General Assembly

STUDY GUIDE

CONTROL FLOW

Overview

Computers make decisions based on comparisons, all of which boil down to a boolean value: true or false

Basic Comparison Statements in JavaScript

- Greater than: >
- Greater than or equal to: >=
- Less than: <
- Less than or equal to: <=
- Loose equality: ==
- Strict equality: ===
- Inequality: !==

Equality and Inequality Operators: In-Depth

Loose Equality: ==

JavaScript uses **type coercion,** which means it transforms values so the data types match. Loose comparisons can lead to confusing, unexpected answers.

- "4" == 4 would evaluate to true
- 4 == 4 would evaluate to true

Strict Equality: ===

JavaScript matches both data type AND value

- "4" === 4 would evaluate to false
- 4 === 4 would evaluate to true

Inequality: !==

If the data types don't match OR the values don't match, it will return true

Using Comparisons to Create Conditionals

- Conditional statements can be used to control what, when, and how things happen in your program
- Conditionals are **if statements** that tell a computer to execute or skip a block of code if a comparison returns true

if (assignmentsCompleted > .80) { console.log("Ready to graduate!"); }

- An **else block** can be added to an if statement. Else blocks tell the computer what to do if a conditional returns false.

if (assignmentsCompleted > .80) { console.log("Ready to graduate!"); } else { console.log("Better catch up on your homework!"); }

- An **if... else if... else chain** can account for more than two possibilities. The first conditional that returns true is executed, and the rest are skipped.

```
if (assignmentsCompleted > .80) { console.log("Ready to graduate!"); } else if (assignmentsCompleted > .65) { console.log("Better catch up on your homework!"); } else { console.log("You've got a lot of work left to do!"); }
```

Loops

A loop is a conditional statement that will continue to execute as soon as a certain condition is met

For Loops

for loops are known for their accuracy and precision. for loops are often used to loop through each element of an array.

```
for (let i = 0; i < 100; i++) { console.log(i); }
```

Parts of a for loop:

- Keyword - indicates the kind of loop you're running, in this case, for
- Control Statement - found in parentheses after the keyword, the control statement tells the computer how the loop should operate. The parts of the control statement must be separated by semicolons.
  - Parts of the Control Statement:
    - Iterator - the variable operated on by the loop
    - Conditional Statement - tells the loop whether or not to keep going. If the Conditional Statement returns true, the loop will execute the code block and then check the conditional again. Once the condition evaluates to false, the loop will stop.
    - The Final Expression - tells the loop what to do after each iteration of the code block. The final expression should ensure that the loop is progressing towards making the conditional false.
    - Code Block - tells the computer what to do in each iteration of the loop

While Loops

while loops only include a conditional. As long as the conditional is met, the loop will continue.

When writing a while loop, ensure that the code inside the code block will eventually make the conditional statement false, otherwise you may end up with an infinite loop. Infinite loops crash browsers.

```
let number = 0; while (number < 11) { number = number + 1; }
```

Determining Which Loop to Use

- If you know how many times the loop should execute, use a for loop
- If you know when the loop should stop, but not how many times it should run, use a while loop