Javascript Beginner Tutorial Cheatsheet

By: supersimple.dev
Tutorial link: https://www.youtube.com/watch?v=bArwRwHey6c

Intro

Programming = giving instructions to a computer.
The computer will follow your instructions exactly.

- Here are the Javascript instructions we learned in this course
- Use these instructions and combine them into more complex instructions
- Eventually, the complex instructions come together to form a full software application

`alert('hello');`	Opens a popup with the text "hello" inside
`console.log('hi');`	Displays the text "hi" in the Chrome console
`document.body.innerHTML = 'hello';`	Replaces entire web page with the text "hello"

`<button>Click</button>` Creates a button with the text "Click" inside

```
<button>Click</button>
```

opening tag **closing tag**

```
<script>
  alert('hello');
</script>
```

Runs the Javascript code inside the `<script>` tags

`'hello'`
`"hello"`

String = a value in Javascript that represents a piece of text
Strings can be written like `'...'` or `"..."`

`let myVar;`	Variable = a container that can store a value
`let name = 'Simon';`	Creates a variable named `name` that stores the string `'Simon'`
`let age = 27;`	Creates a variable named `age` that stores the number `27`
	Access the value inside a variable by typing the variable name
`console.log(name);`	Will `console.log('Simon');` since `name` is a variable
`console.log('name');`	Will `console.log('name');` since `'name'` is just a string
`typeof 'hello'`	Returns the type of the value: `'string'`
`typeof 27`	`'number'`
`typeof name`	Returns the type of the value inside the variable: `'string'`
`'Hello ' + 'World'`	Combine 2 strings together: `'Hello World'`
`4 + 5 * 3`	Do math on numbers: `19`
`(4 + 5) * 3`	`27`
`(4 - 2) / 2`	`1`

```
<html>
  <head>
    <title>Title</title>
  </head>
  <body>
    <button>Click</button>
    ...
  </body>
</html>
```

All code for the website should be inside `<html></html>`
`<head>` contains information about the page
`<title>` sets the title in the tabs

`<body>` contains everythings that appears on the page

```
<div>Text</div>
```

Creates an invisible box on the page, containing whatever is inside the `<div>` tags (can be text or other HTML elements)

```
<div>
  <button>Button1</button>
  <button>Button2</button>
</div>
```

This `<div>` displays an invisible box containing 2 buttons

```
let div = document.createElement('div');
```

Creates an HTML `<div>` element and saves it in a variable

```
div.innerText = 'Hello';
document.body.appendChild(div);
```

Sets the text inside the div (`<div>Hello</div>`)
Adds the `<div>` to the end of `<body>`

```
function addTodo() {
  console.log('Add Todo');
}
addTodo();
```

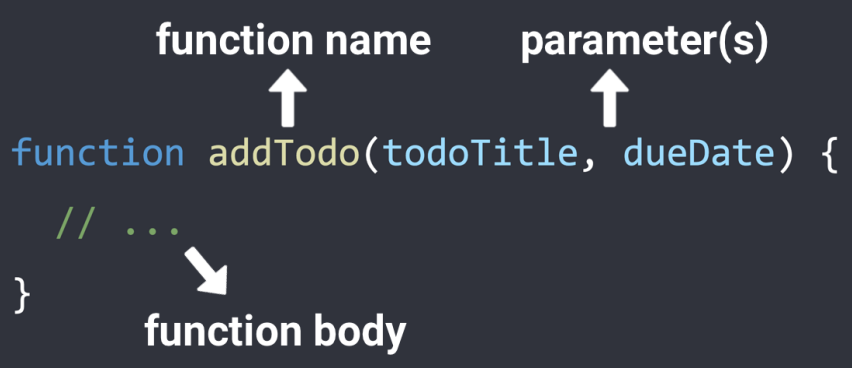
Creates a function named `addTodo`

Runs the code inside the function

```
function addTodo(todoTitle) {
 console.log(todoTitle);
}
addTodo('Wash car');
```

Create a parameter named `todoTitle` (parameter = variable that can be used inside the function)

Set the value of the parameter to `'Wash car'`



Function Terminology:

1. Creating a function = "defining" a function
2. Running a function = "calling" a function
3. `function addTodo(todoTitle) {}` = the function "takes" a parameter named `todoTitle`
4. `addTodo('Wash car');` = we "pass in" an "argument" (a value `'Wash car'`) to the function

---

```
let todos = [
 'Get groceries',
 'Wash car'
];
```

Array = a list of values
Creates an array of string values and saves it in a variable named `todos`

```
todos.push('new todo');
todos.pop();
```

Adds new value to the end of the array
Removes the last value from the array

```
todos.push('new todo');
```

↓

method

Method = function attached to a value

```
'hello'.toUpperCase();
```

Gives the uppercase version of the string

```
todos.forEach(function (title) {
  // ...
});
```

Loop = for each value in the array, it will:
1. Save the value in the `title` parameter
2. Run the inner function

```
<button disabled="true">Click</button>
<button hidden="true">Click</button>
```

HTML attribute = modifies the appearance and behaviour of HTML elements

```
<button onclick="console.log('hi')">
  Click
</button>
```

onclick attribute = runs the code inside the quotes `"..."` when button is clicked

```
<input type="text" />
<input type="checkbox" />
<input type="date" />
```

Creates a text box
Creates a checkbox
Creates a date picker

```
<input id="text-box" type="text" />
```

id attribute = allows you to get the element and manipulate it using JavaScript

---

```
let textbox = document.getElementById('text-box');
```

Gets the HTML element using its `id` and saves it in a variable

```
let value = textbox.value;
```

Gets the text that's currently inside the textbox and saves it in a variable

```
// comment
/*
  multi line
  comment
*/
```

Add comments to add extra info. They will be ignored by the computer

```
<div id="my-div"></div>
<script>
  let div = document.getElementById('my-div');
  let button = document.createElement('button');

  div.appendChild(button);
  div.innerHTML = 'hello';

  div.innerHTML = '';
</script>
```

Add element inside `<div id="my-div"></div>`
Erases everything inside `<div id="my-div"></div>` and replaces it with what's inside the string (`'hello'`)
To erase everything in the div, just use empty string `''`

```
let num = 5;
num = 6;
let name;

const num = 5;

var num = 5;
```

Creates a variable that can be reassigned

Create a variable without assigning it a value

Creates a variable that can not be reassigned (constant)

Old way of defining a variable (generally not used anymore)

```
let person = {
  name: 'Taylor',
  age: 27
};
```

property (key)

value

Object = a value in JavaScript that groups related values together

```
person.name
person.age
person['name']

const property = 'name';
person[property];
```

Get the value of the name property: `'Taylor'`
`27`
Same as `person.name` write it like this if the property contains hyphens: `person['first-name']`

Same as `person['name']`

```
<button style="background-color: red;
font-size: 18px">Click</div>
```

Style attribute = changes the appearance of the HTML element using CSS

CSS syntax

```
style="background-color: red; font-size: 18px"
```

style name **style value**

```
const btn = document.createElement('button');
btn.style = 'background-color: red';
btn.disabled = true;
btn.hidden = true;
```

Set HTML attributes using JS. This is a feature of the Document Object Model (DOM)

```
const id = new Date().getTime();
btn.id = id;
btn.onclick = function () {};
```

Gets the current time in milliseconds
Set the button's id using JavaScript
Set the onclick attribute

```
const id = btn.id;
```

Get the value of an attribute

```
function onClick(event) {

  event.target;
}
btn.onclick = onClick;
```

When the button is clicked, this function gets an `event` object containing information about the click event
Gets the HTML element that was clicked

```
function func() {
 return 100;
}
const result = func();
```

Sets the return value to `100`

Save return value in a variable

```
function func() {
 return 'one hundred';
}
```

Sets the return value to a string

```
true
false
```

Boolean = a JavaScript value representing whether something is true or false

```
typeof true
```

'boolean'

```
1 < 5
1 > 5
1 === 5
1 !== 5
1 >= 5
1 <= 5
```

Comparison operators = compares 2 values and returns whether the result is `true` or `false`

```
let result = 1 < 5;
```

boolean values can be saved in a variable

```
'apple' > 'banana'
```

Alphabetical comparison. Returns `false`

```
if (result) {
  console.log('If 1');
} else if (result2) {
  console.log('If 2');
} else {
  console.log('Else');
}
```

Runs this line if `result` is `true`

Otherwise, runs this line if `result2` is `true`

Runs this line if all conditions are `false`

```
let array = [1, 3, 5, 7, 9];
array.filter(function (num) {
 if (num > 5) {
   return true;
 } else {
   return false;
 }
});
```

`filter` = creates a new copy of the array and loops through each value. If the inner function returns `true` keep the value in the new copy. If the inner function returns `false` remove the value from the new copy.

Does not modify the original array.

```
+'99'
'' + 99
```

Convert a string to a number
Convert a number to a string

MVC (Model View Controller)

We split our code into 3 sections:
1. Model = saves and manages data
2. View = manages how to render the data onto the web page
3. Controller = responds to interactions (user clicks a button), tell the model and view to update
More info: https://en.wikipedia.org/wiki/Model–view–controller

MVC Flow

1. View code renders the web page using the data in the Model
2. User clicks a button on the web page
3. Controller code responds to the click and then tells the Model to update its data
4. After the Model updates, the Controller tells View to re-render using the updated data

---

```
const personString = JSON.stringify({
 name: 'Taylor'
});
```

Converts an object into a string `'{"name":"Taylor"}'`

```
const personObject = JSON.parse(personString);
```

Converts a stringified object back into an object

```
localStorage.setItem('key', 'data');
localStorage.getItem('key');
Array.isArray(myVar);
```

Saves a string in browser under the key: `'key'`
Retrieves the string saved under `'key'`
Checks if a variable is an array

```
const func = function () {
 console.log('hi');
};
```

Alternative way of defining a function

```
const func = () => {};
const func = name => {
  console.log(name);
};
```

Arrow functions
Brackets are optional if only 1 parameter

```
const func = (name, age) => {
  console.log(name);
  console.log(age);
};
```

Brackets are required if 2 or more parameters

```
const func = () => 'hi';
```

If all on one line, `return`s the value after =>

```
const createCounter = function () {
 let count = 0;
 return function () {
   count = count + 1;
   console.log(count);
 };
};
```

Usually when a function `return`s, all variables inside the function are deleted.

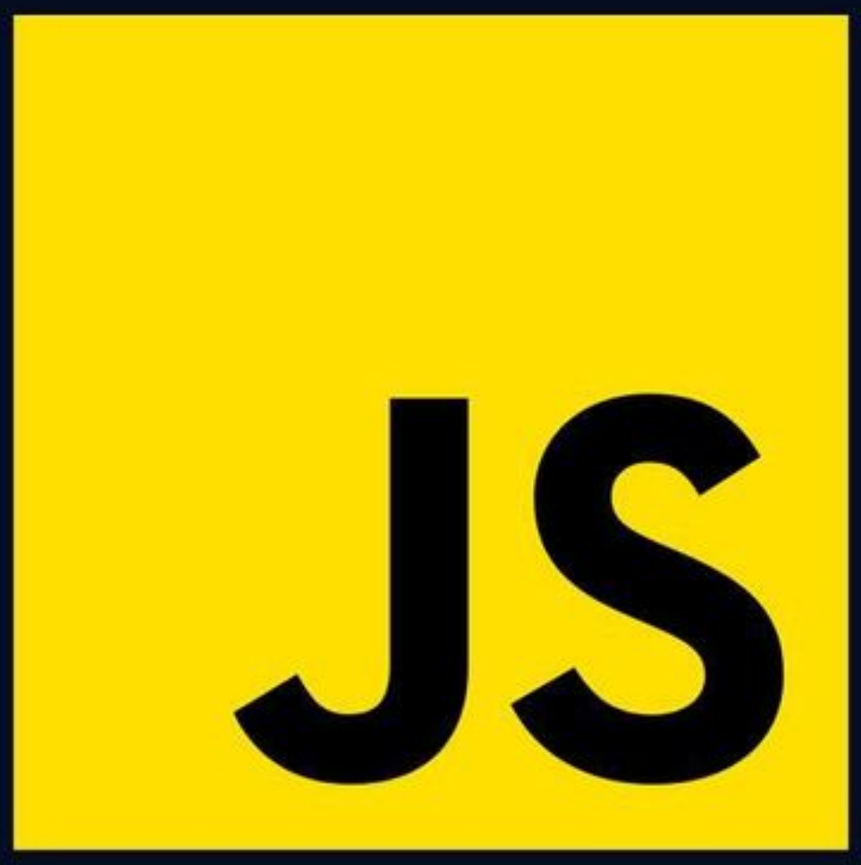
Closure = return a function from a function. The inner function will have <u>permanent</u> access to the variables in the outer function (`count`).

```
const count = createCounter();
count();
```

This will return a function. Calling the function will `console.log(1);`

JS

Advance JS Cheat sheet



Saad Irfan
@DevWithSaad

JS

Arrays in JavaScript

```
const fruits = ['🍎', '🍌', '🍓', '🍑', '🍒', '🍎'];

// converts the array to a string
fruits.toString(); // 🍎,🍌,🍓,🍑,🍒,🍎

// adds element at the end of the array
fruits.push('🍋'); // ['🍎', '🍌', '🍓', '🍑', '🍒', '🍎', '🍋']

// removes the last element of the array
fruits.pop(); // '🍋'

// checks if the array contains an element
fruits.includes('🍌'); // true

// returns the index of the element
fruits.indexOf('🍑'); // 3

// join the elements of the array with the given separator
fruits.join('+'); // 🍎+🍌+🍓+🍑+🍒+🍎

// return a portion of the array
fruits.slice(1, 3); // ['🍌', '🍓']

// add elements to the array
fruits.splice(1, 0, '🥑'); // ['🍎', '🥑', '🍌', '🍓', '🍑', '🍒', '🍎']
```

Saad Irfan

@DevWithSaad

JS

Objects in JavaScript

index.js

```
const person = {
    name: 'John',
    age: 30,
    gender: 'male',
};

const jobObject = {
    job: 'developer',
    salary: 1000,
};

// get all object keys
Object.keys(person); // ['name', 'age', 'gender']
// get all object values
Object.values(person); // ['John', 30, 'male']
// get all object entries
Object.entries(person); // [ [ 'name', 'John' ], [ 'age', 30 ], [ 'gender', 'male' ]]

// assign object to another object
Object.assign(person, jobObject);
// { name: 'John', age: 30, gender: 'male', job: 'developer', salary: 1000 }
```

Saad Irfan
@DevWithSaad

JS

Scope in JavaScript

```javascript
/* global scope */
const PIE = 3.14;

function foo() {
    console.log(PIE); // 3.14

    /* function scope */
    const age = 32;
    console.log(age); // 32
}

/* block scope */
if (true) {
    const fullName = 'John Doe';
    console.log(fullName); // John Doe
}

console.log(PIE); // 3.14
console.log(age); // ReferenceError: age is not defined
console.log(fullName); // ReferenceError: fullName is not defined
```

Saad Irfan
@DevWithSaad

JS

Date in JavaScript

index.js

```
const date = new Date(); // 2023-01-22T09:44:48.175Z
date.getDate(); // month's date: 22
date.getMonth(); // Month with 0 index: 0
date.getFullYear(); // Year: 2023
date.getHours(); // Hours: 9
date.getMinutes(); // Minutes: 44
date.getSeconds(); // Seconds: 48
date.getMilliseconds(); // Millisecond: 175
date.getTime(); // Time: 1648101488175
date.setDate(23); // Set date: 23
date.setMonth(3); // Set month: 3
date.setFullYear(2024); // Set year: 2024
date.setHours(10); // Set hours: 10
date.setMinutes(45); // Set minutes: 45
date.setSeconds(49); // Set seconds: 49
date.setMilliseconds(176); // Set Milliseconds: 176
date.setTime(1648101488176); // Set time: 1648101488176
```

Saad Irfan
@DevWithSaad

JS

Events in JavaScript

index.html

```
<!— when use clicks —>
<input type="text" onclick="" />
<!— when user double clicks —>
<input type="text" ondblclick="">
<!— when user moves the mouse over an element, it's called mouse down—>
<input type="text" onmousedown="">
<!— when an element looses focus —>
<input type="text" onblur="">
<!— when an element gets focus —>
<input type="text" onfocus="">
<!— when a user moves the mouse over an element —>
<input type="text" onmouseover="">
<!— when a user moves the mouse out of an element —>
<input type="text" onmouseout="">
<!— when there is a change —>
<input type="text" onchange="">
<!— when a user presses a key —>
<input type="text" onkeydown="">
<!— when a user releases a key —>
<input type="text" onkeyup="">
<!— when a user presses a key —>
<input type="text" onkeypress="">
<!— when a user submits a form —>
<form onsubmit=""></form>
<!— when a user resets a form —>
<form onreset=""></form>
<!— when a user selects a text —>
<input type="text" onselect="">
```

Saad Irfan

@DevWithSaad

JS

Async/Await JavaScript

```
// Used async to make the function act asynchronous
async function getWeatherData() {
  try {

    // Used await to make the code wait until promise returns a result
    const res = await fetch('https://jsonplaceholder.typicode.com/posts')
    const data = await res.json()

    return data
  } catch (err) {
    console.log(err)
  }
}
```

Saad Irfan
@DevWithSaad

JS

Error handling in JS

index.js

```
/* error handling in JavaScript */
function foo() {
    // try catch block
    try {
        //  ...
    } catch (e) {
        // catch error
        //  ...
    }
}

// executes when when a JavaScript Promise that
// has no rejection handler is rejected
window.addEventListener('unhandledrejection', function () {});
```

Saad Irfan
@DevWithSaad

Saad Irfan

@DevWithSaad

Did You Find it Useful?

Share Your thoughts.