

THE // FLATIRON SCHOOL

Today we will be learning JabbaScript...

2



Just kidding, actually we will be learning...

3

JahvaScript



but seriously, we will be learning...

4

da da da

da da da da



JavaScript Fundamentals

- Variables
- Constants
- Data Types
- Operators
- Conditional Statements
- Browser Messages
- Where to put JavaScript
- The DOM

Variables & Data Types

JavaScript Variables

7

A variable is like a bucket that we can store some data inside and then later change the data it stores.



JavaScript Variables

8

//Let's declare some variables

var x;

//var is a keyword that declares a variable

//naming variables: start with letter a-z A-Z remaining 0-9 a-z A-Z, you can also use _ never use spaces ' ' or or illegal characters (! % \$?)

Most use camelCase or snake_case

JavaScript Variables

9

//Let's assign a value in x (remember we said var x; previously)

x = 10; //I can refer directly to x and change its value using = symbol

//Declare and assign a value simultaneously.

var y = 20;

//Declare multiple variables at once using comma separation.

var a = 1,

 b = 2,

 c = 3;

JavaScript Variables

10

//perform math on variables

var x = 10,

y = 20,

myMath = x + y;

//What does myMath equal?

x = 5;

myMath = x + y;

//What does myMath equal now?

JavaScript Constants

11

A constant is like a bucket with a lid that is glued shut, and once you store a value inside it, you cannot change that value again.



JavaScript Constants

12

```
const STAY_THE_SAME = 1;
```

//just like var, const is a keyword that declares a constant

```
STAY_THE_SAME = 2;
```

//How much does STAY_THE_SAME equal now?

//Still equals 1... Always equals 1...

What are the types of values
I can store inside of variables?

Numbers: integers (whole numbers) like **2**, floats (decimals) like **3.56789**

Strings: (text) like **'hello'** or **"hello"**

Boolean: **true** or **false**

null (empty)

undefined (not yet defined)

What if I would like to check
what type a value is?

JavaScript Checking Data Type

16

```
const STAY_THE_SAME = 1;
```

```
console.log(typeof STAY_THE_SAME); //reports type of number
```

//typeof is a keyword that allows us to check the data type.

```
var hamburger = 'Yumm';
```

```
console.log(typeof hamburger); //reports type of string
```

Concatenation (attaching two parts into one)

17

```
var combineStuff = '10' + 5; //string concatenation  
console.log('Adding a string to a number does this: '+combineStuff+'  
weird!');
```

```
//prints Adding a string to a number does this: 105 weird!
```

//JavaScript uses + symbol for math when surrounded by numbers, but when any content is a string, it will concatenate the text. If Boolean and a number: true = 1 and false = 0. So true + 2 = 3... weird huh? This can produce unexpected results unless we are careful how we use + symbol. It's also good idea to be aware what data types are involved.

DATA TYPE Conversion

18

```
var stringy = 12 + '';
```

```
//concatenating empty string changes number to a string.
```

```
console.log(typeof stringy); //prints "string"
```

```
combineStuff = parseInt('10') + 5;
```

```
//parseInt changes a string to a whole number.
```

```
console.log(combineStuff); //prints 15
```

DATA TYPE Conversion

19

//what the heck happens if I try converting text to a number?

```
var userName = parseInt('bob');
```

```
console.log(userName); //prints NaN
```

//NaN stands for Not a Number!

DATA TYPE Conversion

20

//what if I want to convert to a floating number instead of an integer?

```
var gallonsGas = parseFloat('10.25');
```

```
Console.log(gallonsGas); //prints 10.25 as number
```

//parseInt stops at whole number and ignores characters that are not numbers. 10xxxxxx same as 10.12345 becomes 10. Whereas parseFloat maintains decimal values.

Operators

Arithmetic Operators

22

operator	name	description	usage
-	Negation	Subtracts	$4 - 3 = 1$
+	Plus	Adds	$4 + 3 = 7$
*	Multiply	Multiplies	$3 * 2 = 6$
/	Divide	Divides	$12 / 2 = 6$
%	Modulus	Returns the remainder.	$12 \% 5 = 2$
++	Increment	Increases the value by 1	$x = 1; x++; x = 2$
--	Decrement	Decreases the value by 1	$x = 1; x--; x = 0$

COMPARISON Operators

23

operator	name	description	example
==	Equal	Returns true if the operands match. Does not compare data type.	3 == var1
!=	Not equal	Returns true if the operands are not equal. Does not compare data type.	var1 != 4
===	Strict equal	Returns true if the operands are strictly including their data types.	3 === var1
!==	Strict not equal	Returns true if the operands are not equal and/or not of the same type.	var2 !== 3
>	Greater than	Returns true if the left operand is greater than the right operand.	var2 > var1
>=	Greater than or equal	Returns true if the left operand is greater than or equal to the right operand.	var2 >= var1
<	Less than	Returns true if the left operand is less than the right operand.	var1 < var2
<=	Less than or equal	Returns true if the left operand is less than or equal to the right operand.	var2 <= 5

Logical Operators

24

operator	name	description	usage
&&	Logical AND	Returns expr1 if it can be converted to false; otherwise, returns expr2. Thus, when used with Boolean values, && returns true if both operands are true; otherwise, returns false.	expr1 && expr2
	Logical OR	Returns expr1 if it can be converted to true; otherwise, returns expr2. Thus, when used with Boolean values, returns true if either operand is true; if both are false, returns false.	expr1 expr2
!	Logical NOT	Returns false if its single operand can be converted to true; otherwise, returns true.	!expr

operator	name	description	usage
&	Bitwise AND	Returns a one in each bit position for which the corresponding bits of both operands are ones.	a & b
	Bitwise OR	Returns a one in each bit position for which the corresponding bits of either but not both operands are ones.	a b
^	Bitwise XOR	Returns a one in each bit position for which the corresponding bits of either but not both operands are ones.	a ^ b
~	Bitwise NOT	Inverts the bits of its operand.	~ a
<<	Left shift	Shifts a in binary representation b (< 32) bits to the left, shifting in zeros from the right.	a << b
>>	Sign-propagating right shift	Shifts a in binary representation b (< 32) bits to the right, discarding bits shifted off.	a >> b
>>>	Zero-fill right shift	Shifts a in binary representation b (< 32) bits to the right, discarding bits shifted off, and shifting in zeros from the left.	a >>> b

ASSIGNMENT Operators

26

Shorthand operator	Meaning	Shorthand operator	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	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		
x >>= y	x = x >> y		

Conditional Statements

A conditional statement is a set of commands that executes if a specified condition is true. JavaScript supports two conditional statements: if...else and switch.

Use the if statement to execute a statement if a logical condition is true.
Use the optional else clause to execute a statement if the condition is false. An if statement looks as follows:

```
if (condition)
    statement_1
[else
    statement_2]
```

IF...Else Statement example

30

```
var gasTank = 34; //gallons of gas
```

```
if (gasTank === 34) {  
    console.log('Tank is full');  
} else {  
    console.log('Tank is not full');  
}
```

IF...Else Statement example

31

```
if (gasTank === 34) {  
    console.log('Tank is full.');} else if(gasTank > 34) {  
    console.log('Oops, tank is overfilled!');} else if(gasTank < 34 && gasTank > 5) {  
    console.log('Tank refill suggested.');} else {  
    console.log('Tank refill mandatory!');}
```

A switch statement allows a program to evaluate an expression and attempt to match the expression's value to a case label. If a match is found, the program executes the associated statement. A switch statement looks as follows:

Switch Statements

33

```
switch (expression) {  
    case label_1:  
        statements_1  
        [break;]  
    case label_2:  
        statements_2  
        [break;]  
    ...  
    default:  
        statements_def  
        [break;]  
}
```

Switch Statement example

34

```
switch (fruitType) {  
  case "Oranges":  
    console.log("Oranges are $0.59 a pound.");  
    break; //break prevents fall through!  
  case "Apples":  
    console.log ("Apples are $0.32 a pound.");  
    break;  
  case "Bananas":  
    console.log ("Bananas are $0.48 a pound.");  
    break;  
  case "Mangoes":  
  case "Papayas":  
    console.log ("Mangoes and papayas are $2.79 a pound. ");  
    break;  
  default:  
    console.log ("Sorry, we are out of " + fruitType );  
}
```

Messages

```
var answer = prompt('How old are you?');
```

```
var proceed = confirm('Click OK to proceed?');
```

```
alert('Your standing on my foot!');
```

```
console.log('This appears in the browser console');
```

**Where does JS
live at?**

// embedded

<script> var x = 12; </script>

//external js files

<script src="my-app.js"></script>

//inline

submit

DONUT PARTY

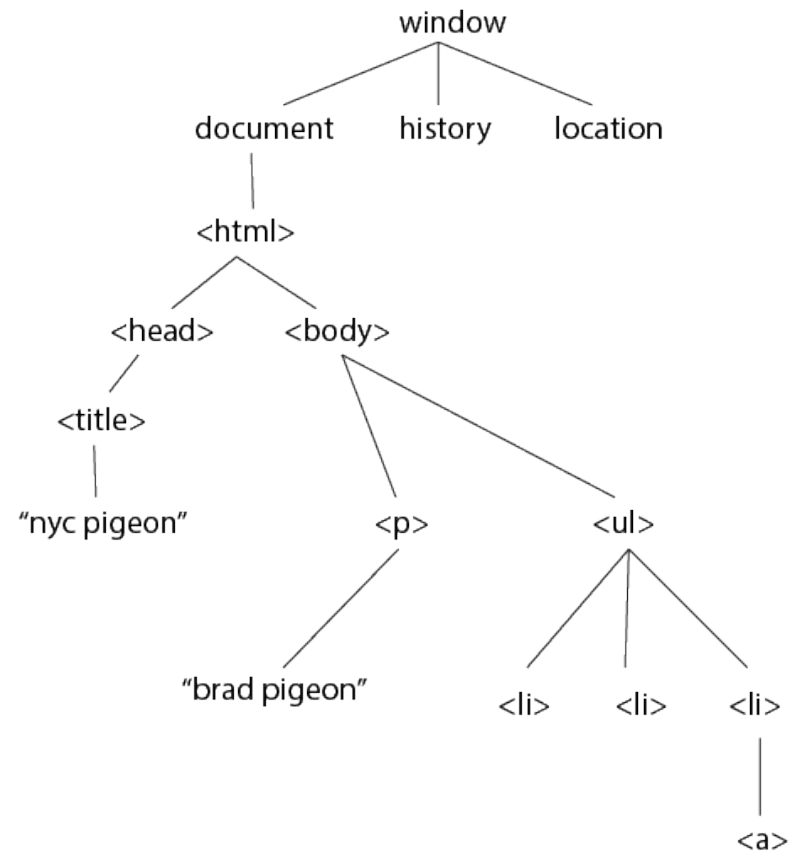
1. Create an HTML file called donut-party.html and within `<script>` tags write the JS code to accomplish the rest of the steps below.
2. Prompt the user for the number of guests they expect at a party and store it for later use.
3. Then Prompt the user for the number of of donuts available and then store it for later use.
4. Check if there are enough donuts for each guest if there are not enough alert that guests are hungry otherwise alert that guests are happy.

DOM wat?

Document
Object
Model

The DOM is a tree

42



// core JS selecting DOM element
`document.getElementsByTagName("p")[0];`

//jQuery selecting a DOM element
`$("p").first();`

getting started with jquery

- ›Introducing jQuery
- ›Why use frameworks?
- ›Setting up your page
- ›Selectors
- ›Methods

Getting Started with jQuery

Q: What is jQuery?

A: jQuery is a JavaScript library (framework)

Q: What is a framework?

A: A framework is a library of code that extends the abilities and features of a core programming language, offering additional methods and often simplifying the process to build in that native language.

Q: Why use jQuery? A:

- It just works everywhere! jQuery has been written to solve many cross browser issues that exist in core JavaScript.
- Terse code. You can often write fewer lines of code than you would in using core JavaScript to accomplish the same thing. Hence jQuery's slogan "Write Less Do More".
- Popularity. jQuery is currently at the time of writing this by far the most popular JavaScript framework. That means more forums, more code sharing, and more plugins.
- Easy extending methods. Coders can create and share their own custom plugins easily.
- Familiar DOM selectors. If you already know CSS you're a step ahead as jQuery uses all our familiar CSS selector statements.

Q: Are there other JS frameworks out there?

A: Yes, there are a lot each with their strengths and weaknesses and some are general frameworks, while others suit specific purposes.

At the time of writing this, here are some other popular JS frameworks:

MooTools, YUI, Prototype, Backbone, Cappuccino, Sammy, Angular, Ext JS, Knockout, Dojo, MochiKit, The M Project, Embed JS, MobilizeJS, Popcorn JS,...

Setting up a document for use with jQuery

49

To setup a document to run jQuery we must link to the jQuery core library first! Imagine that JavaScript is like a pet dog. On it's own it knows how to eat, sleep, and play. Loading jQuery is like teaching the dog new tricks such as roll over, fetch, etc... Imagine what would happen if we gave the command for our dog to fetch before we had taught it this trick (linked to jQuery). Therefore, we always link to the jQuery core library first. This is true when using any framework.

Setting up a document for use with jQuery

50

Remote link:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
```

Local link (downloaded from jquery.com):

```
<script src="js/jquery-1.8.2.min.js"></script>
```

Both using local as fallback solution:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
```

```
<script>window.jQuery || document.write('<script src="js/jquery-1.8.2.min.js"></script>')</script>
```

`$(selector).method(parameters);`

'\$' refers to the jQuery function object.

'selector' asks jQuery to go out into the DOM (Document Object Model) web page and select an element(s). In jQuery we can use all of the familiar selectors we use in CSS yay!

'method' these are various actions and commands attached to the jQuery object that allow us to do things like fade elements in and out or change the content of the DOM and much much more.

'parameters' are options we can set for each method.

```
$('#h1').css({'background':'yellow'});
```

'\$' refers to the jQuery object.

'h1' asks jQuery to go out into the DOM (Document Object Model) web page and select all <h1>.

'css' method applies some CSS upon the selected element(s).

'{'background':'yellow'}' changes the background color to yellow.

Usage:

<http://api.jquery.com/css/>

A Few jQuery Selectors

53

Select Elements

`$('h1')`

Select ID

`$('#nav')`

Select Class

`$('.celeb')`

Select Attribute

`$('a[href^=http://]')`

Select Descendent(s)

`$('#nav li')`

Select Child(ren)

`$('#nav > li')`

Select Sibling(s)

`$('h3 + p')`

Select by position

`$('li:eq(1)')` //selects the second list item

A Few jQuery Methods

54

AJAX

`$.post()`, `$.get()`, `$.getJSON()`, `ajaxComplete()`

Effects

`animate()`, `fadeTo()`, `show()`, `hide()`, `SlideToggle()`

Events

`click()`, `hover()`, `focus()`, `blur()`, `keypress()`

Manipulation

`addClass()`, `clone()`, `empty()`, `attr()`, `val()`

Traversing

`eq()`, `each()`, `has()`, `closest()`, `find()`

Q: Do I have to remember all of these methods there are a lot?

A: No, having an idea of what possibilities are available is useful, and familiarizing yourself with some that you will use from day to day is useful, unless you're a jQuery guru you probably will not memorize every method. Fortunately we can look them up and see their usage at sites like:

API: <http://api.jquery.com/>

Documentation: <http://docs.jquery.com/>

Or nifty cheatsheets like: <http://oscarotero.com/jquery/>

and <http://overapi.com/jquery>

Q: How will I know when to use jQuery versus core JavaScript?

A: Firstly, jQuery is written in JavaScript so you can think of them as variations of the same thing instead of two separate things. To help answer this, jQuery methods are useful for many things; however there are certain fundamental parts of JavaScript that we still need in order to make flexible web applications. For example jQuery on its own does not have method for declaring variables, functions, if statements, or doing math...

This means that we can get a lot more mileage by integrating jQuery with core JavaScript.

Jquery and JS working as one.

57

```
var x = 12,
```

```
    y = 5,
```

```
    total = x + y;
```

```
if (total === 17) {
```

```
    $('p').text('Correct!');
```

```
} else {
```

```
    $('p').text('Incorrect.');
```

```
}
```

Setup Steps

58

1. Include link to jQuery core library:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.1/  
jquery.min.js"></script>
```

2. Link to your custom js file

```
<script src="./js/app.js"></script>
```

3. Start your app.js with document ready command:

```
$(document).ready(function() {  
    // your app code goes here.  
});
```

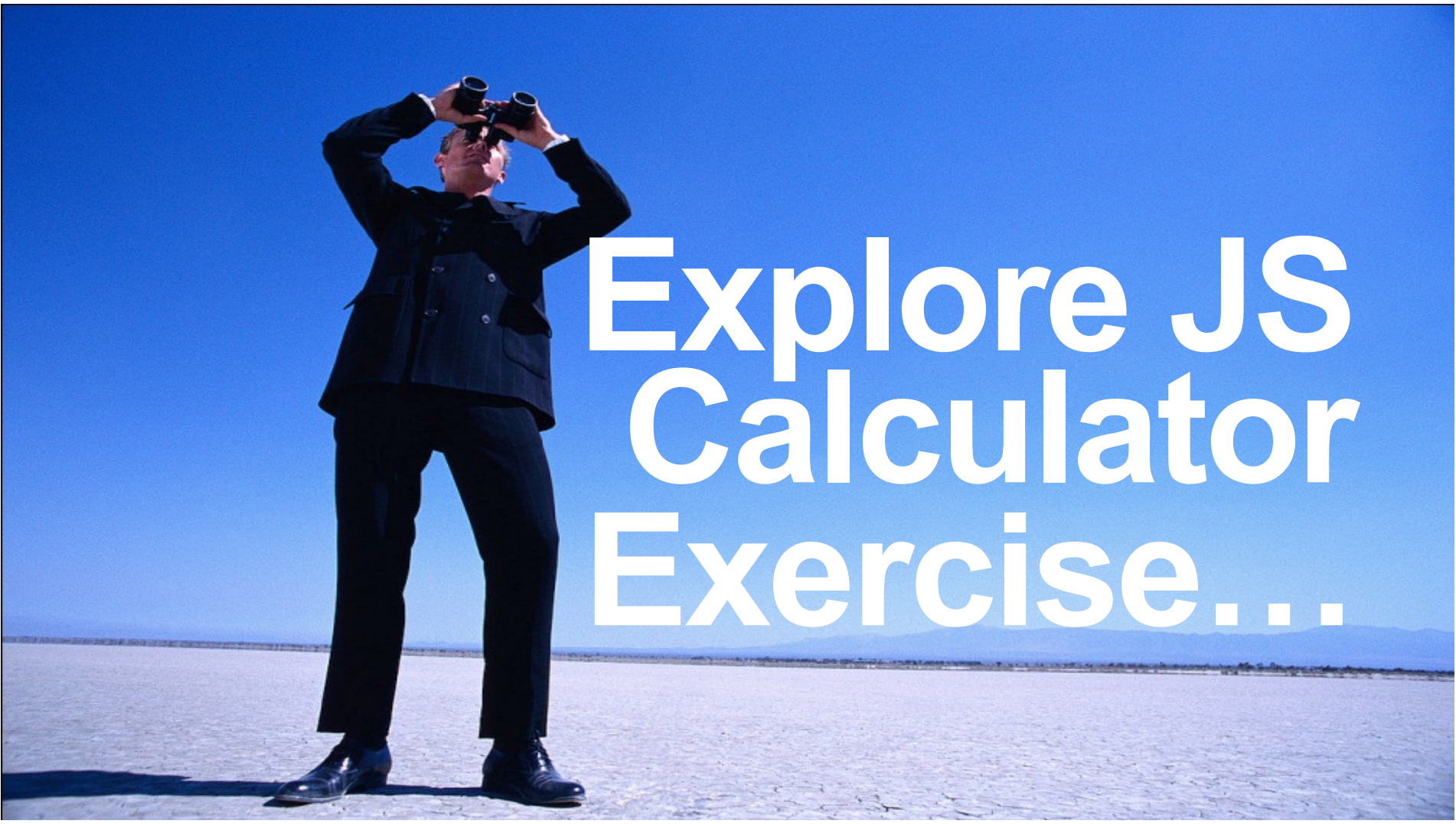
Explore Selectors & Methods...



Retrieving form values with jQuery

```
<form id="mailto" action="#" method="post">
  <input type="text" id="address" name="address">
  <input type="submit" value="update">
</form>
<p></p>
```

```
$("#mailto").submit( function() {
  var address = $("#address").val(); //retrieves the value
  $("#p").text("Mail to: "+address);
  return false; //prevent form submission.
});
```

Explore JS Calculator Exercise...

DONUT PARTY II – FROSTED FURY

1. Duplicate the first donut apps html file.
2. Integrate jQuery this time by replacing the prompts with form inputs and by replacing the alerts by writing the text into the html document.