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
```

Multiline string

```
console.log(`
    This is a
    multiline string
`)
```

Template strings

```
const name = 'Leon'
const message = `Hello ${name}`

console.log(message) // prints "Hello Leon"
```

String includes()

```
console.log('apple'.includes('pl')) // prints true
console.log('apple'.includes('tt')) // prints false
```

String startsWith()

```
console.log('apple'.startsWith('ap')) // prints true
console.log('apple'.startsWith('bb')) // prints
false
```

String repeat()

```
console.log('ab'.repeat(3)) // prints "ababab"
```

Destructuring array

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

console.log(a); // 3
console.log(b); // 7
```

Destructuring object

```
let obj = {
      a: 55,
      b: 44
};

let { a, b } = obj;

console.log(a); // 55
console.log(b); // 44
```

object property assignement

```
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 function assignement

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

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

spread operator

```
const a = [ 1, 2 ]
const b = [ 3, 4 ]

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

console.log(c) // [1, 2, 3, 4]
```

Object.assign()

```
const obj1 = { a: 1 }
const obj2 = { b: 2 }

const obj3 = Object.assign({}, obj1, obj2)

console.log(obj3) // { a: 1, b: 2 }
```

Object.entries()

```
const obj = {
      firstName: 'Vipul',
      lastName: 'Rawat',
      age: 22,
      country: 'India',
};

const entries = Object.entries(obj);
/* returns an array of [key, value]
       pairs of the object passed
*/

console.log(entries);
/* prints
       [
             ['firstName', 'Vipul'],
             ['lastName', 'Rawat'],
             ['age', 22],
             ['country', 'India']
       ];
*/
```

spread operator

```
const a = {
		firstName: "Barry",
		lastName: "Manilow",
}

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

console.log(a) // {firstName: "Barry", lastName: "Manilow"}

console.log(b) // {firstName: "Barry", lastName: "White", canSing: true}

// great for modifying objects without side effects/affecting the original
```

Destructuring Nested Objects

```
const Person = {
	name: "John Snow",
	age: 29,
	sex: "male",
	materialStatus: "single",
	address: {
		country: "Westeros",
		state: "The Crownlands",
		city: "Kings Landing",
		pinCode: "500014",
	},
};

const { address : { state, pinCode }, name } = Person;

console.log(name, state, pinCode) // John Snow The Crownlands 500014
console.log(city) // ReferenceError
```

Exponent operator

```
const byte = 2 ** 8

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

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.
```