Introduction to JavaScript

Class 3

Wi-Fi

Guest Wifi	(QGuest Network)
Username	user1@guest.com
Password	cEh7Yu7K
GitHub	d3nise
Slides	Class 3

3.1

Loops

Loops *offer a quick and easy way to do something repeatedly.*

while Loops

- **`while`** will repeat the same code over and over until some condition is met.

```
var bottlesOfBeer = 50;

while (bottlesOfBeer > 0) {
  console.log(bottlesOfBeer + ' bottles of beer on the wall');
  bottlesOfBeer = bottlesOfBeer - 1;
}
```

for Loops

- `for` loops are very similar, but you declare a counter in the statement.

```
// will count 1 to 10
for (var i = 1; i <= 10; i++) {
  console.log(i);
}
```

Loops and logic

- You can add other statements or logical operators inside the loops.

```
// Count from 1 to 100

for (var i = 1; i <= 100; i++) {
  if (i % 3 === 0) {
    // Says 'Fizz' after multiples of three
    console.log(' Fizz');
  } else if (i % 5 === 0) {
    // Says 'Buzz' after multiples of five
    console.log(' Buzz');
  } else {
    console.log(i);
  }
}
```

Infinite Loops

Break

- To exit a loop, use the **break** statement.

```
for (let current = 20; ; current = current + 1) {
    if (current % 7 == 0) {
    console.log(current);
    break;
    }
}
// → 21
```

Let's Develop It

- Write a loop that gives you the 9's times table, from 9 x 1 = 9 to 9 x 12 = 108.
- Finish early? Try using a loop inside a loop to write all the times tables, from 1 to 12.

3.2

Arrays

***Arrays** in JavaScript are dynamic*

Arrays

- Arrays are ordered lists of values.

```
var arrayName = [value0, value1];
```

- You can put different types of data into an array.

```
var rainbowColors = ['Red', 'Orange', 'Yellow',
'Green',
  'Blue', 'Indigo', 'Violet'];

var lotteryNumbers = [33, 72, 64, 18, 17, 85];

var myFavoriteThings = ['Broccoli', 1024, 'Sherlock'];
```

Array Length

- The `length` property tells you how many things are in an array.

```
var rainbowColors = ['Red', 'Orange', 'Yellow',
'Green',
'Blue', 'Indigo', 'Violet'];

console.log(rainbowColors.length);
```

Using Arrays

- You can access items with **bracket notation** by using the position of the item you want.

```
var rainbowColors = ['Red', 'Orange', 'Yellow',
'Green',
'Blue', 'Indigo', 'Violet'];

var firstColor = rainbowColors[0];
var lastColor  = rainbowColors[6];
```

JS arrays are **zero-indexed**, so counting starts at 0.

Changing arrays

- You can use bracket notation to change an item in an array.

```
var myFavoriteThings = [Cats, 6, Coding];

myFavoriteThings[0] = Kittens;
```

Expanding arrays

- Arrays do not have a fixed length. You can use **push** to add something to an array.

```
var myFavoriteThings = [Cats, 6, Coding];

myFavoriteThings.push('Javascript');
```

Let's Develop It

- Create an array of your favorite foods.
- Print a few values onto your screen.

Arrays + loops = BFF

Iterating through arrays

- Use a `for` loop to easily work with each item in an array.

```
var rainbowColors = ['Red', 'Orange', 'Yellow', 'Green',
  'Blue', 'Indigo', 'Violet'];

for (var i = 0; i < rainbowColors.length; i++) {
  console.log(rainbowColors[i]);
}
```

Let's Develop It

- Use a `for` loop to print a list of all your favorite foods.

3.3

Objects

JavaScript is designed on a simple ***object-based paradigm.***

Objects

- Objects let us store a collection of properties.

```
var objectName = {
  propertyName: propertyValue,
  propertyName: propertyValue
};
```

```
var user = {
  hometown: 'Atlanta, GA',
  hair: 'Auburn',
  likes: ['knitting', 'code'],
  birthday: {month: 10, day: 17}
};
```

Accessing Objects

- You can retrieve values using dot notation.

```
var user = {
  hometown: 'Atlanta, GA',
  hair: 'Auburn'
};

var usersHometown = user.hometown;
```

- Or using bracket notation (like arrays)

```
var usersHair = user['hair'];
```

Changing Objects

- You can use dot or bracket notation to change properties.

```
var user = {
  hometown: 'Atlanta, GA',
  hair: 'Auburn'
};

user.hair = 'blue';
```

- Add new properties

```
user.married = true;
```

- Or delete them

```
delete user.married;
```

Arrays of Objects

- Because arrays can hold any data type, they can also hold objects.

```
var users = [
  {name: 'Jolene', age: 21},
  {name: 'Alexa',  age: 18}
];

for (var i = 0; i < users.length; i++) {
  var user = users[i];
  console.log(user.name + ' is ' + user.age + ' years
old.');
}
```

Objects

- Just like other data types, objects can be passed into functions:

```
var jolene = {
  age: 21,
  hairColor: 'Auburn',
  likes: ['pizza', 'tacos'],
  birthday: {month: 3, day: 14, year: 1995}
}

function describeUser(user) {
  console.log('You are ' + user.age + ' years old with '
  + user.hairColor + ' hair.');
}

describeUser(jolene);
```

Let's Develop It

- Create an object to hold information on your favorite recipe. It should have properties for:
  - `recipeTitle` (a string)
  - `servings` (a number)
  - `ingredients` (an array of strings)
  - `directions` (a string)
- Try displaying some information about your recipe.
- **Bonus**: Create a loop to list all the directions.

Object methods

- Objects can also hold functions.

```
var jolene = {
  age: 21,
  hairColor: 'Auburn',
  talk: function() {
    console.log('Hello!');
  },
  eat: function(food) {
    console.log('Yum, I love ' + food);
  }
};
```

- Call object methods using dot notation:

```
jolene.talk();
jolene.eat('pizza');
```

Let's Develop It

- Go back to your recipe object. Add a function called `letsCook` that says "I'm hungry! Let's cook..." with the name of your recipe title.
- Call your new method.

Resources

- JavaScript Guide, from the Mozilla Developers Network.
- Code Academy, with interactive JavaScript lessons to help you review.
- W3schools
- Freecodecamp
- udemy

YOU DID IT!

Any questions?