JAVASCRIPT CHEAT SHEET

FUNCTIONS, SCOPES, LOOP, ARRAYS.

by meak
@meakcodes

1.0

FUNCTIONS

1. **The function adds** *two numbers* **together:**

```javascript
// Defining the function:
function sum(num1, num2) {
  return num1 + num2;
}

// Calling the function:
sum(3, 6); // 9
```

2. **The rocketToMars function returns** *'BOOM!'*.
There are two ways to write it:
as a named function or as a function stored in a variable.

```javascript
// Named function
function rocketToMars() {
  return 'BOOM!';
}

// Anonymous function
const rocketToMars = function() {
  return 'BOOM!';
}
```

FUNCTIONS

3. **The checkWeight function logs the weight provided as an argument. When called with** *checkWeight(25)* **it prints** *"Weight: 25"* **to the console.**

```
const checkWeight = weight => {
  console.log(`Weight : ${weight}`);
};
checkWeight(25); // => Weight : 25
```

4. **The sum function adds two numbers together. When called with** *console.log(sum(2,5))* **it prints** *7* **to the console.**

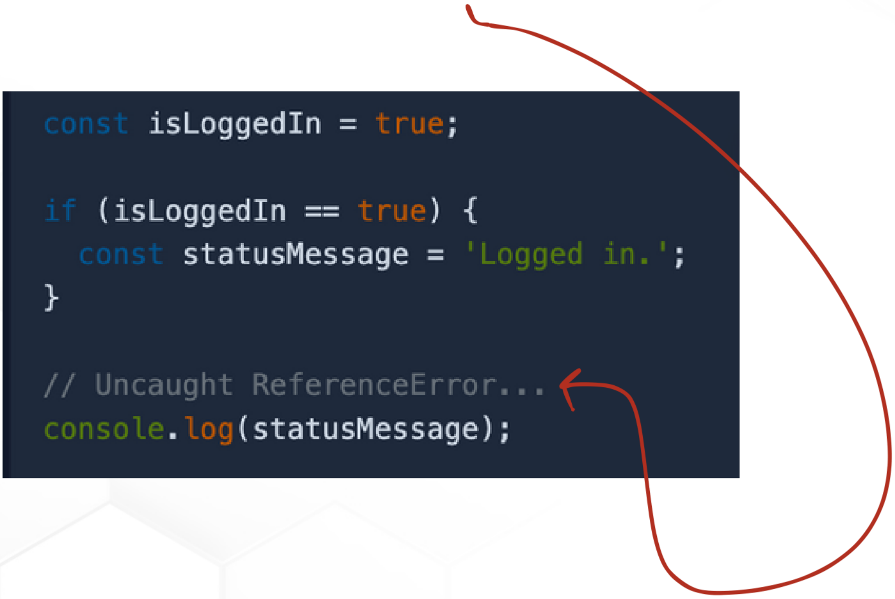
```
const sum = (param1, param2) => {
  return param1 + param2;
};
console.log(sum(2,5)); // => 7
```

SCOPES 1.0

1. **The isLoggedIn variable is set to true. Inside the if statement, the statusMessage variable is declared with the message** *'Logged in.'*

However, since it's declared inside the block, it's not accessible outside of it.

Therefore, attempting to log statusMessage outside the if block will result in a -> *ReferenceError.*



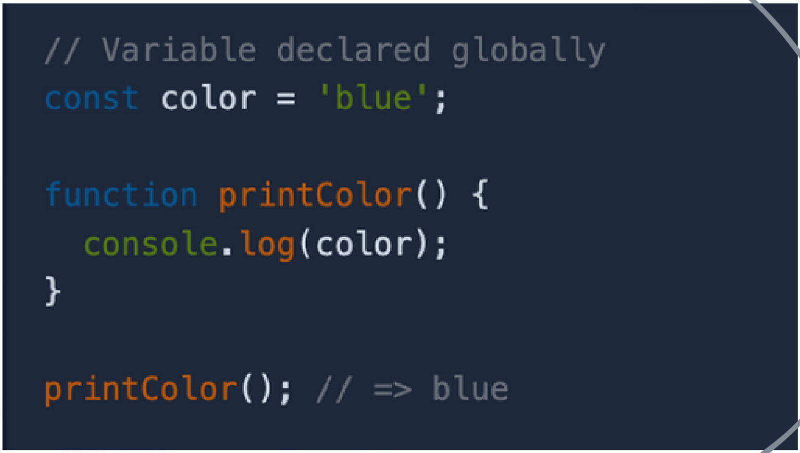
```
const isLoggedIn = true;

if (isLoggedIn == true) {
  const statusMessage = 'Logged in.';
}

// Uncaught ReferenceError...
console.log(statusMessage);
```

1.1

SCOPES

2. **The function printColor logs the value of the globally declared variable color, which is 'blue'.**

When *printColor()* **is called, it prints** 'blue' **to the console.**



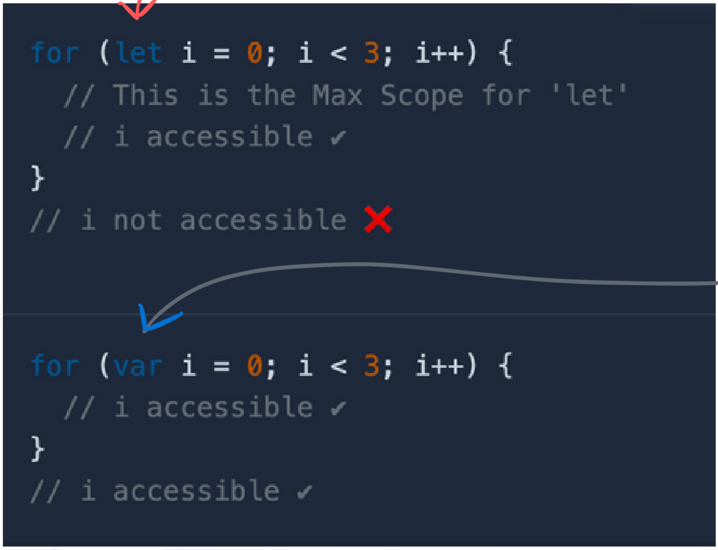
```javascript
// Variable declared globally
const color = 'blue';

function printColor() {
  console.log(color);
}

printColor(); // => blue
```

// Function prints the value of the globally declared variable 'color'
// When called, it logs 'blue' to the console ✓

SCOPES 1.3

3. In the first loop, *i* is declared with let, making it accessible only within the loop's scope. After the loop, *i* is ***not*** accessible.

In the second loop, *i* is declared with *var*, making it accessible both within and outside the loop's scope.



```
for (let i = 0; i < 3; i++) {
  // This is the Max Scope for 'let'
  // i accessible ✔
}
// i not accessible ❌

for (var i = 0; i < 3; i++) {
  // i accessible ✔
}
// i accessible ✔
```

By the way, my recommendation: Please do not use “var” anymore.

ARRAYS 1.0

1. **<u>This code declares two arrays:</u>**
   - **one with "*strings*" representing fruits**
   - **another with *<u>different data types</u>* including numbers, strings, and a boolean value.**

```javascript
const fruits = ["apple", "orange", "banana"

// Different data types
const data = [1, 'chicken', false];
```

2. **<u>This code declares an array of numbers</u> and retrieves its length, which is *<u>4.</u>***

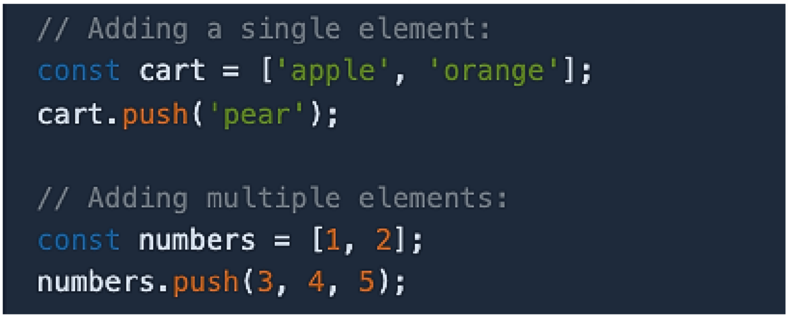
```javascript
const numbers = [1, 2, 3, 4];

numbers.length // 4
```

3. **<u>This code creates an array of numbers</u> and then prints the elements *at index 0 and index 1* using console.log**

```javascript
// Accessing an array element
const myArray = [100, 200, 300];

console.log(myArray[0]); // 100
console.log(myArray[1]); // 200
```

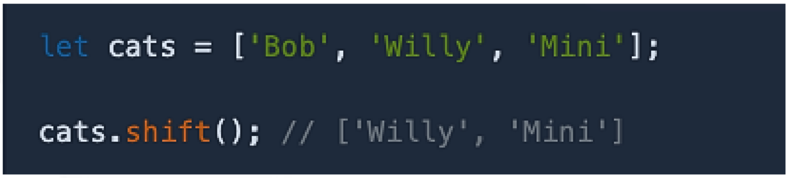
ARRAYS 1.1

4. **<u>This code adds one item, "pear,"</u> to the cart array and three items,** *3, 4, and 5,* **to the numbers array.**



```
// Adding a single element:
const cart = ['apple', 'orange'];
cart.push('pear');

// Adding multiple elements:
const numbers = [1, 2];
numbers.push(3, 4, 5);
```

5. **<u>This code removes the first element</u> from the cats array,** *leaving ['Willy', 'Mini'].*



```
let cats = ['Bob', 'Willy', 'Mini'];

cats.shift(); // ['Willy', 'Mini']
```

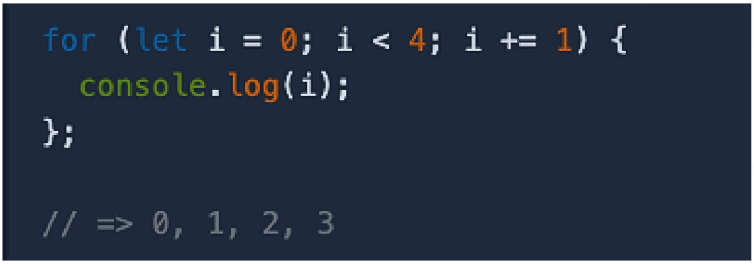
leaving ['Willy', 'Mini']. ✓

‘Bob’ is gone.

1.0

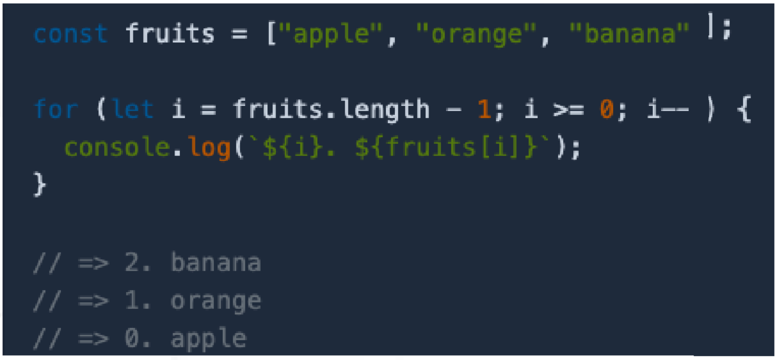
LOOPS

1. **This code loops from 0 to 3, printing** *each number* **to the console.**



```
for (let i = 0; i < 4; i += 1) {
  console.log(i);
};

// => 0, 1, 2, 3
```

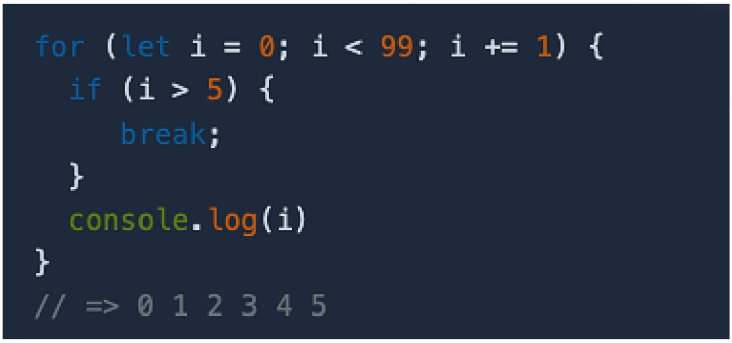
2. **This code loops through the fruits array in reverse order,** *printing each index* **and its corresponding fruit.**



```
const fruits = ["apple", "orange", "banana" ];

for (let i = fruits.length - 1; i >= 0; i-- ) {
  console.log(`${i}. ${fruits[i]}`);
}

// => 2. banana
// => 1. orange
// => 0. apple
```

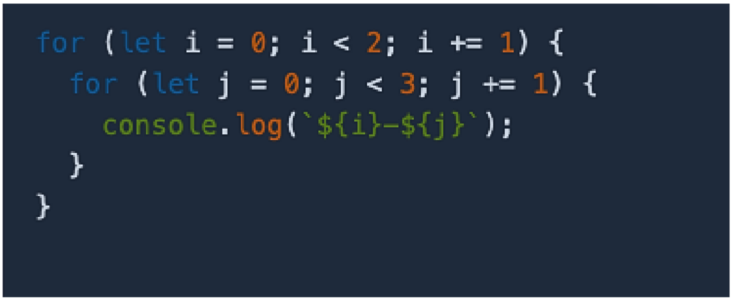
LOOPS

3. **<u>This code loops from 0 to 98</u> but stops and exits the loop when** i *becomes greater than 5*, **printing the numbers** *0 through 5.*



```
for (let i = 0; i < 99; i += 1) {
  if (i > 5) {
    break;
  }
  console.log(i)
}
// => 0 1 2 3 4 5
```

4. **<u>This code uses nested loops</u> to** print pairs **of indices, resulting in the output:**



```
for (let i = 0; i < 2; i += 1) {
  for (let j = 0; j < 3; j += 1) {
    console.log(`${i}-${j}`);
  }
}
```

```
0-0
0-1
0-2
1-0
1-1
1-2
```

PLEASE READ

THIS IS THE END

THANK YOU FOR DOWNLOADING THIS SHORT EBOOK.

I HOPE IT HELPED YOU UNDERSTAND THESE CONCEPTS BETTER.

I'M PLANNING TO CREATE MORE EBOOKS IN THE FUTURE, COVERING TOPICS LIKE OBJECTS OR CLASSES IN JAVASCRIPT.

IF YOU'D LIKE MORE EBOOKS LIKE THIS, PLEASE LEAVE A REVIEW ON GUMROAD AND SHARE YOUR THOUGHTS!

THANKS,
MEAK

Hey I'm Meak

A frontend engineer and tech content creator.

I make unique tech content.

110K+ followers **1M+** likes **7M+** views

By the way, I'm working on something big for developers, but *psst* don't tell anyone ;)

if you are interested leave me a DM now.