Loops in ES6

- The for...in loop is used to iterate over the properties of an object.
- The for...of loop is used to iterate over the values of an iterable object, such as an array, object, or string.

For in loop example

```
let obj = {
  name: "John",
  age: 30,
};
for (let key in obj) {
  console.log(key); // name, age
}
```

- The for..in loop should not be used to iterate over an array as the JavaScript engine can not guarantee the order of the elements in the array.
- Also instead of iterating variable being number it is string so if we do math over it it will do string concatenation instead of math addition.

For of loop example

- During the for..of loop, the value of the variable is the value of the current element.
- The for...of loop works for both arrays and iterable objects.

```
// for string
let str = "Hello";
for (let char of str) {
  console.log(char); // H, e, l, l, o
}
```

- for...of loop also works on newly introduced collections, such as Map and Set.

Quick Quiz

- Fill in the blanks to complete the following for...of loop statement:

```
___ (let ch __ "javascript) {
  console.log(ch); // j, a, v, a, s, c, r, i, p, t
}
```

Functions in ECMA Script 6

- Prior to ES6, functions were not first class objects.
- In ES6, functions are first class objects.
- In ES6, functions can be used as values.
- In ES6, functions can be passed as arguments to other functions.
- In ES6, functions can be returned from other functions.
- Prior syntax:

```
function sayHello() {
  console.log("Hello");
}
sayHello();
```

- ES6 syntax:

```
const sayHello = () => {
  console.log("Hello");
};
sayHello();
```

- With arguments:
- Prior syntax:

```
function sayHello(name) {
  console.log("Hello " + name);
}
```

- ES6 syntax:

```
const sayHello = (name) => {
  console.log(`Hello ${name}`);
};
sayHello("John");
```

- New syntax is quite handy when you just need a simple function with one argument, we can skip writing braces and return keyword.

```
const greet = (name) => `Hello ${name}`;
```

- for no parameters, empty parentheses are used.

```
const greet2 = () => console.log("Hello");
```

- Syntax is quite usefull for inline functions.
- Lets take an example, we have an array and for each element we want to execute a function.

```
var arr = [1, 2, 3, 4, 5];
arr.forEach(function (element) {
  console.log(element * 2); // 2, 4, 6, 8, 10
});
```

- However in ES6 we can use arrow functions to do the same.

```
arr.forEach((element) => console.log(element * 2));
```

Quick Quiz

- Fill in the blanks to declare an arrow function that takes an array `arr` and returns odd numbers:

```
const printOdds = (___) => {
  ___.forEach((____) => {
    if (element % 2 !== 0) {
      console.log(element);
    }
  });
};
```

Default parameters in ES6

- In ES6, default parameters are added to functions right in the signature of the function.

- for example:

```
function greet(name = "John") {
  console.log(`Hello ${name}`);
}
greet(); // Hello John
greet("Sara"); // Hello Sara
```

- Here's an arrow function with default parameters:

```
const greet = (name = "John") => console.log(`Hello ${name}`);
greet(); // Hello John
greet("Sara"); // Hello Sara
```

- Default value expressions are evaluated at function call time from left to right.
- This also means that default expressions can use the values of parameters that have been passed to the function previously.

Quick Quiz

- What is the output of the following code?

```
function magic(a, b = 30) {
  return a + b;
}
console.log(magic(2));
```

> Answer: 32