

HTML link	<code><script src = "main.js"></script></code>
debugging	
single line comments	<code>//</code>
multi	<code>/* multi line "block" comment */</code>
comment out	<code>command-/*</code>
log in to console	<code>command-option-j, >> to console</code>
console log	<code>console.log("Hello");</code>
alert	<code>alert("Hello");</code>
data types	
string - text itself	<code>"Hello"</code>
number - all are floats	<code>5, 5.5, 1000</code>
boolean	<code>true, false</code>
undefined	<code>no value</code>
variables - container for a value, any data type	
1. declare variable	<code>var name;</code>
2. assign value	<code>var name = "Jeff";</code>
OR - both at once	<code>var name = "Jeff"; console.log(name);</code>
	<code>>Jeff</code>
once declared, can change value	<code>name = "Steve";</code>
no need to redeclare	<code>console.log(name); >Steve</code>
always end with	
	<code>;</code>
basic math	<code>10 + 10; >20 var x = 100; x * 40; >4000</code>
<code>i = i + 1</code>	<code>i++</code>
<code>i = i - 1</code>	<code>i--</code>
string concatenation - connect strings	<code>"Hello" + "Class"; >HelloClass</code>
arrays - hold collections of data, any data type(s)	
can be stored as variable	<code>["Snoopy", "Charlie", "Patty"];</code>
access - index of item is numbered position, starting at zero	<code>[11, "elephant"];</code>
1. declare array	<code>var class_names = ["Julie", "Rob"];</code>
2. access element	<code>myArray[0]</code>
array can store arrays - multi-dimensional array	<code>peanuts = ["Snoopy", "Charlie", "Patty"]; console.log(peanuts[0]); <Snoopy 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</code>

objects - contain key-value pairs

1. create object with strings for keys
2. retrieve data with bracket notation

or - "dot notation" to retrieve data

can put over multiple lines

```
var car = {make: 'Toyota', model: 'Prius'};
console.log(car['make']);
>"Toyota"
var user = {first: 'John', last: 'Smith'};
console.log(user.first);
>"John"
var user = {
  first: "John",
  last: "Smith"
};
```

logic - control flow of your program

value comparison

not equal

also

&&

||

!

```
"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
!= - value
!== - value and type
<, >, <=, >=
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

else - runs only if "if" evaluates to false

else if

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

<p>functions - procedure that performs a specific action, to encapsulate code for later use</p> <p>1. define - what want function to do</p> <p>2. call - tell function to execute</p> <p>argument - input specific to function, can't be accessed outside of function definition</p> <p>1. define</p> <p>2. call</p> <p>but can't access myString here</p> <p>return - to access value outside function</p> <p>call</p> <p>a function can return only once then ends, can't add another variable and return after first</p> <p>no arguments</p>	<pre>function shoutHello() { alert("Hello"); shoutHello(); }</pre> <pre>function shoutToWorld(myString) { console.log(myString); alert(myString); } shoutToWorld("Hi");</pre> <pre>function addNums(num1, num2) { var sum = num1 + num2; return sum; } var mySum = addNums(1,2); >3</pre> <pre>function getName(name) { console.log("Hi " + name); } getName("Irin"); >Hi Irin</pre> <pre>function shoo() { console.log("Go away!"); } shoo(); >Go away!</pre>
<p>check data type of assigned value, type of operator</p> <p>index of - to check index of a particular value</p>	<pre>var yourData = "Data"; console.log(typeof yourData); >String</pre> <pre>var peanuts = ["Charlie", "Snoopy"]; var SnoopyPosition = peanuts.indexOf("Snoopy"); console.log(SnoopyPosition); >1 console.log(peanuts[SnoopyPosition]); >"Snoopy"</pre>

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

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