PREGRADO

UNIDAD 2 | FRONTEND WEB APPLICATIONS

JAVASCRIPT REVIEW

SI653 | Aplicaciones Web

1

Al finalizar la unidad, el estudiante desarrolla y comunica una solución de aplicación frontend integrada con RESTful web services con una Arquitectura Orientada a Servicios en un ambiente de desarrollo ágil e inclusivo

2

AGENDA

OBJECTS & FUNCTIONS

PROTOTYPES

CLASSES

3

Object

Creando un object.

```
const person = {
  name : {
    firstName: 'Bob',
    lastName: 'Smith’
  },
  age: 32,
  gender: 'male',
  interests: ['music', 'skiing'],
  bio: function() {
    alert(this.name.firstName + ' ' + this.name.lastName + ' is ' + this.age +
    ' years old. He likes ' + this.interests[0] + ' and ‘ +
    this.interests[1] + '.');
  },
  greeting: function() {
    alert('Hi! I\'m ' + this.name[0] + '.');
  }
};
```

4

Constructor and object instances

Function.

```
function createNewPerson(name) {
  const obj = {};
  obj.name = name;
  obj.greeting = function() {
    alert('Hi! I\'m ' + obj.name + '.');
  };
  return obj;
}
```

5

Constructor and object instances

Constructor function.

```
function Person(name) {
  this.name = name;
  this.greeting = function() {
    alert('Hi! I\'m ' + this.name + '.');
  };
}



let person1 = new Person('Bob');
let person2 = new Person('Sarah');
```

6

Constructor and object instances

Constructor function.

```
function Person(firstName, lastName, age, gender, interests) {
  this.name = {
     first : firstName,
     last : lastName
  };
  this.age = age;
  this.gender = gender;
  this.interests = interests;
  this.bio = function() {
    alert(this.name.first + ' ' + this.name.last + ' is ‘ +
    this.age + ' years old. He likes ' + this.interests[0] + ' and ‘ +
    this.interests[1] + '.');
  };
  this.greeting = function() {
    alert('Hi! I\'m ' + this.name.first + '.');
  };
}


let person1 = new Person('Bob', 'Smith', 32, 'male', ['music', 'skiing']);
```

7

Constructor and object instances

Object() function.

```
let person1 = new Object({
  name: 'Chris',
  age: 38,
  greeting: function() {
    alert('Hi! I\'m ' + this.name + '.');
  }
});
```

8

Constructor and object instances

create() method.

```
let person2 = Object.create(person1);
person2.name;
person2.greeting();
```

9

AGENDA

OBJECTS & FUNCTIONS

PROTOTYPES

CLASSES

10

Prototype property

Object.prototype

```
let person2 = Object.create(person1);
person2.__proto__
// will return the person1 object
```

11

create() and prototypes

constructor property

```
let person3 = new person1.constructor('Karen', 'Stephenson', 26, 'female',
['playing drums', 'mountain climbing']);
```

12

Modify prototypes

Modifying prototype property

```
function Person(first, last, age, gender, interests) {

  // property and method definitions

}

let person1 = new Person('Tammi', 'Smith', 32, 'neutral', ['music', 'skiing',
'kickboxing']);

Person.prototype.farewell = function() {
  alert(this.name.first + ' has left the building. Bye for now!');
};
// correct, at this point, this references the function scope

Person.prototype.fullName = this.name.first + ' ' + this.name.last;
// incorrect, at this point, this references the global scope
```

13

AGENDA

OBJECTS & FUNCTIONS

PROTOTYPES

CLASSES

14

ECMAScript Classes

class statement

```
class Person {
  constructor(first, last, age, gender, interests) {
    this.name = {
      first,
      last
    };
    this.age = age;
    this.gender = gender;
    this.interests = interests;
  }


  greeting() {
    console.log(`Hi! I'm ${this.name.first}`);
  };


  farewell() {
    console.log(`${this.name.first} has left the building. Bye for now!`);
  };
}
```

15

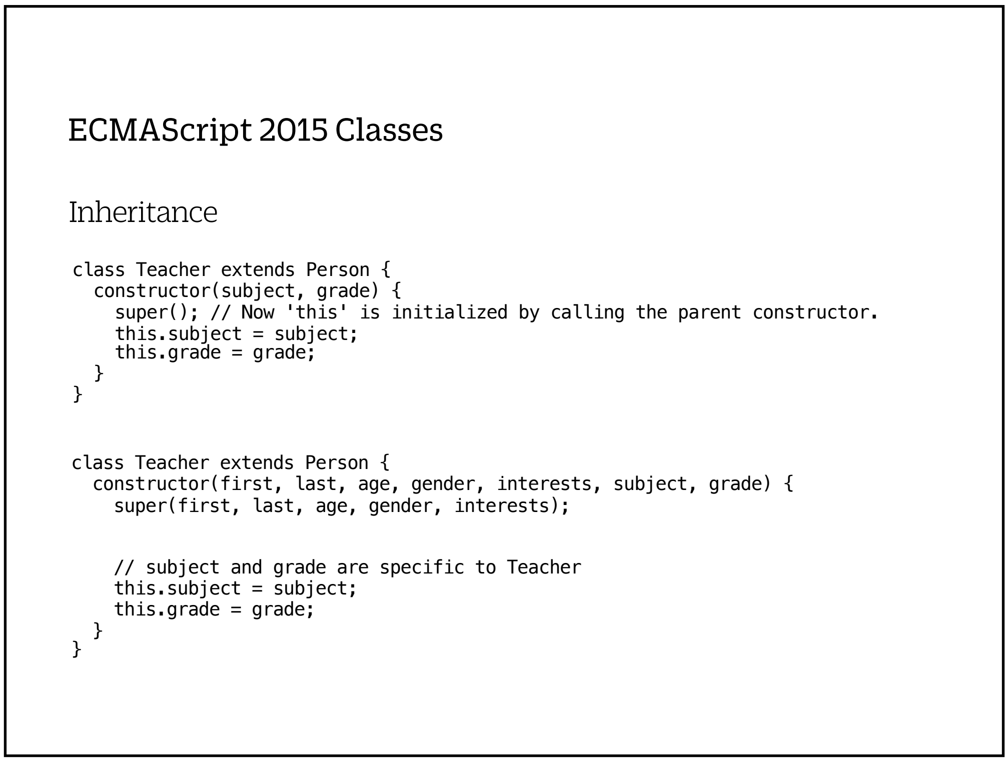
ECMAScript Classes

Instantiate objects

```
let han = new Person('Han', 'Solo', 25, 'male', ['Smuggling']);
han.greeting();
// Hi! I'm Han


let leia = new Person('Leia', 'Organa', 19, 'female', ['Government']);
leia.farewell();
// Leia has left the building. Bye for now
```

16



ECMAScript 2015 Classes

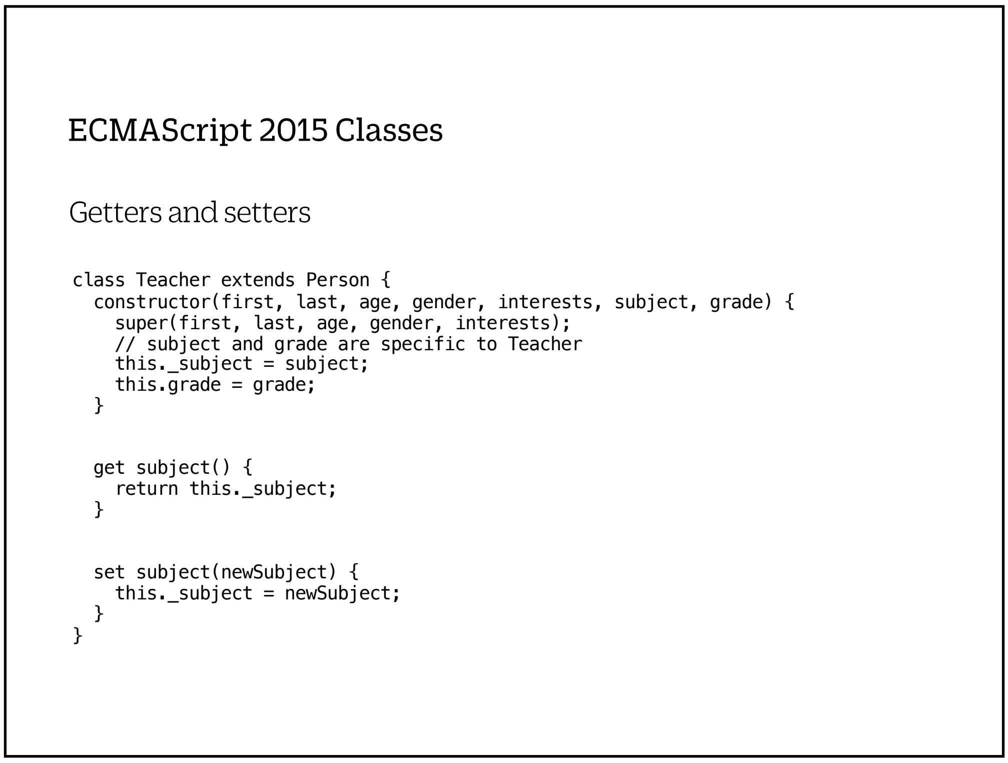
Inheritance

```
class Teacher extends Person {
  constructor(subject, grade) {
    super(); // Now 'this' is initialized by calling the parent constructor.
    this.subject = subject;
    this.grade = grade;
  }
}


class Teacher extends Person {
  constructor(first, last, age, gender, interests, subject, grade) {
    super(first, last, age, gender, interests);


    // subject and grade are specific to Teacher
    this.subject = subject;
    this.grade = grade;
  }
}
```

17



ECMAScript 2015 Classes

Getters and setters

```
class Teacher extends Person {
  constructor(first, last, age, gender, interests, subject, grade) {
    super(first, last, age, gender, interests);
    // subject and grade are specific to Teacher
    this._subject = subject;
    this.grade = grade;
  }


  get subject() {
    return this._subject;
  }


  set subject(newSubject) {
    this._subject = newSubject;
  }
}
```

18

ECMAScript 2015 Classes

Getters and setters

```
// Check the default value
console.log(snape.subject) // Returns "Dark arts"

// Change the value
snape.subject = "Balloon animals" // Sets _subject to "Balloon animals"

// Check it again and see if it matches the new value
console.log(snape.subject) // Returns "Balloon animals"
```

19

RESUMEN

Recordemos

JavaScript implementó prototypes, concepto que permitía convertir al lenguaje en un lenguaje basados en objetos. ECMAScript 2015 introdujo el concepto de classes. Sobre la bse de prototypes, ofrecía una sintaxis más familiar para developers con experiencia en otros lenguajes orientados a objetos.

20

REFERENCIAS

Para profundizar

https://www.ecma-international.org/ecma-262/6.0/

https://github.com/lukehoban/es6features

https://hackr.io/blog/top-10-web-development-frameworks-in-2020

https://vuejs.org

www

21

PREGRADO

Ingeniería de Software
Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería

22