CS4220 Node.js & Vue.js

Cydney Auman
Albert Cervantes
CSULA

Intro to Javascript

Objects ES5 vs ES6

ES5

```
var name = 'Cydney'
var
    key = 'occupation',
    val = 'software engineer'
var person = { name: name }
person[key] = val
```

ES6

```
const name = 'Cydney'
const
    key = occupation,
    val = software engineer

const person = { name, [key]: val }
```

Object & Array Destructuring

The **destructuring assignment** syntax is a JavaScript expression that makes it possible to extract data from arrays or objects into distinct variables.

```
const arr = [ 1, 2, 3, 4 ]
const [ a, b ] = arr
console.log(a) // 1
console.log(b) // 2
```

```
const fido = {
    type: 'dog',
    breed: 'border collie',
    colors: ['black', 'white']
}
const { type, breed, colors } = fido
console.log(type) // dog
```

Default Values

A variable can be assigned a default, in the case that the value pulled from the array or object is undefined.

```
const arr = [ 1, 2 ]
const [ a, b, c = 0 ] = arr
console.log(a) // 1
console.log(b) // 2
console.log(c) // 0
```

```
const fido = {
    breed: 'border collie',
    colors: ['black', 'white']
}
const { type = 'cat' , breed, colors } = fido
console.log(type) // cat
```

Template Literals

Template literals are string literals allowing embedded expressions. You can use multi-line strings and string interpolation features with them.

Template literals are enclosed by the back-tick (` `) instead of double or single quotes. Template literals can contain placeholders. These are indicated by the dollar sign and curly braces (${expression}).

```
const food = 'sandwiches'
console.log(`i like ${food}`)
// i like sandwiches
```

ES6 Classes

Nearly all objects in JavaScript are instances of Object - a typical object inherits properties and methods from Object.prototype.

JavaScript classes introduced in ES6 are syntactical sugar over JavaScript's existing prototype-based inheritance. The class syntax is not introducing a new object-oriented inheritance model to JavaScript.

ES6 Classes

To declare a class, you use the **class** keyword.

```
class Polygon {


}
```

The **constructor** method is a special method for creating and initializing an object created with a class.

```
class Polygon {
    constructor(height, width) {
          this.height = height
          this.width = width
    }
}
```

ES6 Classes

```
class Polygon {
     constructor(height, width) {
          this.height = height
          this.width = width
     }

     calcArea() {
          return this.height * this.width
     }
}

const square = new Polygon(10, 10)
console.log(square.calcArea()) // 100
```

ES6 Classes

The **get** syntax binds an object property to a function that will be called when that property is looked up.

```
class Polygon {
     constructor(height, width) {
          this.height = height
          this.width = width

     }

     get area() {
          return this.calcArea()
     }

     calcArea() {
          return this.height * this.width
     }
}
const square = new Polygon(10, 10)
console.log(square.area) // 100
```

References and Reading

Mozilla Developer Network (Methods and Properties on Array and Objects)

-- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects

Mozilla Developer Network (Classes)

-- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes

Eloquent Javascript

-- http://eloquentjavascript.net/

-- Chapters 6