JavaScript Cheat Sheet by Irin

10/9/15

HTML link	<script src = "main.js"></script>

debugging	
single line comments	//
multi	/* multi line "block" comment */
comment out	command-/
log in to console	command-option-j, >> to console
console log	console.log("Hello");
alert	alert("Hello");

data types	
string - text itself	"Hello"
number - all are floats	5, 5.5, 1000
boolean	true, false
undefined	no value

variables - container for a value, any data type	
1. declare variable	var name;
2. assign value	var name = "Jeff";
OR - both at once	var name = "Jeff"; console.log(name); >Jeff
once declared, can change value no need to redeclare	name = "Steve"; console.log(name); >Steve

always end with	;
basic math	10 + 10; >20 var x = 100; x * 40; >4000
i = i + 1	i++
i = i - 1	i--
string concatenation - connect strings	"Hello" + "Class"; >HelloClass

arrays - hold collections of data, any data type(s)	["Snoopy", "Charlie", "Patty"]; [11, "elephant"];
can be stored as variable	var class_names = ["Julie", "Rob"];
access - index of item is numbered position, starting at zero	myArray[0]
1. declare array	peanuts = ["Snoopy", "Charlie", "Patty"];
2. access element	console.log(peanuts[0]); <Snoopy

array can store arrays - multi-dimensional array	var toyotas = ["A", "B"];
	var porsches = ["C", "D"];
	var cars = [toyotas, porsches];
	or - var cars = ["A", "B"], ["C", "D"];
	console.log(cars[1][0]);
	>C

objects - contain key-value pairs

1. create object with strings for keys

```
var car = {make: 'Toyota', model: 'Prius'};
```

2. retrieve data with bracket notation

```
console.log(car['make']);
>"Toyota"
```

or - "dot notation" to retrieve data

```
var user = {first: 'John', last: 'Smith'};
console.log(user.first);
>"John"
```

can put over multiple lines

```
var user = {
  first: "John",
  last: "Smith"
};
```

logic - control flow of your program

value comparison

```
"hi" = "hi"     >true
"hi" = bye"     >false
(10 - 5) == 5       >true
(10 - 5) == "5"     >true - only checks value, not type
(10 - 5) === "5"  >false - checks value and type
```

not equal

```
!= - value
!== - value and type
```

also

```
<, >, <=, >=
5>10;
>false
```

&& and - true if both true, otherwise false

|| or - true if either one true, if both are false then false

! not

conditionals - if statement, runs code if true

```
if (5>10) {
  console.log("won't see");
}
if (5<10) {
  console.log("will see");
}
```

else - runs only if "if" evaluates to false

```
if (5>10) {
  console.log("won't see");
} else {
  console.log("will see");
}

var x = 2
if (x<10) {
  alert(x + " is less than 10");
} else {
  console.log("Your var was " + x + " and is not less than 10");
}
```

else if

```
if (5>10) {
  console.log("won't see");
} else if (5 === 5) {
  console.log("will see");
} else {
  console.log("won't get here");
}
```

functions - procedure that performs a specific action,
to encapsulate code for later use

1. define - what want function to do

```
function shoutHello() {
  alert("Hello");
}
```

2. call - tell function to execute

```
shoutHello();
```

argument - input specific to function, can't be accessed outside of function definition

1. defintе

```
function shoutToWorld(myString) {
  console.log(myString);
  alert(myString);
}
```

2. call

```
shoutToWorld("Hi");
```

but can't access myString here

return - to access value outside function

```
function addNums(num1, num2) {
  var sum = num1 + num2;
  return sum;
}
```

call

```
var mySum = addNums(1,2);
>3
```

a function can return only once then ends,
can't add another variable and return after first

```
function getName(name) {
  console.log("Hi " + name);
}
getName("Irin");
>Hi Irin
```

no arguments

```
function shoo() {
  console.log("Go away!");
}
shoo();
>Go away!
```

---

check data type of assigned value, type of operator

```
var yourData = "Data";
  console.log(typeOf yourData);
>String
```

index of - to check index of a particular value

```
var peanuts = ["Charlie", "Snoopy"];
  var SnoopyPosition = peanuts.indexOf("Snoopy");
  console.log(SnoopyPosition);
>1
console.log(peanuts[SnoopyPosition]);
>"Snoopy"
```

loops - execute block of code multiple times, typically one variable or condition changes each time it's run

for loops - executes set # of times, set the # of times using iterator variable
define - value before iteration,
a condition under which the loops continues,
how it changes value after each iteration

```
i
i=0
1<10
i++1
for (var i=0; i<10; i++) {
  console.log(i); - another example console.log(i+1);
}
>0 … 9
var beers = ["Lagunitas", "Peak"];
  for (var i=0; i<beers.length; i++) {
  console.log(i);
}
>Lagunitas, Peak

var names = ["A", "B", "C"];
for (var i=0; i<names.length; i++) {
  console.log(names[i] = " is my friend.");
}
```

while loops - execute as long as condition is true, specify condition, all changes to condition must happen in block of loop or will never terminate

```
var x=6;
while (x<10) {
  console.log("on number " + x);
  x++;
}
>on number 6 … on number 9

var x=0;
while (x<names.length) {
  console.log(names[x] + " is my friend.");
x++;
}
```