# Learning about JavaScript

Angelia Lau

December 25, 2017

## 1 Primitive Datatypes

1. Numbers
2. Strings
3. Boolean
4. Null (explicitly defined as nothing)
5. Undefined (when a variable is declared but not initialised to a value yet)

### 1.1 Numbers

You can do math in the console!

### 1.2 Strings

- Single quotes and double quotes are treated equally, as long as they are matched!
- Concatenation works
- Escape characters: \" or \'
- "A string".length returns len
- "The Beatles"[5] gives "e"

### 1.3 Quick Exercise

1. 100 % 3 = 1
2. ("blah" + "blah")[6] = "a"
3. $``hello".length \,\%\, ``hi\backslash".length = 5\%3 = 2$

## 2 Variables in JS

1. var name = whatever
2. var names have to be camelCase

## 3 Methods in JS

1. clear() is an example method, it clears the console
2. Other built in functions:
   - `alert(‘‘Hello there!")` shows a pop up
   - `console.log(‘‘hello from the console")` is like a LogCat message in AS
   - `var userName = prompt(‘‘What is your name?");`
     stores user input to the variable userName

## 4 Boolean logic in JS

Apart from the usual >, >=, <, <=, &&, ||, ! there are four more:

`Assuming` $x = 5$,

| Operator | Name | Example | Result |
|---|---|---|---|
| == | Equal to | x==“5” | true |
| = | Not equal to | x=9 | true |
| === | Equal value and type | x===“5” | false |
| == | Not equal value and type | x==“5” | true |

*note that “==” does type coercion, and converts different data types to the same one*

Here are some interesting cases:

| Example | Result |
|---|---|
| true == “1” | true |
| false == “0” | true |
| null == undefined | true |
| NaN == NaN | false |

*note that NaN = Not A Number*

### 4.1 Truthy and Falsey

Inherently *falsey* values; false, 0, ””, null, undefined, NaN.
Everything else is *truthy*.

### 4.2 Quick Exercise

#### 4.2.1 Question 1

$$\begin{aligned}\text{var x} &= 10;\\ \text{var y} &= \text{“}a\text{”};\\ (y === \text{“}b\text{”})\ ||\ (x >= 10) &= true\end{aligned}$$

#### 4.2.2 Question 2

$$\begin{aligned}\text{var x} &= 3;\\ \text{var y} &= 8;\\ ![(x == \text{“3”})\ ||\ (x === y)]\ \&\&\ ![(y! = 8)\&\&(x <= y)] &= false\end{aligned}$$

#### 4.2.3 Question 3

$$\begin{aligned}\text{var str} &= \text{“”};\\ \text{var msg} &= \text{“}haha!\text{”};\\ \text{var isFunny} &= \text{“}false\text{”};\\ !((str\ ||\ msg)\ \&\&\ isFunny) &= false\end{aligned}$$

Both parts are true since they have values, invert with ! and so it becomes false.

## 5 Loops

while and for loops are just like in java. Go rock em!

# 6 Functions

## 6.1 Function Declaration

### 6.1.1 Example syntax

```
function capitalize(str){
     return str.charAt(0).toUpperCase() + str.slice(1);
}
```

## 6.2 Function Expression

### 6.2.1 Example syntax

```
var capitalize=function(str){
     return str.charAt(0).toUpperCase() + str.slice(1);
}
```

However, like all variables, when the var value is changed to a normal value (ie a string or integer for example), the function expression will be lost.

# 7 Scope

Every function has its own scope and the contents within the scope are not shared between functions. If variables within a function are not initialised with `var`, they will access global vars. Else, they will take on the initialised value.

## 7.1 Example:

```
 var phrase = "hi there!"
function doSomething(){
 var phrase = ‘‘Goodbye!’’;
 console.log(phrase);
}
```

When you enter the function `doSomething()`, it returns `Goodbye!`. But globally, when you search for `phrase`, you will get `"hi there!"`.

# 8 Higher order functions!

Passing functions to other functions.

## 8.1 Example

```
setInterval(<some function>, <time interval in ms>)
```

So, an example would be setInterval(singTwinkle, 1000) which calls the `singTwinkle` method for 1000ms. Note that the function called doesn't have parenthesis. This is because the method isn't called by us, but called by the higher order method.

## 8.2 Anonymous functions

All of this can also be written as `setInterval(function(){code here...;})`