



# Javascript Beginner Tutorial Cheatsheet

By: [supersimple.dev](https://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;  
let name = 'Simon';  
let age = 27;
```

Variable = a container that can store a value  
Creates a variable named `name` that stores the string `'Simon'`  
Creates a variable named `age` that stores the number `27`

```
console.log(name);  
console.log('name');
```

Access the value inside a variable by typing the variable name  
Will `console.log('Simon')`; since `name` is a variable  
Will `console.log('name')`; since `'name'` is just a string

```
typeof 'hello'  
typeof 27  
typeof name
```

Returns the type of the value: `'string'`  
`'number'`  
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 everything 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'

The diagram shows the syntax of a function definition: `function addTodo(todoTitle, dueDate) {`. Above `addTodo` is the label **function name** with an upward arrow. Above `(todoTitle, dueDate)` is the label **parameter(s)** with an upward arrow. Inside the function, there is a comment `// ...`. Below the opening curly brace `{` is the label **function body** with a downward arrow pointing to the comment.

```
function addTodo(todoTitle, dueDate) {  
  // ...  
}
```

## 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 = function attached to a value



**method**

```
'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

<code>&lt;div id="my-div"&gt;&lt;/div&gt;</code>	
<code>&lt;script&gt;</code>	
<code>let div = document.getElementById('my-div');</code>	
<code>let button = document.createElement('button');</code>	
<code>div.appendChild(button);</code>	Add element inside <code>&lt;div id="my-div"&gt;&lt;/div&gt;</code>
<code>div.innerHTML = 'hello';</code>	Erases everything inside <code>&lt;div id="my-div"&gt;&lt;/div&gt;</code> and replaces it with what's inside the string ('hello')
<code>div.innerHTML = '';</code>	To erase everything in the div, just use empty string ''
<code>&lt;/script&gt;</code>	

---

<code>let num = 5;</code>	Creates a variable that can be reassigned
<code>num = 6;</code>	
<code>let name;</code>	Create a variable without assigning it a value
<code>const num = 5;</code>	Creates a variable that can <u>not</u> be reassigned (constant)
<code>var num = 5;</code>	Old way of defining a variable (generally not used anymore)

<pre>let person = {   name: 'Taylor',   age: 27 };</pre> <div style="display: flex; align-items: center; margin-top: 10px;"> <div style="text-align: center; margin-right: 20px;">  <p><b>property (key)</b></p> </div> <div> <p><b>value</b></p> </div> </div>	Object = a value in JavaScript that groups related values together
---	--

<code>person.name</code>	Get the value of the name property: 'Taylor'
<code>person.age</code>	27
<code>person['name']</code>	Same as <code>person.name</code> write it like this if the property contains hyphens: <code>person['first-name']</code>
<code>const property = 'name';</code>	
<code>person[property];</code>	Same as <code>person['name']</code>

---

<code>&lt;button style="background-color: red; font-size: 18px"&gt;Click&lt;/div&gt;</code>	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

---

<pre>const personString = JSON.stringify({   name: 'Taylor' }); const personObject = JSON.parse(personString);</pre>	<p>Converts an object into a string <code>'{"name": "Taylor"}'</code></p> <p>Converts a stringified object back into an object</p>
<pre>localStorage.setItem('key', 'data'); localStorage.getItem('key');</pre>	<p>Saves a string in browser under the key: <code>'key'</code></p> <p>Retrieves the string saved under <code>'key'</code></p>
<pre>Array.isArray(myVar);</pre>	<p>Checks if a variable is an array</p>

---

<pre>const func = function () {   console.log('hi'); };</pre>	<p>Alternative way of defining a function</p>
<pre>const func = () =&gt; {}; const func = name =&gt; {   console.log(name); }; const func = (name, age) =&gt; {   console.log(name);   console.log(age); }; const func = () =&gt; 'hi';</pre>	<p>Arrow functions</p> <p>Brackets are optional if only 1 parameter</p> <p>Brackets are required if 2 or more parameters</p> <p>If all on one line, <code>returns</code> the value after <code>=&gt;</code></p>
<pre>const createCounter = function () {   let count = 0;   return function () {     count = count + 1;     console.log(count);   }; }; const count = createCounter(); count();</pre>	<p>Usually when a function <code>returns</code>, all variables inside the function are deleted.</p> <p>Closure = return a function from a function. The inner function will have <u>permanent</u> access to the variables in the outer function (<code>count</code>).</p> <p>This will return a function. Calling the function will <code>console.log(1)</code>;</p>