Skills Network

Introduction to ES6

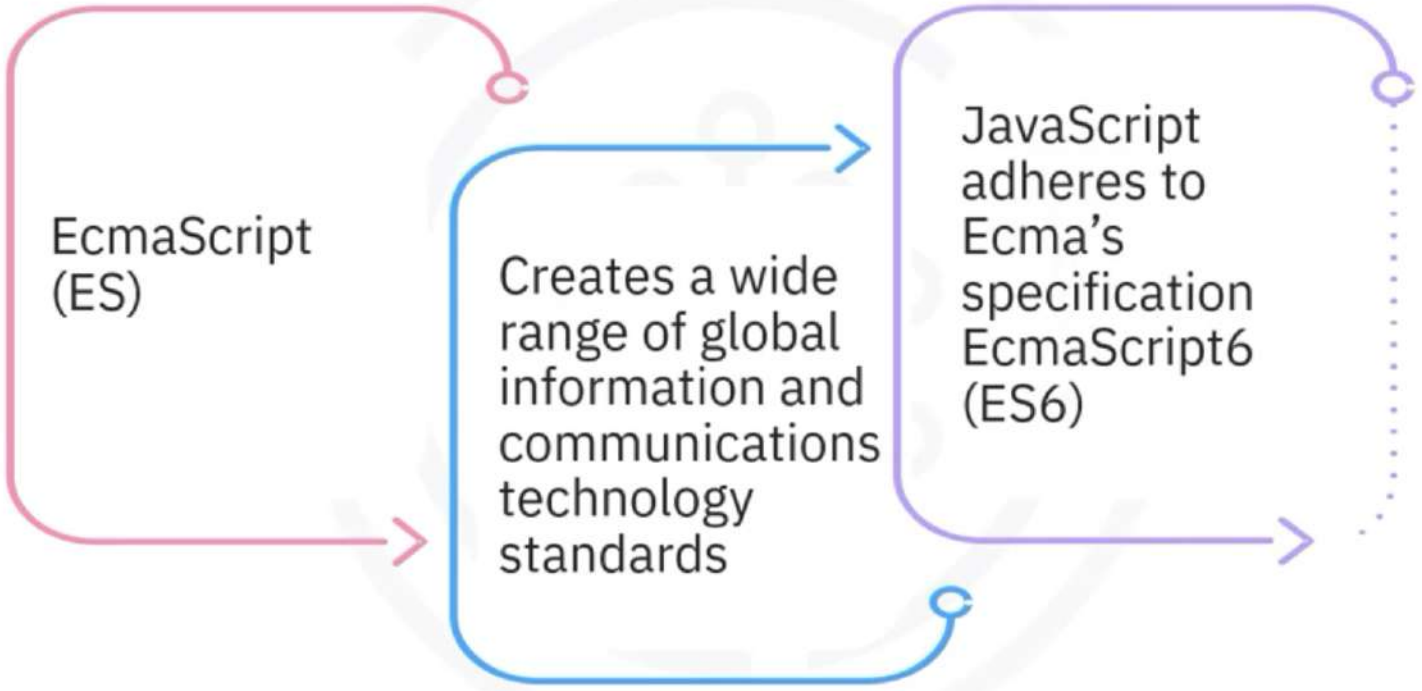
IBM

What you will learn

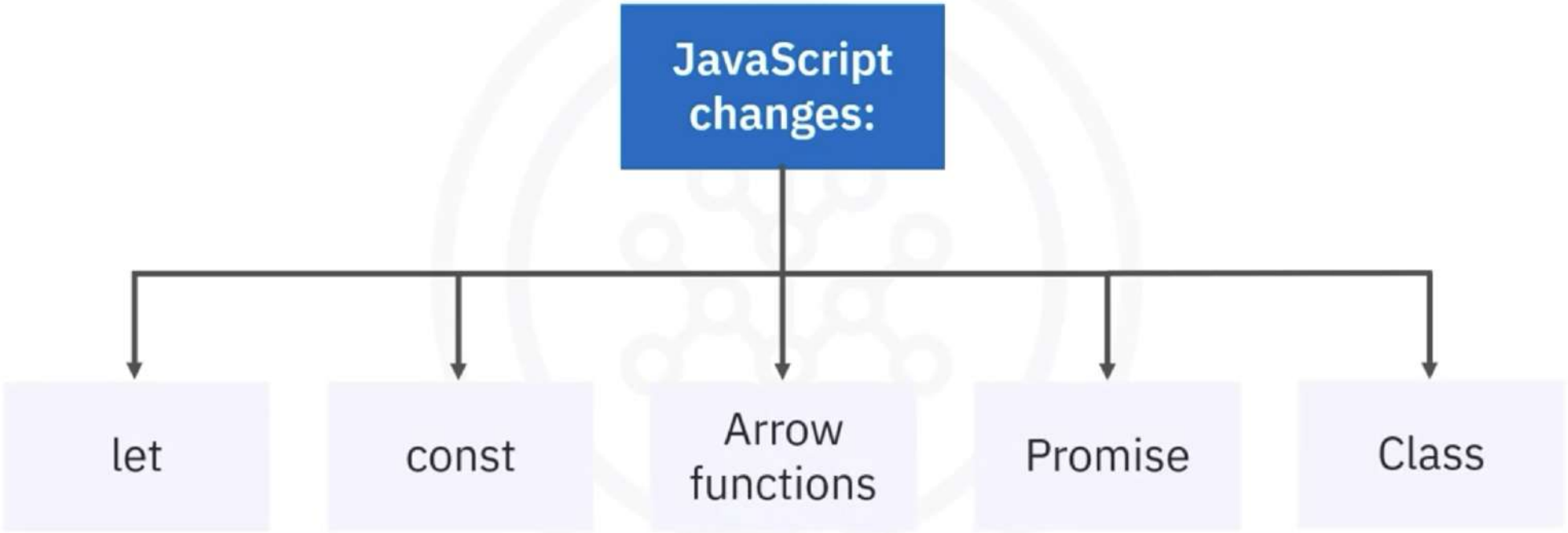
Define EcmaScript6 (ES6)

Describe how to use new features that have been added to JavaScript as part of ES6

ES6 standard



JavaScript ES6: New features



Keywords: let and const

Global scope: `var`

`let`

Local scope

```
{
  let num = 5
  console.log(num);
  num = 6;
  console.log(num);
}
console.log(num)
```

num: out of scope

`const`

Cannot change

```
const num = 5
console.log(num)
num = 6
console.log(num)
```

num: error

Arrow functions

- Similar to calling variables
- Shorter and cleaner way to work with functions

```
function sayHello() {
  console.log("ES6 function - Hello World!")
}
```

```
const sayHello = () =>
  console.log("ES6 function - Hello World!")
```

Calling arrow functions

- Arrow functions can be called
- Arrow functions can be passed as parameters for callbacks
- Pass `sayHello()` as a callback parameter to `setTimeout()`

```
const sayHello = () => console.log("Hello World");
setTimeout(sayHello,1000);
```

Arrow functions with parameters

- Take parameters like normal functions
- Can return a data type or an object

```
const oneParamArrowFunc = name => { return "hello" + name };
```

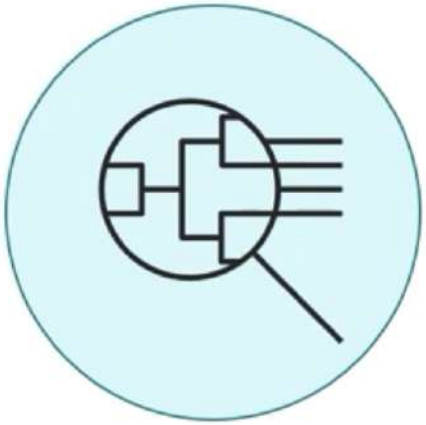
```
const twoParamsArrowFuncWithoutReturn = (first,last)=>
  console.log("hello" + first + " " + last);
```

```
const twoParamsArrowFuncWithReturn = (first, last) => {
  return "hello" +first+ " " + last };
```

```
const twoParamsTwoLinesArrowFunc = (first, last) => {
  const greeting = "hello"; return greeting+" "+first+" " +last; }
```

Promise

- Object which represents the completion of an asynchronous operation and its return value
- After invoking, a promise is in the pending state
- After successful execution, the promise is fulfilled
- If the operation fails, the promise is rejected



Promise example

Function with two parameters:

- **Resolve**: action when fulfilled
- **Reject**: action if it fails

```
let promiseArgument = (resolve,reject) =>{
  setTimeout(()=>{
   let currTime = new Date().getTime();
   if(currTime % 2 === 0){
       resolve("Success!")
     else{ reject("Failed!!!") }
  },2000) };
let myPromise = new Promise(promiseArgument);
```

Promise example

```
let promiseArgument = (resolve,reject) =>{
  setTimeout(()=>{
    let currTime = new Date().getTime();
    if(currTime % 2 === 0){
      resolve("Success!")
    }else{ reject("Failed!!!") }
  },2000) };
```

Classes

- Makes object-oriented programming feasible
- Blueprint for creating objects
- Built on prototypes in JavaScript

```
function Person(name, age) {
  this.name = name;
  this.age = age;
  return this;
}
let person1 = Person("Jason", 20)
console.log(person1)
console.log(person1.name);
console.log(person1.age);
```

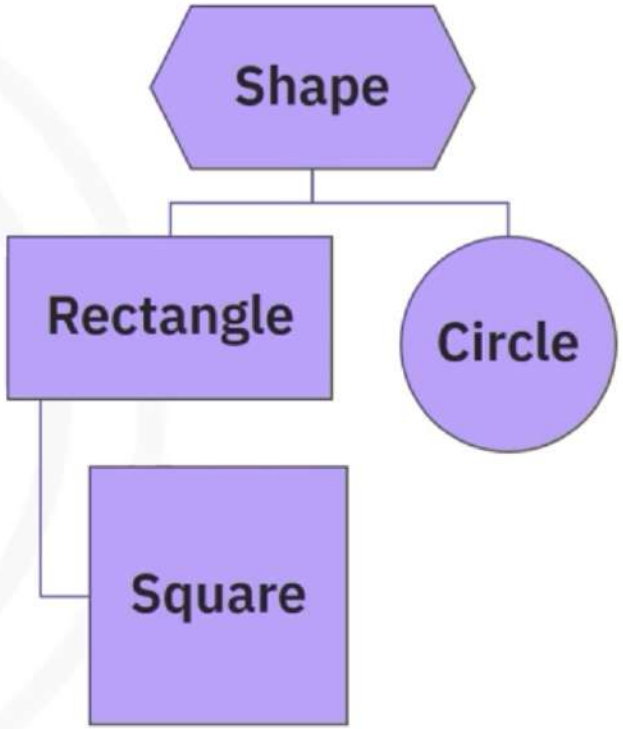
Class

- **Constructor:** method that creates an object of a class
- **Body:** contained in curly brackets
- **Object:** Create using a new keyword

```
class Rectangle {
  constructor(height,width){
    this.height = height;
    this.width = width;
    console.log("Rectangle Created")
    console.log("Height" +this.height);
    console.log("Width" +this.width);
  }
};
let myRectangle = new Rectangle(10,5)
```

Inheritance

- A class can inherit from another class
- The subclass inherits from its class
- The subclass inherits all the attributes and methods of the superclass
- React components use inheritance



Inheritance

- The subclass may call the superclass constructor with a `super()` method call

```
class Square extends Rectangle
{
  constructor(height,width){
    if(height === width){
      super(height,width)
    }else{
      super(height,width);
    }
  }
}
let mySquare = new Square(5,5)
```

Recap

In this video, you learned that:

- New features that were introduced in JavaScript as a part of ES6 are let, const, arrow functions, promise, and class
- You can create different types of arrow functions depending on the parameters, return values, and lines of code
- Object-oriented programming was made feasible in JavaScript with the introduction of class