GOMYCODE.TN

ES6 - Cheat Sheet

Arrow Function

```
const sum = (a,b) => a + b;

console.log(sum(2,6)) // prints 8
```

Default Parameters

```
function print(a = 5) {

        console.log(a)

}

print() // prints 5

print(22) // prints 22
```

Let Scope

```
let a = 3

if (true) {

      let a = 5

      console.log(a) // prints 5

    }
console.log(a) // prints 3
```

Const

```
// can be assigned only once

       const a = 55

   a = 44 // throws an error
```

Multi-line String

```
 console.log(
    `This is a
    multi-line string`
)
```

Template String

```
const names = "World"
const message = `Hello ${names}`
console.log(message)

// prints "Hello World"
```

Exponent Operator

```
const byte = 2 ** 8

// expected result = 256

// Same as: Math.pow(2, 8)
```

Spread Operator

```
const a = [ 1, 2 ]

const b = [ 3, 4 ]

const c = [ ...a, ...b ]


console.log(c)

// [1, 2, 3, 4]
```

String Includes()

```
console.log('scripts'.includes('scripts'))

// prints true

console.log('scripts'.includes('prints'))

// prints false
```

// The includes() method is case sensitive.

For example, the following expression returns false:

```
console.log('scripts'.includes('Scripts'))

// prints false
```

String startsWith()

```
console.log('scripts'.includes('sc'))

// prints true


console.log('scripts'.includes('rt'))

// prints false
```

String repeat()

```
console.log('st'.repeat(3))

// prints "ststst"
```

Destructuring array

```
let [a,b] = [3,7];

console.log(a); // 3

console.log(b); // 7
```

Destructuring object

```
let obj = {

    a: 77,

    b: 66

};

let { a,b } = obj;

console.log(a); // 77

console.log(b); // 66
```

Object Property Assignment

```
const a = 2  const b = 5

const obj = { a, b }

// Before es6:
// obj = { a: a, b:b }

console.log(obj)

// prints { a:2, b:5 }
```

Object.assign()

```
const obj1 = { a: 1 }

const obj2 = { b: 2 }

const obj3 = Object.assign({},

    obj1, obj2)

console.log(obj3)

// { a: 1, b: 2 }
```

Promises with finally

```
Promise

    .then((result) => { ... })

    .catch((error) => { ... })

    .finally(() => {

       /* logic independent of success/error */
       })
```

/* The handler is called when the promise is fulfilled or rejected. */

Spread Operator

```
const a ={
  firstName: "FirstName",
  lastName: "LastName1",
 }

const b ={
  ...a,
  lastName: "LastName2",
  canSing: true,
 }

console.log(a)
//{firstName:"FirstName",lastName:"LastName1"}
console.log(b)
/* {firstName: "FirstName", lastName: "LastName2",
 canSign: true} */
/* great for modifying objects without side effects/affecting the original */
```

Destructuring Nested objects

```
const Person ={
    name: "Rezaul karim",
    age: 23,
    sex: "male",
    maritalstatus: "single",
    address: {
     country: "BD",
     state: "Dhaka",
     city: "N.Ganj",
     pincode: "123456",
    },
  };

const { address: { state, pincode }, name } = Person;
```

```
console.log(name, state, pincode)
//Rezaul Karim Dhaka 123456
console.log(city) // ReferenceError
```

Object function assignment

```
const obj = {
    a: 5,
    b(){
    console.log('b')
    }
  }

obj.b() // prints "b"
```

Object.entries()

```
const obj = {
firstName: "FirstName",
lastName: "LastName1",
age: 23,
country: "Bangladesh",
};

const entries = Object.entries(obj);
console.log(entries)
 /* prints [
    ['firstName', 'FirstName'],
    ['lastName', 'LastName'],
    ['age', 23],
    ['country', 'Bangladesh']
  ];    */
```