General Assembly

Conditionals And Intro to Functions

- Not "And loops" we did that last lesson
- Class Rules reminder
- Thanks for Exit Tickets
- Any completed homework - send it via Slack
- Joke: Use a lot of internet resource to plan lessons, I went to www.conjunctivitis.com. *That's* a site for sore eyes!

JS1

Last Lesson Recap

- Command Line JS
- Data Types + Variables
- Arrays (`var ages = [14, 56]`)
- Array Methods
- Loops + Iterating Arrays
- [Ask class] Questions / clarifications?

Last Lesson Recap

Homework

Last Lesson Recap

Homework

```
var ratingsAllowed = ages.map(function(age) {
  return ratings.filter(function(rating, index) {
    return age >= ratingAges[index]
  })
})

ages.forEach(function(age, index) {
  console.log('Age ' + age + ' is allowed to see: '
              + ratingsAllowed[index].toString())
})
```

Last Lesson Recap

Homework

```
var allowed = ages.forEach(function(num){
  function ratingsFilter(value, i){
    return num >= minAgeForRating[i]
  }

  var filteredRatings = ratings.filter(ratingsFilter);

  console.log("If you're aged\t", num,
              "\tthen you can watch ratings\t",
              filteredRatings.toString() )
});
```

- June 13th Queen's Birthday
- June 16th for class? Thursday?

General Assembly

Conditionals And Intro to Functions

JS1

Objectives

- Use `if/else` conditionals and Logical Operators (`!`, `&&`, `||`)
- Differentiate `true` / `false` / truth-y / false-y
- Understand the purpose of functions
- Declare functions in your code

JS1

Conditionals And Intro to Functions

Conditional Statements

- A way to select which block of code to execute
- conditionals have 3 forms: `if`/`else`, ternary, `switch`/`case`
  - Wont be looking at `switch`/`case`
  - Can provide homework for it if interested
- Every condition is `true` or `false`

- [on board]:

```
if (<condition>) {
  <code when true>
}
```

- Optional `else` when not true:

```
if (<condition>) {
  <code when true>
} else {
  <code when false>
}
```

- Already saw this in our psuedo code from lesson 00
- Point out < & > notation

Conditionals And Intro to Functions

Conditional Statements

```
if (true) {
  console.log('Foo')
} else {
  console.log('Bar')
}
```

- simplest form
- `else` will never execute

Conditionals And Intro to Functions

Conditional Statements

```
if (false) {
  console.log('Foo')
} else {
  console.log('Bar')
}
```

- only `else` will ever execute

Conditionals And Intro to Functions

Conditional Statements

```
if (5 > 10) {
  console.log('Foo')
} else {
  console.log('Bar')
}
```

- Anything that is a yes/no question
- Note, equivalent to `if (5 > 10 === true)`

Conditionals And Intro to Functions

Conditional Statements

```
if (5 > 10) {
  console.log('Foo')
} else if (4 > 2) {
  console.log('Bar')
} else {
  console.log('Zip')
}
```

- `else if` When you have multiple conditions

Conditionals And Intro to Functions

Ternary Operators

```
if (<condition>) {
  <code when true>
} else {
  <code when false>
}
```

- Given our original if/else definition

Conditionals And Intro to Functions

Ternary Operators

```
<condition> ?
  <code when true>
:
  <code when false>
```

- Ternary means "3 parts"
- Which simplifies to:

Conditionals And Intro to Functions

Ternary Operators

```
<condition> ?
  <code when true>
:
  <code when false>
```

```
<condition> ? <code when true> : <code when false>
```

Conditionals And Intro to Functions

Ternary Operators

```
5 > 10 ? console.log('Foo') : console.log('Bar')
```

- ternary also returns a value

Conditionals And Intro to Functions

Ternary Operators

```
var name = 5 > 10 ? 'Foo' : 'Bar'
console.log(name)
```

- No curly braces, so no block
  - No block, means different scope (next lesson)
- Only 1 statement / value per true/false

Comparison Operators

Conditionals And Intro to Functions

Comparison Operators

Equality Operators

==, ===, !=, !==

Relational Operators

>, >=

<, <=

- We've seen > already.
- Relational Same as math at school

Conditionals And Intro to Functions

Equality Operators

1. =: Assignment Operator
2. ==: Loose Equality (! =: Loose *in*equality)
3. ===: Strict Equality (! ==: Strict *in*equality)

- [on board] Assignment is `var foo = 'bar'`
- Loose is useful if you understand how the *type coercion* works
  - *coercion* is trying to get two types to be the same, having to convert the value from one type to the other.
- Strict is safest
- [in node], examples:
  - `1 === true` vs `1 == true` (`true` coerced to Number)
  - `"5" === 5` vs `"5" == 5` (`"5"` coerced to Number)
  - `{} === {}` (no coercion, references differ)
  - `'[object Object]' == {}` (`{}` coerced to String)
- `!` is *bang*, or *not symbol* ,used to negate
  - `#!` is *shebang* (Ricky Martin She Bangs from 2000)
- More: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Equality_comparisons_and_sameness#Loose_equality_using

Conditionals And Intro to Functions

Relational Operators

1. >: Greater than
2. >=: Greater than or Equal To
3. <: Less than
4. <=: Less than or Equal To

- There is no inequality (bang) versions
- "equal to" means; "loosely equal to"
- Should only use on numbers
- [example]:
    - `"10" >= 5`

Conditionals And Intro to Functions

`truth`-y and `false`-y

- Ie; what is true/false in a conditional
  - Point to the conditional on the board

Conditionals And Intro to Functions

`truth`-y and `false`-y

For Strict Equality

`true`

`false`

- Forget about comparison operators for now

Conditionals And Intro to Functions

`truth`-y and `false`-y

For Loose Equality

- `false`
- `0`
- `""` (empty string)
- `NaN`
- `null`
- `undefined`

- Remember loose equality coerces the type
- Here, we're coercing to Boolean
- Everything else coerces to `true`

Conditionals And Intro to Functions

`truth`-y and `false`-y

bang bang

- Bang Bang the song from Kill Bill (Cher, then Nany Sinatra 1966)
- [on board] First ! coerces to Boolean, and negates
  - Second ! negates again
  - Examples:
  - `!!true === true`
  - `!!{} === true`

Conditionals And Intro to Functions

Logical Operators

- Used to check >1 condition at a time

Conditionals And Intro to Functions

Logical Operators

AND: &&

OR: ||

NOT: !

- Ampersand, Pipe, Bang

Conditionals And Intro to Functions

Logical Operators

```
if (true && false) {
}
```

```
if (true || false) {
}
```

- [Ask class]: `true && true`, `false || false`, etc

Conditionals And Intro to Functions

Logical Operators

Short Circuits

```
var name = (person && person.name) || "Sammy Default"
```

- Returns value of first `truth`-y part
- Walk through execution order


• Break + fizzbuzz = 45min

Fizzbuzz Code Challenge

http://bit.ly/fizzbuzz-challenge

- Popular as an interview Q
- Now; popular ironically. If you haven't heard of it, you obviously don't participate in the community

Fizzbuzz Code Challenge

http://bit.ly/fizzbuzz-challenge

Hints:

- https://mdn.io/for
- https://mdn.io/remainder

- 30 min
- Further work:
  - Comparison Operators: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Comparison_Operators
  - Logical Operators: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Logical_Operators

Conditionals And Intro to Functions

Functions

- What?
  - Designers: don't build page top-to-bottom
  - Build a Pattern Library
  - Re-usable parts of the whole
  - Analogy: Uber driver has a function
- Why?
  - Reusable statement(s)
  - Another variable
  - Don't Repeat Yourself (DRY)
- [Ask Class]: Functions in `fizzbuzz()`? inside the for loop
- [Anecdote]: Calculate age app | compare ages | duplicated an error (non-DRY)

Conditionals And Intro to Functions

Defining

Function Declaration

Function Expression

- [on whiteboard]
  - Declaration: `function speak() {}`
  - Expression: `var speak = function() {}`
    - Just another data type
- Hoisting
- Walk through the syntax
  - Name only *required* for declaration

Conditionals And Intro to Functions

Defining

Function Declaration

Function Expression

↳ ES6 Arrow Function

- "some diff, revisit next lesson"

Conditionals And Intro to Functions

Calling

(aka: Invoking)

- defining vs invoking/calling/executing
- parens to call
  - examples already seen: `.toString()`, `.forEach()`
- [codealong]
  - Create a new file `functions.js`
  - `function foo() {}`
  - `var bar = function() {}`
  - Call the declaration & expression examples
- [Checks]:
  - Diff of defintion vs expression?
  - [whiteboard] `var foo = [function speak() {}]`
  - What is `foo[0]`? Declaraion or Expression? Why?
  - [Pair/Share]: How to convert `person.speak` to ES6 arrow function?

Conditionals + Intro to Functions

Exercise: Rolling Dice

http://bit.ly/dice-homework

- Comparison Operators: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Comparison_Operators
- Logical Operators: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Logical_Operators

JS1

Objectives

- Revisit each of the objectives on board

JS1

Next Lesson

- More Functions
- Scope & Scope Chains
- Closures
- What is `this`?

JS1

Questions?

JS1

Exit Tickets

http://ga.co/js1syd

- [share in Slack]

General Assembly

JS1