ES5 vs ES6 Classes (6:57 mins)

ES6 (ECMAScript 2015) introduced several enhancements and syntactic sugar to JavaScript, including the introduction of class syntax for defining object prototypes. This syntax is often compared to the ES5 prototype-based approach.

ES6 Classes:

1. **Syntax Sugar:** ES6 classes provide a more readable and declarative way to define object constructors and their methods. They resemble class-based syntax from other programming languages.

```
class Person {
  constructor(name) {
    this.name = name;
  }
  sayHello() {
    console.log(`Hello, my name is ${this.name}.`);
  }
}
```

2. **Constructor:** The `constructor` method is used to initialize object instances. It is automatically called when a new object is created from the class.

3. **Class Inheritance:** ES6 classes support class inheritance using the `extends` keyword, allowing you to create a subclass that inherits properties and methods from a parent class.

```
class Student extends Person {
  constructor(name, studentId) {
    super(name);
    this.studentId = studentId;
  }
}
```

4. **Super Keyword:** The `super` keyword is used to call the constructor of the parent class or access its methods and properties from the child class.

5. **Static Methods:** You can define static methods within a class, which are called on the class itself rather than on instances of the class.

```
class MathUtils {
  static add(x, y) {
    return x + y;
  }
}
```

ES5 Constructor Functions and Prototypes:

1. **Constructor Functions:** In ES5, constructor functions are used to create objects. These functions are invoked with the `new` keyword to create instances.

```
function Person(name) {
  this.name = name;
}
Person.prototype.sayHello = function() {
  console.log('Hello, my name is ' + this.name + '.');
};
```

1. **Prototypes:** Methods are added to object prototypes to make them available to instances. This is a bit less intuitive than the class-based approach.

2. **Inheritance:** Inheritance is achieved through prototype chaining. Subtypes inherit methods and properties from their prototypes.

```
function Student(name, studentId) {
  Person.call(this, name);
  this.studentId = studentId;
}
Student.prototype = Object.create(Person.prototype);
Student.prototype.constructor = Student;
```

3. **No Built-In Super Keyword:** ES5 does not have a built-in `super` keyword. To call the parent constructor or method, you often need to use `call` or `apply` to invoke them explicitly.

In summary, ES6 classes provide a more structured and intuitive way to work with object-oriented programming in JavaScript, making it easier to define and understand class-based relationships and inheritance. However, under the hood, they still rely on prototypes and constructor functions. ES5 constructor functions and prototypes are less syntactically convenient but offer more flexibility for advanced use cases. The choice between ES6 classes and ES5 constructor functions often depends on project requirements and personal/team preferences.