Node ES6 Review

JS OOP

Why ‘let’ instead of ‘var’ ?

```
var word = 'I am global';
if(true) {
  var word = 'I am local';
}
console.log(word); // What do you expect here as result?
```

I am local will be printed

Why ‘let’ instead of ‘var’ ?

```
Let word = 'I am global';
if(true) {
  var word = 'I am local';
}
console.log(word); // What do you expect here as result?
```

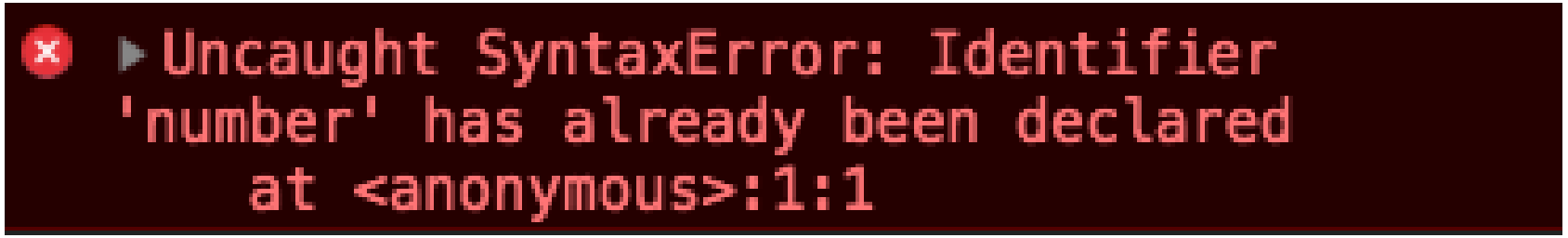
I am global will be printed

Double Declaration

```
var number = 1;

var number = 2;


console.log(number); // No errors here, 2 gets printed
```

Double Declaration

```
let number = 1;

let number = 2;


console.log(number); // let doesn't allow redeclaration
```



Const keyword

```
const birthYear = 1990;

birthYear = 2000; // You cannot re-assign a constant variable
```

Template Literals

```
let oldWay = 'A word';  // JS Way

let newWay = `A word`;  // ES6 Way
```

Old Way

```
let name = 'Cem';

let age = 28;

let profession = 'Software Developer';

console.log("Hello, my name is " + name + ", I'm " + age + " years old and I'm a " + profession);
```

The ES6 way

```
console.log(`Hello, my name is ${name}, I'm ${age} years old and I'm a ${profession}.`);
```

Arrow Functions

```
var sum = function addition (firstNum, secondNum) {
    return firstNum + secondNum;
}
let sum = (firstNum, secondNum) => { return firstNum + secondNum };
```

Spread Operator — Usage for Arrays

```
let numberArray = [1, 19, 21, 85, 42]

console.log(...numberArray);

let charArray = ['a','b','c'];

charArray.push(...numberArray);


console.log(charArray);
```

Import & Export

```
export let myFunction = () => { console.log('I am exported!');
}
```

Then

```
import { myFunction } from './yourFolderPath/fileName';
```

x => x * 2

```
function(x){
 Return x*2;
}
```

Array.prototype.map()

```
const array1 = [1, 4, 9, 16];// pass a function to map

const map1 = array1.map(x => x * 2);

console.log(map1);

// expected output: Array [2, 8, 18, 32]
```

concat

```
var hege = ["Cecilie", "Lone"];
var stale = ["Emil", "Tobias", "Linus"];
var children = hege.concat(stale);
```

filter

```
var ages = [32, 33, 16, 40];


function checkAdult(age) {
  return age >= 18;
}
ages.filter(checkAdult)
//Return an array of all the values in the ages array that are 18 or over:
```

find

```
var ages = [32, 33, 16, 40];


function checkAdult(age) {
  return age >= 18;
}
ages.find(checkAdult)
//Get the value of the first element in the array that has a value of 18 or more:
```

findIndex

```
var ages = [32, 33, 16, 40];


function checkAdult(age) {
  return age >= 18;
}
ages. findIndex(checkAdult)
//Get the index of the first element in the array that has a value of 18 or more:
```

forEach

```
var fruits = ["apple", "orange", "cherry"];
fruits.forEach(myFunction);

function myFunction(item, index) {
  console.log(item);
  console.log(index);
}
```

includes

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var n = fruits.includes("Mango");
```

splice() // very important

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(2, 0, "Lemon", "Kiwi");
```

Inputs

- Index: // from which index to operate
- Howmany: items to remove
- New items // new items to add. this is optional