JavaScript II

DENVER PUBLIC LIBRARY

Conditional Statements

SLIDES FOR THIS CLASS: http://bit.ly/DPLjs2

If Statements

```
if (condition)
  {
  code to be executed if condition is true
  }
```

If/Else Statements

```
if (condition) {
      code to be executed if condition is true
} else {
      code to be executed if condition is false
}
```

If/Else If/Else Statements

```
if (condition1) {
      code to be executed if condition 1 is true
} else if (condition2) {
      code to be executed if condition 2 is true
} else {
      code to be executed neither condition 1 nor 2 is true
}
```

COMPARISONS

`===`	Equality
`!==`	Inequality
`>`	Greater than
`>=`	Greater than or equal to
`<`	Less than
`<=`	Less than or equal to

LOGICAL OPERATORS

`&&`	and
`||`	or
`!`	not

THINGS TO REMEMBER:

If you don't use comparison or logical operator, JS treats it like you're asking if a value is true (or truth-y, at least).

JS reads from left to right and stops as soon as it has an answer. So: `(false && anything) === false`, and `(true || anything) === true`.

Don't confuse **=** (assigns a value) with **==** (equal value) or **===** (equal value and equal type).

When combining multiple conditions, use parentheses to group values properly.

Loops

While loop

```
while (condition) {
      statements to repeat
}
```

If you don't want your `while` loop to run forever, the body of your loop will need to have a statement which updates the variable in the condition. Like so:

```
var coffee=6;
while (coffee > 0) {
    console.log('I've got ' + coffee + ' cups!');
    coffee--;
}
```

This statement decreases the `coffee` variable every time the loop runs, so eventually `coffee` will be `0` and the loop will stop running.

For loop

```
for (initialize; condition; update) {
      statements to repeat
}
```

A for loop, on the other hand, includes how the variable will be updated at the beginning.

```
for (coffee=6; coffee > 0; coffee--) {
      console.log('I still have ' +
      coffee + ' cups!');
}
```

EXIT FROM A LOOP
Remember that you can use **`break`** to exit a loop.

Arrays

An array is a datatype that holds an ordered list of values.

```
var arrayName = [thing0, thing1, thing2];
```

To refer to a value in an array, we can use bracket notation. Inside the brackets, place the index of the array item you want to refer to (which number it is in the array). Just remember that JavaScript starts counting at 0, not 1!

```
var rainbowColors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'];
var firstColor = rainbowColors[0];
var lastColor = rainbowColors[6];
```

so this would set the variable `firstColor` to `'red'`...

...and this would set the variable `lastColor` to `'violet'`

Bracket notation can also change values in an array or add new values to an array:

```
rainbowColors[6] = 'purple' //changed a value
rainbowColors[7] = 'vengeful violet' //added a new value to the array
```