multi-way Ifs

js-13.htm

```
var x = 7;

if ( x < 2 ) {
    console.log("Small");
} else if ( x < 10 ) {
    console.log("Medium");
} else {
    console.log("LARGE");
}

console.log("All done");
```

js-14.htm

```
var fuel = 10;
while (fuel > 1) {
    console.log("Vroom");
}
```

A **while** loop is a “zero-trip” loop with the test at the top before the first iteration starts. We hand construct the **iteration variable** to implement a counted loop.

```
var fuel = 10;
while (fuel > 1) {
    console.log("Vroom");
    fuel = fuel - 1;
}
```

Definite Loops (for)

js-15.htm

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

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

js-12.htm

js-16.htm

Loop runs while TRUE (top-test)

Before loop starts

Run after each iteration.

```
for(var count=1; count<=6; count++ ) {
  console.log(count, 'times 6 is', count * 6);
}
```

```
1 times 6 is 6
2 times 6 is 12
3 times 6 is 18
4 times 6 is 24
5 times 6 is 30
6 times 6 is 36
```

A for loop is the simplest way to construct a *counted* loop.

Breaking Out of a Loop

js-17.htm

- The break statement ends the current loop and jumps to the statement immediately following the loop.

```
for(var count=1; count<=600; count++ ) {
  if ( count == 5 ) break;
  console.log('Count:', count);
}
console.log("Done");
```

```
Count: 1
Count: 2
Count: 3
Count: 4
Done
```

Finishing an Iteration with continue

- The continue statement ends the current iteration and jumps to the top of the loop, and starts the next iteration.

```
for(var count=1; count<=10; count++ ) {
  if ( (count % 2) == 0 ) continue;
  console.log('Count:', count);
}
console.log("Done");
```

js-18.htm

```
Count: 1
Count: 3
Count: 5
Count: 7
Count: 9
Done
```

Try / Catch / Finally

js-19.htm

```
try {
    x = y + 1;
    console.log(x);
}
catch(erval) {
    console.log('Oops - Sorry');
    console.dir(erval);
}
finally {
    console.log('Always runs');
}
```

Summary

- Using JavaScript
- Syntax errors
- Debugging
- Language features
- Global and local scope
- Arrays
- Control structures

Acknowledgements / Contributions