AGENDA

- What is JavaScript?
- Tools
- Syntax
- Variables
- Functions (intro)
- Operators
- Objects
- Functions
- External JavaScript
- jQuery

LEARNING OBJECTIVES

- Describe JavaScript
- Create a JS Variable
- Describe 3 different types of variables
- Apply JS Operators to:
  - preform arithmetic
  - concatenate strings
  - compare variables
- Create a JS Object
- Access a JS Object's properties

WHAT IS JAVASCRIPT?

WHAT DOES GOOGLE SAY?

[JavaScript is] an object-oriented computer programming language commonly used to create interactive effects within web browsers.

WHAT DOES WIKIPEDIA SAY?

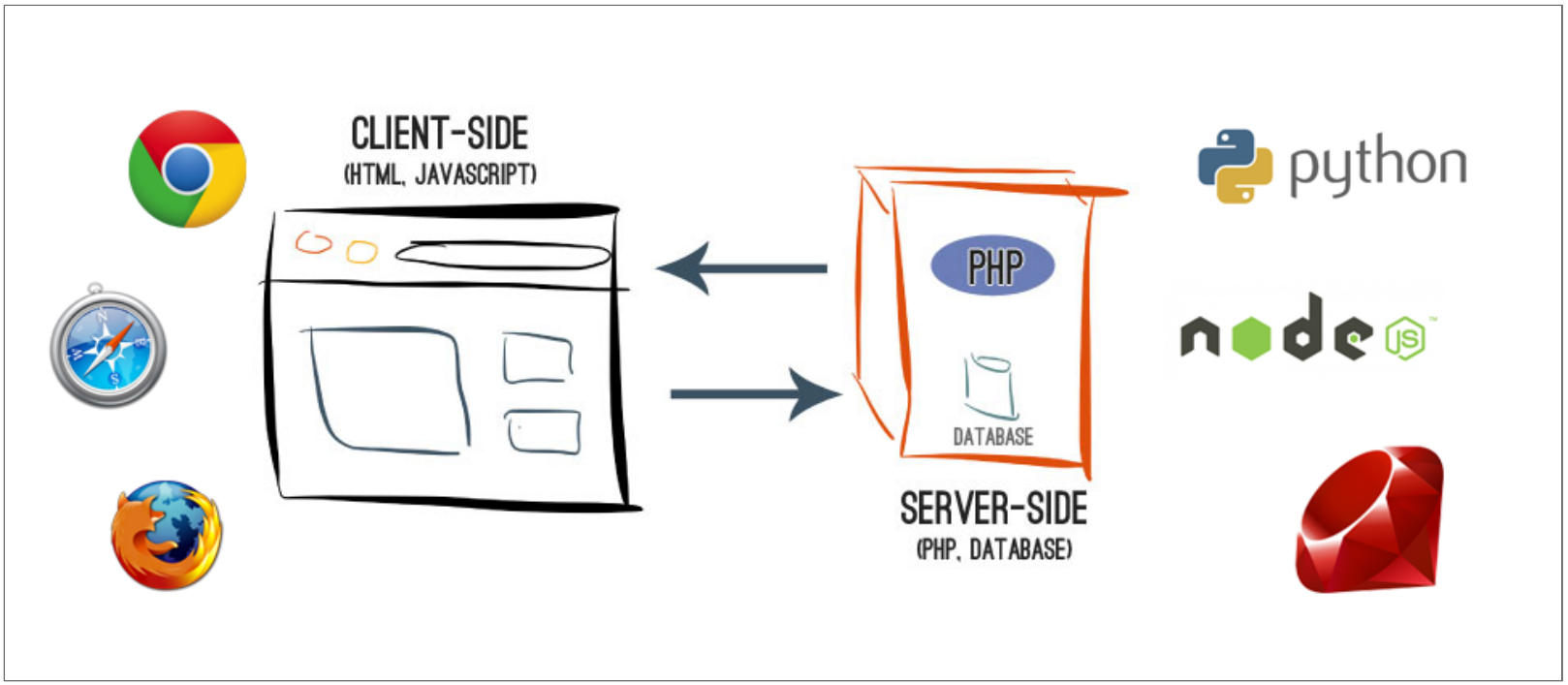
> *[JavaScript] is a dynamic computer programming language.*

PROGRAMMING LANGUAGES

A programming language is a formal constructed language designed to communicate instructions to a machine, particularly a computer.

CHOOSE YOUR OWN ADVENTURE

CLIENT SIDE VS. SERVER SIDE



A BRIEF HISTORY

- LiveScript was created by Brendan Eich in 10 days in May 1995

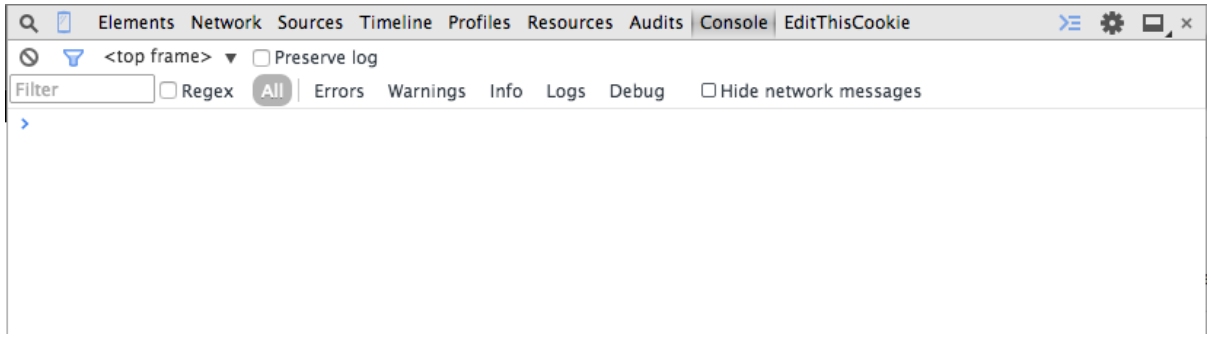
  - Eich worked at Netscape, who was trying to enhance Netscape 2.0
  - Netscape changed the name to JavaScript in a marketing ploy
  - IE adopted JavaScript in August 1996 when releasing v.3.0
  - Ecma Internation standardized JavaScript in 1999 (ECMAScript)

TOOLS

EXERCISE FILES

- **https://github.com/js-workshops/intro**
- Download files using git clone or the ‘Download Zip’ button
- After unzipping the files, copy them to a directory you want to work from (i.e. ~/Document, ~/Desktop)

CONSOLE



- Chrome Menu (top right hamburger button) -> Tools -> Developer Tools
- Mac shortcut: Cmd + Opt + i
- PC shortcut: F12, Ctrl + Shift + i

SYNTAX

RESERVED CHARACTERS

Programming languages are made of reserved characters. This allows you to tell the computer what to do without writting 010101000.

STRINGS

In JavaScript, you can declare strings with single or double quotation marks.

```
"String with double quotes"
'String with single quotes'
```

EXERCISE

File: 001.1-syntax.html

NUMBERS

In JavaScript, you declare numbers - integers and decimals.

```
42
3.14159
```

BOOLEANS

The Boolean data type is a data type, have two values, intended to represent true and false.

```
true
false
```

EXERCISE

File: 001.2-syntax.html

VARIABLES

*You use variables as **symbolic names** for values in your application. The names of variables, called **identifiers**, conform to certain rules.*

MEMORY

The variable name you create is a pointer to that data stored in memory.

VARIABLE RULES

- Must start with a letter, underscore (_), or dollar sign($)
- Subsequent letters can also be digits (0-9)
- Variable names are case sensitive

DECLARING VARIABLES

Using the *var* keyword, we are able to declare variables to the current scope.

```
var x = 18;
```

STATEMENT

A simple variable declaration has two ***operands***, a ***left operand*** and a ***right operand***. The ***left operand*** is on the left side of the equal sign, and the ***right operand*** is on the right side.

```
var y = "string";
```

SEMI-COLONS

Statements need to end with a semi-colon.

```
var z = true;
```

VARIABLE TYPES

Primitive Variables are the most common data types:

Type	**Example**
Number	42, 3.14159
Boolean	true / false
Strings	"howdy"
undefined	value is undefined

EXERCISE

File: 002-variables.html

FUNCTIONS (INTRO)

W3SCHOOLS

A JavaScript function is a block of code designed to perform a particular task. A JavaScript function is executed when "something" invokes it (calls it).

INVOKE

to make use of (a law, a right, etc.)

INVOKE (JS)

To initiate a block of code, that would otherwise not be executed.

BASIC FUNCTION

```
var myFunction = function() {
  // this is a function
  // your code block goes in between the curly brackets {}
}
```

FUNCTION PARAMETER(S)

```
var myFunction = function(myParameter) {
  // this is a function
  // your code block goes in between the curly brackets {}
  // you can access the parameter by it's name above:
  // myParameter in this case
  console.log(myParameter);
}
```

EXERCISE

File: 003-functions.html

OPERATORS

WHAT IS A JAVASCRIPT OPERATOR?

*JavaScript operators are **symbols** that can be used to assign values, compare values, preform arithmetic, test logic, manipulate strings, and more.*

TYPES OF OPERATORS

- Arithmetic operators
- Assignment operators
- String operators

ARITHMETIC OPERATORS

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

ARITHMETIC EXAMPLES

```
var x = 3;       // assigns the value 3 to x
x = x + 2;       // assigns the value 5 to x (3 + 2)
x = x - 3;       // assigns the value 2 to x (5 - 3)
x = x * 12;      // assigns the value 24 to x (2 * 12)
x = x / 2;       // assigns the value 12 to x (24 / 2)
x = x % 5;       // assigns the value 2 to x (12 % 5)
x = x++;         // assigns the value 3 to x (x + 1)
x = x--;         // assigns the value 2 to x (x - 1)
```

ARITHMETIC PARENTHESIS

Parenthesis can be used to indicate which part of an equation you want to evaluate first.

```
var x = 2 * (100 / 5);   // assigns the value 40 to x
```

```
var x = (100 / 5) * 2;   // assigns the value 40 to x
```

ASSIGNMENT OPERATORS

Shorthand	Meaning
x += y	x = x + y
x -= y	x = x - y
x *= y	x = x * y
x /= y	x = x / y
x %= y	x = x % y

ASSIGNMENT EXAMPLES

```
var x = 3;   // assigns the value 3 to x
x = x + 2;   // assigns the value 5 to x (3 + 2)
x = x - 3;   // assigns the value 2 to x (5 - 3)
x = x * 12;  // assigns the value 24 to x (2 * 12)
x = x / 2;   // assigns the value 12 to x (24 / 2)
x = x % 5;   // assigns the value 2 to x (12 % 5)
x = x++;     // assigns the value 3 to x (x + 1)
x = x--;     // assigns the value 2 to x (x - 1)
```

```
var x = 3;   // assigns the value 3 to x
x += 2;      // assigns the value 5 to x (3 + 2)
x -= 3;      // assigns the value 2 to x (5 - 3)
x *= 12;     // assigns the value 24 to x (2 * 12)
x /= 2;      // assigns the value 12 to x (24 / 2)
x %= 5;      // assigns the value 2 to x (12 % 5)
x = x++;     // assigns the value 3 to x (x + 1)
x = x--;     // assigns the value 2 to x (x - 1)
```

EXERCISE

File: 004.1-operators.html

STRING OPERATORS

The + symbol is used as the addition operator with numbers, but it become the concatenation operator with strings. It combines two string values together.

```
var x = "two strings " + "becomes one";
// assigns the value "two strings become one" to x
```

EXERCISE

File: 004.2-operators.html

COMPARISON OPERATORS

*A **comparison operator** compares its operands and returns a logical value based on whether the **comparison is true**.*

EQUAL AND NOT EQUAL

Operator	Description
Equal (==)	Returns true if the operands are equal
Not equal (!=)	Returns true if the operands are not equal

EQUAL AND NOT EQUAL (EXAMPLE)

```
var x = 2;   // assigns the value 2 to x
var y = 2;   // assigns the value 2 to y
var z = 3;   // assigns the value 3 to z
x == y       // returns true
x == z       // returns false
x != y       // returns false
x != z       // returns true
```

STRICT EQUAL AND NOT EQUAL

Operator	Description
Strict equal (===)	Returns true if the operands are equal and of the same type
Strict not equal (!==)	Returns true if the operands are not equal and/or not of the same type

STRICT EQUAL AND NOT EQUAL (EXAMPLE)

```
var x = 2;     // assigns the number 2 to x
var y = 2;     // assigns the number 2 to y
var z = '2';   // assigns the string 3 to z
x === y        // returns true
x === z        // returns false
x !== y        // returns false
x !== z        // returns true
```

GREATER THAN / LESS THAN

Operator	Description
Greater than (>)	Returns true if the left operand is greater than the right operand
Greater than or equal (>=)	Returns true if the left operand is greater than or equal to the right operand
Less than (<)	Returns true if the left operand is less than the right operand
Less than or equal (<=)	Returns true if the left operand is less than or equal to the right operand

GREATER THAN / LESS THAN (EXAMPLE)

```
var x = '10';     // assigns the string 10 to x
var y = 2;        // assigns the number 2 to y
var z = 10;       // assigns the number 10 to z
x > y             // returns true
x < y             // returns false
x > z             // returns false
x >= y            // returns true
y >= x            // returns false
x <= z            // returns true
```

EXERCISE

File: 004.3-operators.html

OBJECTS

The Object constructor creates an object wrapper.

LAYMAN'S TERMS

A variable is a symbol for one specific values. An object is a symbol that can contain many variables.

SYNTAX

- Are created with curly brackets ({})
- Contain key/value pairs separated by a semi-colon (:)
- The key/value pairs are separated by a comma (,)

OBJECT EXAMPLE

```
var person = {
  firstName: 'Jane',
  lastName: 'Doe',
  age: 28
};
```

ACCESSING AN OBJECT'S PROPERTIES

```
var person.firstName;     // returns 'Jane'
```

```
var person['firstName'];  // returns 'Jane'
```

EMBEDDED JAVASCRIPT

Similar to CSS, you can add JavaScript to your HTML file.

```
<script type="text/javascript">
  // some javascript here
</script>
```

EXERCISE

File: 005-objects.html

FUNCTIONS

REVIEW W3SCHOOLS

*A **JavaScript function** is a block of code designed to perform a particular task. A JavaScript function is executed when "something" **invokes it** (calls it).*

REVIEW INVOKE (JS)

To initiate a block of code, that would otherwise not be executed.

SYNTAX

Global Function

Note that we will cover the difference between global and local variables in the next lesson.

- Are created with the keyword ***function***
- If creating a global function, you add the name after the ***function*** keyword
- The function name is followed with parenthesis ***()***

```
function myFunctionName()
```

SYNTAX

Local Function

- To create a local function, you use the ***var*** keyword and the statement character (=)
- The function name is the ***left operand*** of the statement
- The right operand is the ***function*** keyword followed by parenthesis ***()***

```
var myFunctionName = function()
```

SYNTAX

Function Declaration

- The function declaration is after the parenthesis ***()***
- The statement lives between curly brackets ***{}***
- After the closing curly bracket, you end the function with a semi-colon (;)

```
var myFunctionName = function() {
  // declaration here
};

function myFunctionName() {
  // declaration here
};
```

SYNTAX

Params

- Inside the parenthesis, after the ***function*** keyword, you can pass in paramaters ***(param)***
- Parameters are used to pass data into the function to reference in your declaration.

```
var myAlert = function(msg) {
  alert("ALERT: " + msg);
}

myAlert("Are you confused yet?");
// creates the alert: ALERT: Are you confused yet?
myAlert("Don't worry. It will make sense :)");
// creates the alert: ALERT: Don't worry. It will make sense :)
```

SYNTAX

Multiple Params

- Multiple parameters can be passed inside the parenthesis after the ***function*** keyword.
- Each parameter inside the parenthesis is separated with a comma.

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

multiply(2, 4);
// returns the number 8
```

FUNCTIONS AND VARS

Functions can be assigned to variables

```
function logName(myName) {
  console.log(myName);
};

logName('John'); // logs 'John' to the console
```

```
var logName = function(myName) {
  console.log(myName);
};

logName('Jane'); // logs 'Jane' to the console
```

ENCAPSULATE LOGIC

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

var z = multiply(2, 4); // assigns the number 8 to z
```

EXERCISE

File: 006-functions.html

EXTERNAL JAVASCRIPT

INLINE JAVASCRIPT

Inline JavaScript is added directly to the DOM between a <script> and </script> tag. If your not using the HTML5 Document type, then you need to define a type on the opening script tag:

```
<script type="text/javascript"> // JavaScript goes here </script>
```

INLINE JAVASCRIPT USE CASES

Not a maintainable format. Use this only for code that should be fired immediately, and doesn't deserve it's own file (I.e. Google Analytics code).

EXTERNAL JAVASCRIPT

External JavaScript is JavaScript that lives in a separate file than the DOM, but is referenced by the DOM.

```
<script src="path/to/file.js"></script>
```

EXTERNAL JAVASCRIPT

Unlike CSS, which is added to the head, you typically add external JavaScript right before the closing body tag.

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="path/to/css" />
</head>
<body>
  <script src="path/to/file.js"></script>
</body>
</html>
```

EXTERNAL JAVASCRIPT USE CASES

- Maintainability
- Asynchronous loading
- Compiled files

EXERCISE

File: 007-external-js.html

JQUERY

WHAT IS JQUERY?

jQuery is a JavaScript library, or a collection of functions/methods and objects that you can utilize.

JQUERY DEFINITION

It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.

JQUERY MOTTO

“write less, do more”

CONTENT DILVERY NETWORK (CDN)

*A **content delivery network** or **content distribution network** (CDN) is a large distributed system of servers deployed in multiple data centers across the Internet. The goal of a CDN is to serve content to end-users with high availability and high performance.*

EXERCISE

File: 008.1-jquery.html

$ SYMBOL

jQuery, unless assigned otherwise, assigns its methods to the **$** symbol.

JQUERY SELECTOR

With jQuery assigned to the **$** symbol, here is how you select a DOM element.

```
<div id="myId"></div>
<span class="myClass"></span>
```

```
$("#myId")      // returns DOM element with the ID 'myId'
$(".myClass")   // returns DOM element with the class 'myClass'
```

EXERCISE

File: 008.2-jquery.html

JQUERY CHAINED METHODS

You can chain methods to the jQuery object that is returned. For example, after selecting the DOM elements, you might want to preform:

```
$("#myId").addClass('active');
// adds the class 'active' to the DOM element with the ID 'myId'
$(".myClass").hide();
// hides the DOM element with the class 'myClass'
```

JQUERY DOC READY

The browser loads some content synchronously and other content asynchronously. This can cause a problem with our JavaScript, as a lot of it is dependent on the state of the DOM and other assets.

JQUERY DOC READY

Good News! jQuery has a method to handle this.

> *A page can't be manipulated safely until the document is ‘ready.' jQuery detects this state of readiness for you.*

JQUERY DOC READY EXAMPLE

```
$(document).ready(function() {
  console.log("Ready!");
});
```

The console log statement is fired after the Document has loaded.
http://learn.jquery.com/using-jquery-core/document-ready/

EXERCISE

File: 008.3-jquery.html

JQUERY EVENTS

Events are actions that happen in the browser. jQuery allows you to bind functions to events that happens in the invent.

Event types:

- ready
- hover
- click

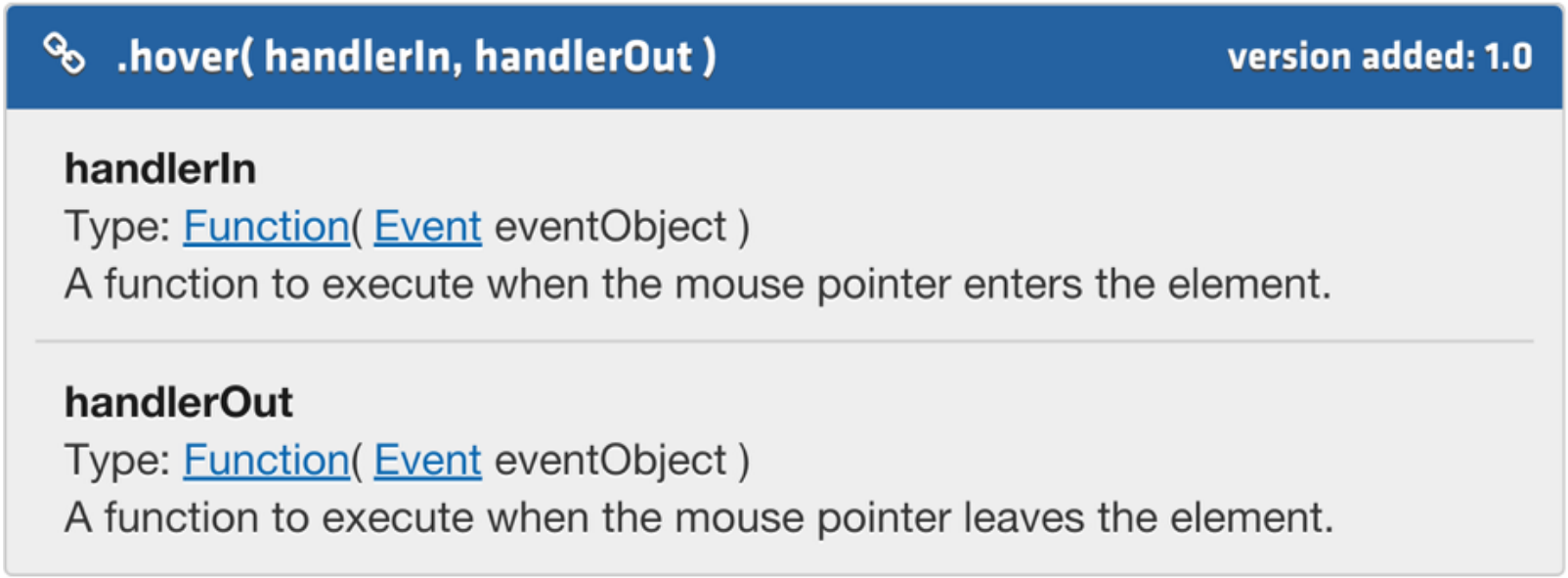
WHY EVENTS?

Up to this point, we have manually invoked functions. With jQuery events, we can bind functions to actions that occur in the browser.

Examples:

- Once the Document has loaded, preload popular assets to improve latency
- On hover over button, show tooltip with additional info
- On click of button, show contact form in modal

JQUERY HOVER (EVENT)



FUNCTIONS AS VARIABLES

```
$("mySelector").hover(function() {
    // hover in statement
  }, function() {
    // hover out statement
  }
);
```

```
var hoverIn = function() {
  // hover in statement
};

var hoverOut = function() {
  // hover out statement
};

$("mySelector").hover( hoverIn, hoverOut );
```

EXERCISE

File: 008.4-jquery.html

CLICK EVENT

.click(handler) — version added: 1.0

handler
Type: Function(Event eventObject)
A function to execute each time the event is triggered.

.click([eventData], handler) — version added: 1.4.3

eventData
Type: Anything
An object containing data that will be passed to the event handler.

handler
Type: Function(Event eventObject)
A function to execute each time the event is triggered.

CLICK EVENT (EXAMPLE)

```
$("mySelector").click(function(e) {
  // e is the click data
  // this prevents the default click event from occuring
  e.preventDefault();
  // click occured
});
```

EXERCISE

File: 008.5-jquery.html

JQUERY ANIMATIONS

jQuery animations allow you to add visual effects to the browser with JavaScript. There are a number of jQuery animations, but will go over a commonly used one called show()

JQUERY SHOW

.show() — version added: 1.0

This signature does not accept any arguments.

.show([duration] [, complete]) — version added: 1.0

duration (default: `400`)
Type: Number or String
A string or number determining how long the animation will run.

complete
Type: Function()
A function to call once the animation is complete.

JQUERY SHOW (EXAMPLE)

```
$("mySelector").show(300)
```

- The first parameter is the duration of the animation (i.e. how long until the element is showed).
- An option second parameter can passed it. It has to be a function, or callback for when the animation is complete.

EXERCISE

File: 008.6-jquery.html

LEARNING OBJECTIVES

- Describe JavaScript
- Create a JS Variable
- Describe 3 different types of variables
- Apply JS Operators to:
  - preform arithmetic
  - concatenate strings
  - compare variables
- Create a JS Object
- Access a JS Object's properties