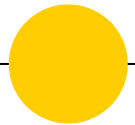
CS4220 Node.js & Vue.js

Cydney Auman
Albert Cervantes
CSULA



Javascript Continued

Object Destructuring

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

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

Default Values

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

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

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

Classes support prototype-based inheritance, super calls, instance and static methods and constructors.

We can use the keyword **extends** to inherit from another class. We can use the **super** keyword to call the parent class.

ES6 includes **getters** and **setters** for object properties. They allow us to run the code on the reading or writing of a property.

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 Numbers, Strings, Array and Objects)
-- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects

Eloquent Javascript
-- http://eloquentjavascript.net/
-- Chapters 4, 5, 6

Destructuring
-- https://leanpub.com/understandinges6/read#leanpub-auto-destructuring-for-easier-data-access

Template Literals
-- https://leanpub.com/understandinges6/read#leanpub-auto-template-literals

Classes
-- https://leanpub.com/understandinges6/read#leanpub-auto-class-declarations
-- https://medium.com/ecmascript-2015/es6-classes-and-inheritance-607804080906