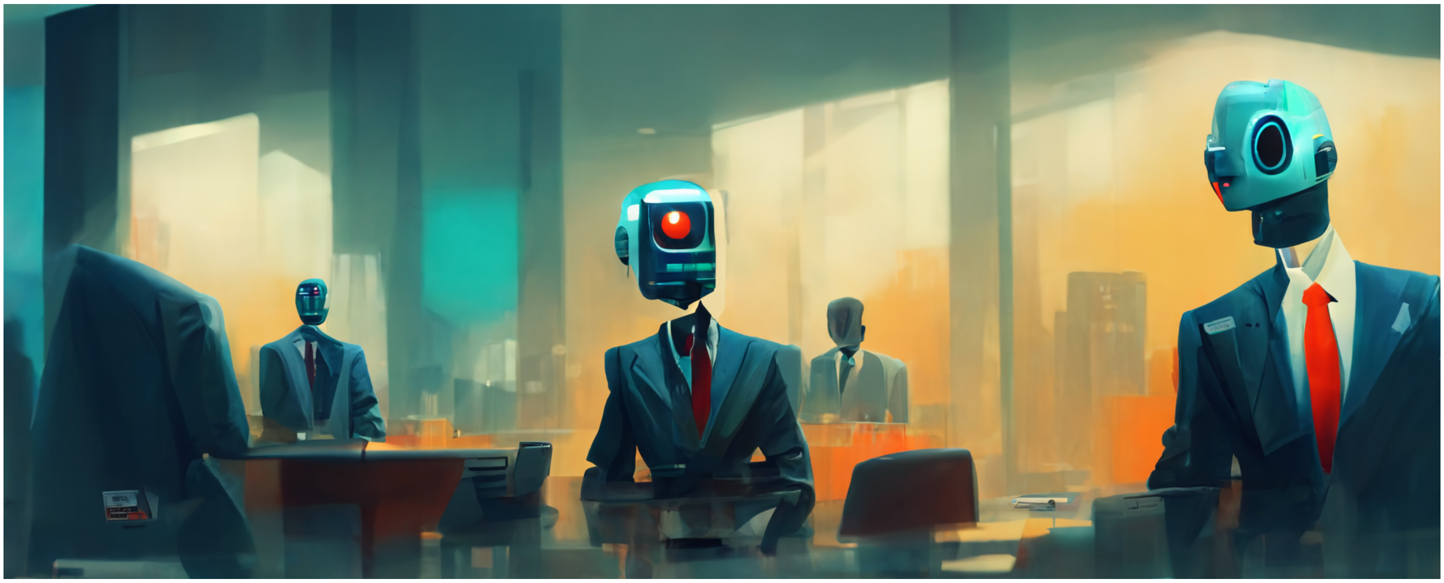
7.11 JS ES6



mechanical workforce - answered by Christian Jay Salvino

- ECMAScript 2015 was the second major revision to Javascript.
- ECMAScript 2015 is also known as ES6 and ECMAScript 6.

New arrow functions

- **Arrow functions** allows a short syntax for writing function expressions.
- You don't need the `function` keyword, the `return` keyword, and the curly brackets.

```
// ES5
var x = function(x, y) {
   return x * y;
}

// ES6
const x = (x, y) => x * y;
```

- Arrow functions do not have their own this. They are not well suited for defining object methods.
- Arrow functions are not hoisted. They must be defined before they are used.
- Using `const` is safer than using `var`, because a function expression is always a constant value.
- You can only omit the `return` keyword and the curly brackets if the function is a single statement. Because of this, it might be a good habit to always keep them:

```
const x = (x, y) => { return x * y };
```

Setting default parameter values

- ES6 allows function parameters to have default values.

```
function myFunction(x, y = 10) {
  // y is 10 if not passed or undefined
  return x + y;
}
myFunction(5); // will return 15
```

Indefinite argument count

- The rest parameter (`...`) allows a function to treat an indefinite number of arguments as an array:

```
function sum(...args) {
  let sum = 0;
  for (let arg of args) sum += arg;
  return sum;
}

let x = sum(4, 9, 16, 25, 29, 100, 66, 77);
```

String functions

Keyword(s)	Description	Sample code	Console output

`includes()`	• Checks if a given substring is present within the source string and returns true if found, or false if not.	`const sample_string = "Hello, World!";`
		
		`const constains_hello = sample_string.includes("Hello");`
		`const contains_goodbye = sample_string.includes("Goodbye");`
		
		`console.log("Contains 'Hello':", constains_hello);`
		`console.log("Contains 'Goodbye':", contains_goodbye);`
		
		`const starts_with_hello = sample_string.startsWith("Hello");`
			`Contains 'Hello': true`
			`Contains 'Goodbye': false`
			`Starts with 'Hello': true`
			`Starts with 'World': false`
			`Ends with '!': true`
			`Ends with ',': false`

`startsWith()`	• Determines whether the source string begins with a specified prefix and returns true if it		

	does, or false if it does not.	`const starts_with_world = sample_string.startsWith("World");`
		
		`console.log("Starts with 'Hello':", starts_with_hello);`
		`console.log("Starts with 'World':", starts_with_world);`	

`endsWith()`	- Checks if the source string ends with a specified suffix and returns true if it does, or false if it does not.	`const ends_exclamation = sample_string.endsWith("!");`
		`const ends_comma = sample_string.endsWith(",");`
		
		`console.log("Ends with '!':", ends_exclamation);`
		`console.log("Ends with ',':", ends_comma);`	

Array functions

Keyword(s)	Description	Sample code	Console output

`from()`	- Creates a new array from an iterable or array-like object.	`const iterable = "Hello";`
		`const char_array = Array.from(iterable);`
		
		`console.log("Array from iterable:", char_array);`
		
		`const keys_array = Array.from(char_array.keys());`
		
		`console.log("Indices of array elements:", keys_array);`
		
		`const numbers = [1, 2, 5, 8, 9];`
		`const evenNumber = numbers.find((num) => num % 2 === 0);`
		
		`console.log("First even number:", evenNumber);`
			`Indices of array elements: (5) [0, 1, 2, 3, 4]`
			`First even number: 2`
			`Index of person with age 25: 1`

`keys()`	- Returns an array iterator containing the indices of an array's elements.		
`find()`	- Returns the first element in an array that satisfies a specified condition or predicate function.		

`findIndex()`	• Returns the index of the first element in an array that satisfies a specified condition or predicate function.	`const people = [`
		`  { name: "Alice", age: 30 },`
		`  { name: "Bob", age: 25 },`
		`  { name: "Charlie", age: 35 },`
		`];`
		
		`const index = people.findIndex((person) => person.age === 25);`
		
		`console.log("Index of person with age 25:", index);`	

Additional Material

- **Learn more**
  - W3Schools