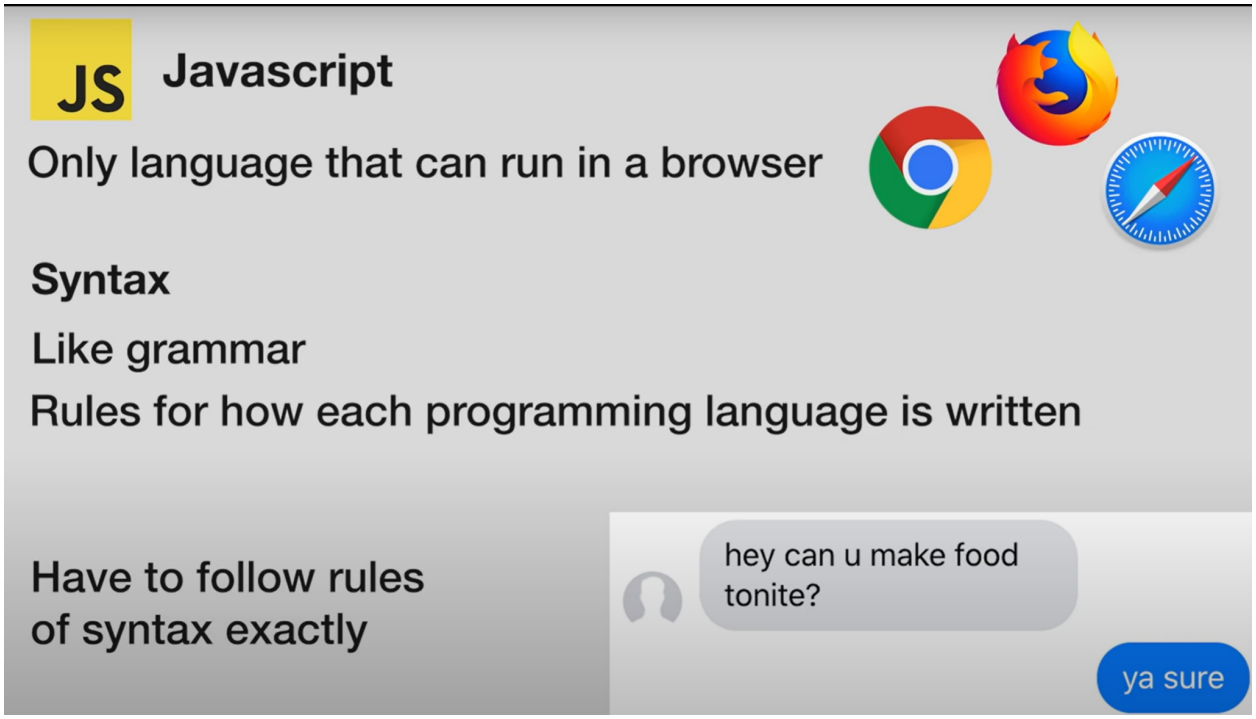
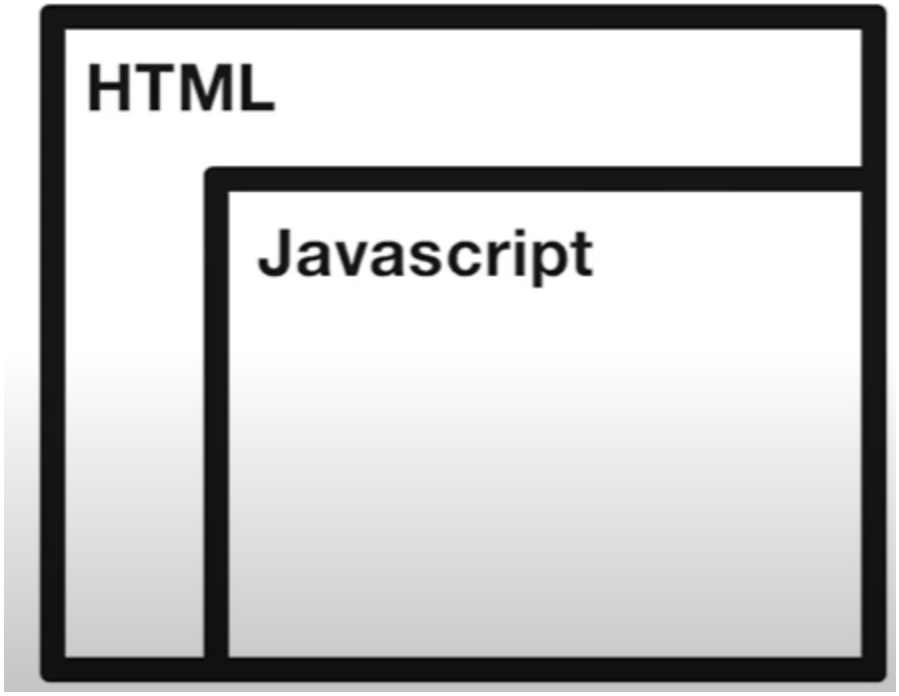
JavaScript



Unlike normal conversations, in programming we have to follow rules exactly otherwise we get errors.

- ; is like fullstop in English.
- In the console it is optional. Won't give errors unless two instructions are written on the same line.

HTML creates visual elements - also called as rendering. JS is used to manage information. They both work together.




```
1    <button>Press me</button>
2    <script>
3      alert("hello");
4    </script>
```

JavaScript has been loaded within HTML.

```
> console.log(todo1)
  Get groceries
< undefined
> console.log('todo1')
  todo1
< undefined
> typeof('todo1')
< 'string'
> typeof 99
< 'number'
> typeof todo1
< 'string'
> console.log('Hello' + 'Simon')
  HelloSimon
< undefined
```

Strings are pieces of text. Can be joined together.
Combining a string and a number results in a string.

HTML Standard Structure

```
<html>
  <head>
    ...Information about the webpage...
  </head>
  <body>
    ...Everything that appears on the webpage...
  </body>
</html>
```

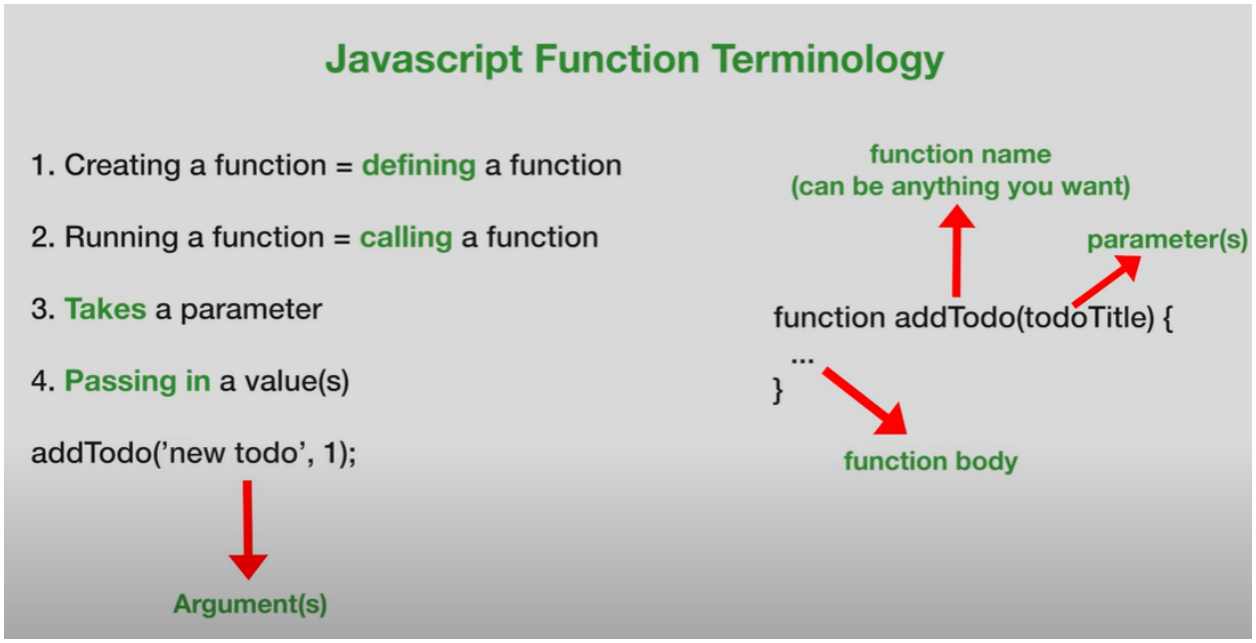
Adding HTML element using JavaScript:

```
// Dynamically add an HTML element
      // Learn new instructions one by one. Most important is to know what
you want to do. Use Google to search any required instructions.
      let element = document.createElement("div");
      element.innerText = todo1;
      document.body.appendChild(element);
```

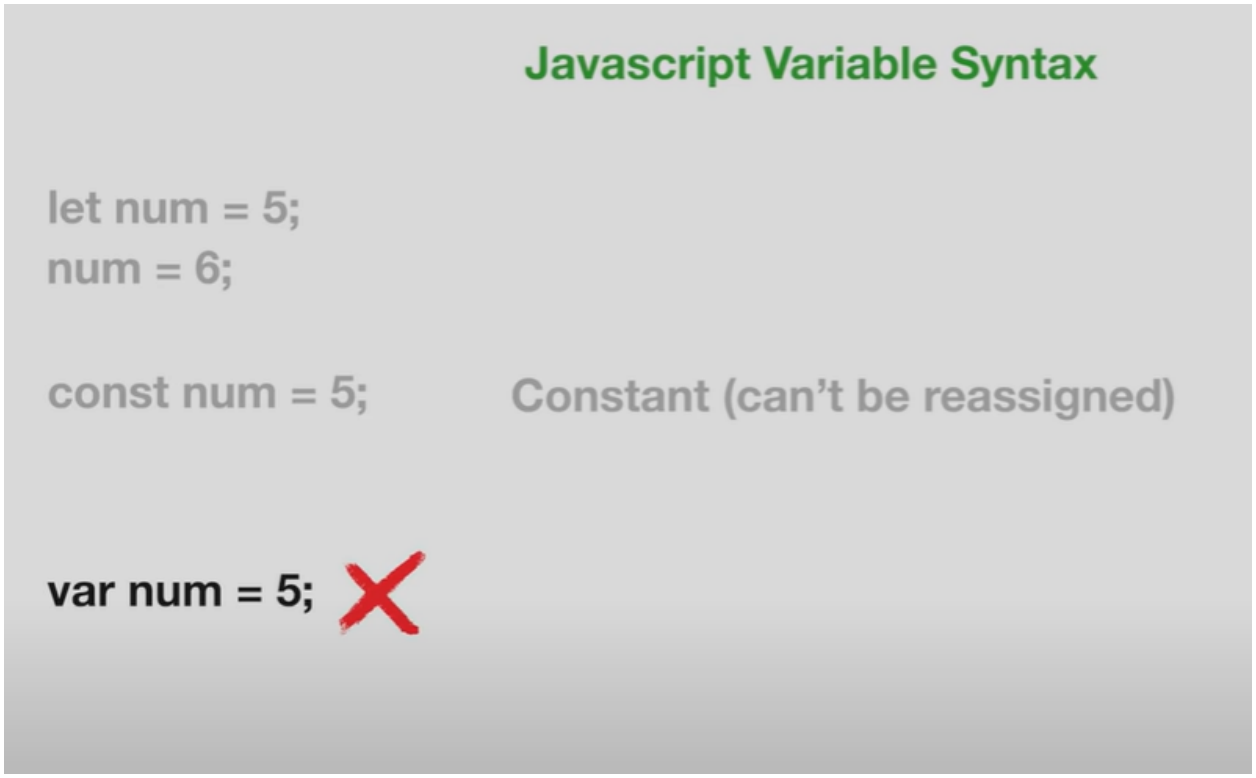
Note:

- When this script is placed in the <script> tag in the head, the new div is not rendered on webpage.
- When <script> tag is added at the end of the body, it gets rendered at the end.
- When <script> tag is added at the beginning of the body, it gets rendered at the beginning.

Functions - reusable pieces of code.



Doubt: What is the sequence followed to execute the script tag?
What is the sequence of execution of the entire HTML file?



let allows variables to be reassigned. var is an older way of assigning variables. Not used now.
const has to be assigned the value right away. let allows to assign values later.
Best Practice: Use const everywhere unless reassigning the variable is required. If reassigning is required, use let.

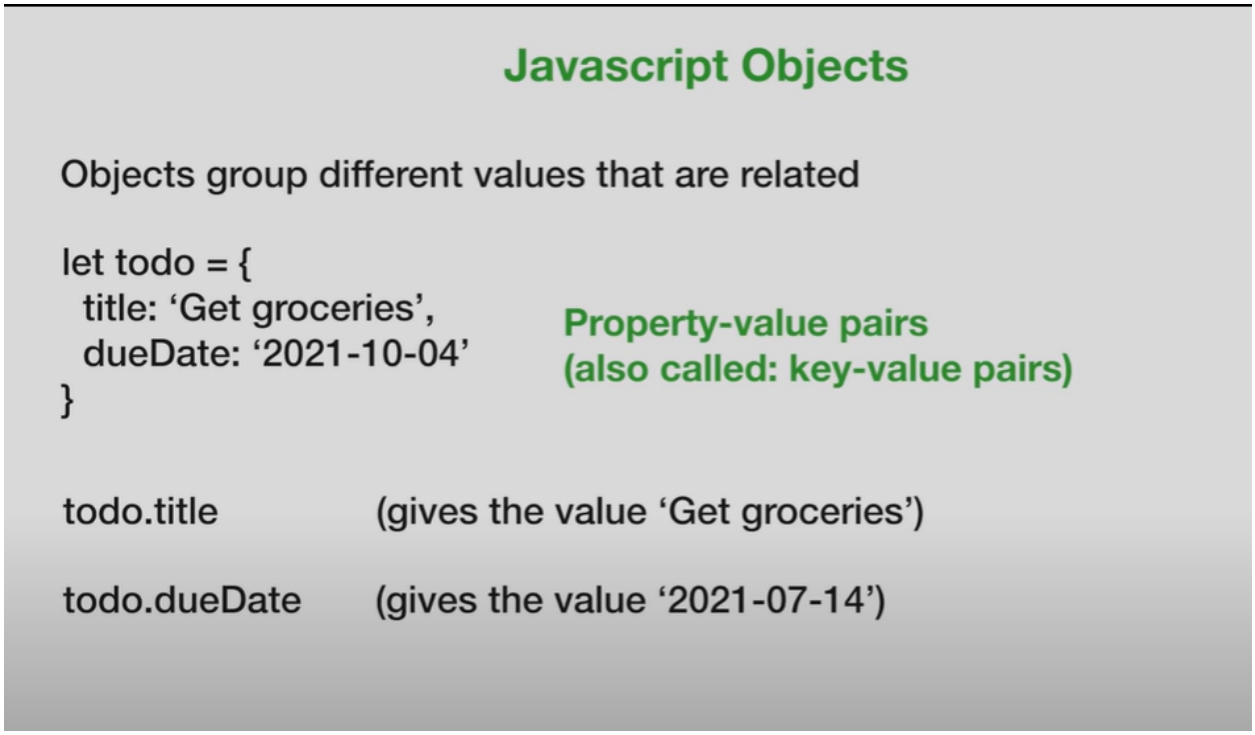
```
> todos
◂ ▶ (9) ['Get groceries', 'Wash car', 'Make dinner', 'another todo', '', 'New Todo', '', '', '']
> todos
◂ ▶ (10) ['Get groceries', 'Wash car', 'Make dinner', 'another todo', '', 'New Todo', '', '', '', 'New Todo 2']
```

Get groceries
Wash car
Make dinner
another todo
New Todo
New Todo 2

Div element in HTML has height: 0 for empty elements.

+'10' converts the string '10' to number.

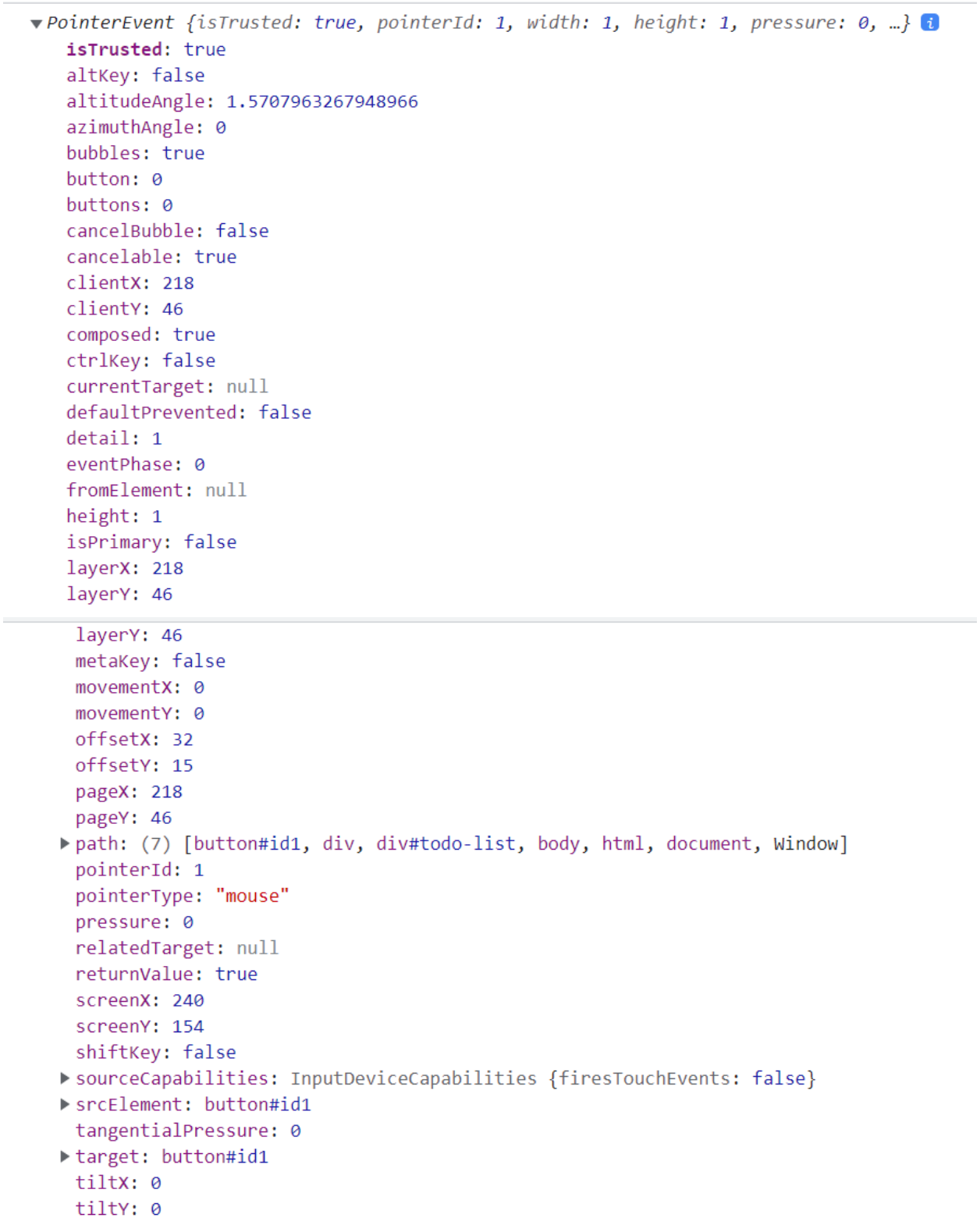
```
> +'10'
< 10
> -'10'
< -10
```



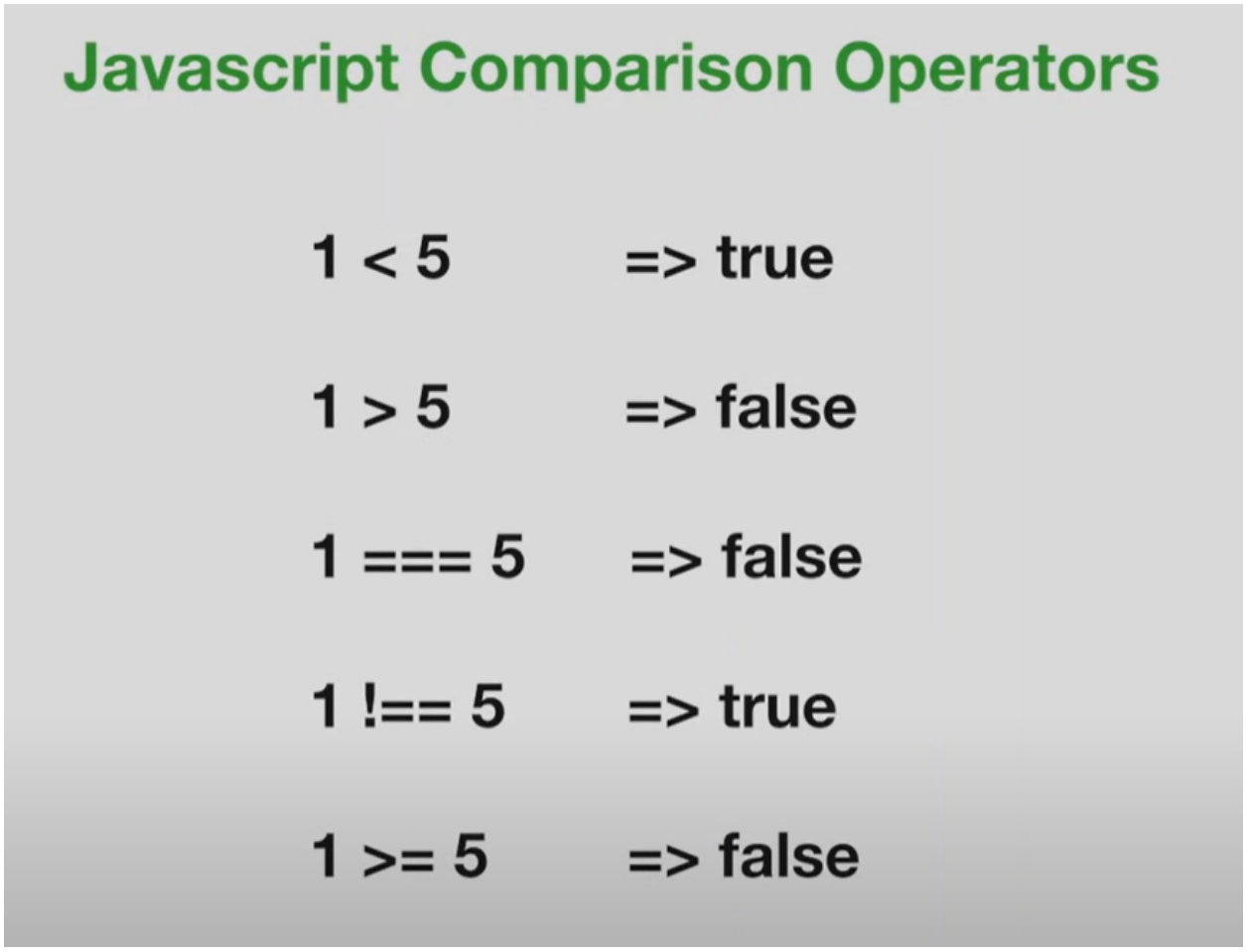
Document Object Model (DOM) basically means that we can grab any HTML element and that gets converted into a JS object so that we can use it seamlessly in JS.

```
// Delete ToDo
function deleteTodo(event) {
  event.target;
}

// Render todos array on the DOM
function render() {
  // reset our list
  document.getElementById("todo-list").innerHTML = "";

  // Append into the todo-list div from the array
  todos.forEach(function (todo) {
    const element = document.createElement("div");
    element.innerText = todo.title + " " + todo.dueDat

    const deleteButton = document.createElement("butto
    deleteButton.innerText = "Delete";
    deleteButton.style = "margin-left: 12px";
    deleteButton.onclick = deleteTodo;
    deleteButton.id = todo.id;
```

Onclick when set using Javascript is a little different from the HTML way. While using Javascript, we do not add ().
Here, the function deleteTodo can take a parameter named event that catches the event that triggered the function.



Target refers to the button that was clicked to trigger the event.

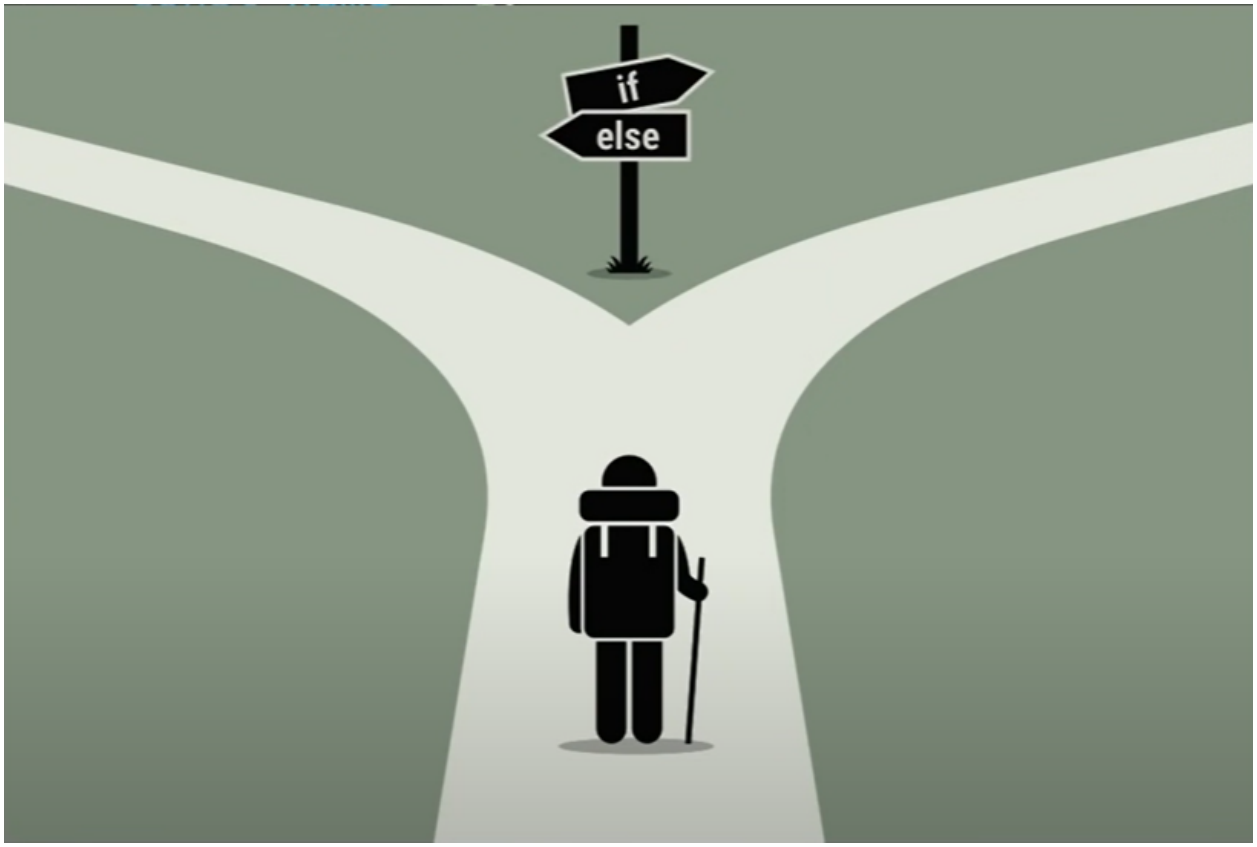

Javascript Comparison Operators
1 < 5 => true
1 > 5 => false
1 === 5 => false
1 !== 5 => true
1 >= 5 => false

Also, 1<= 5 => true

Javascript If-Statements

returns true or false

```
if (code()) {
  thisCode();

} else {
  otherCode();
}
```



TypeError: When we expect an element to be of one type but it turns out to be of another type. TypeScript is getting popular. It helps to deal with type errors.

ChatGPT:

How are JavaScript arrays stored in memory?
JavaScript arrays are stored in memory as objects. An array is an object that has a series of elements, each of which is a property of the object and has a unique key or index.

For example, consider the following array:

```
let arr = [1, 2, 3, 4, 5];
```

In memory, this array would be represented as an object with properties that correspond to the elements of the array. The keys for these properties would be the indices of the elements in the array, starting with 0 for the first element and increasing by 1 for each subsequent element. The values of the properties would be the elements of the array.

So, in this case, the array object would have the following properties:

```
{
  "0": 1,
  "1": 2,
  "2": 3,
  "3": 4,
  "4": 5
}
```

Note that JavaScript arrays are dynamic, which means that you can add or remove elements from the array at any time. When you add an element to the array, the array object will create a new property with the appropriate key and value. When you remove an element from the array, the array object will delete the corresponding property.

What happens when we assign the array variable with a new array?
When you assign a new array to an array variable, the value of the variable is updated to refer to the new array. This does not affect the memory location of the original array, which remains unchanged.

For example:

```
let arr1 = [1, 2, 3];
let arr2 = [4, 5, 6];

console.log(arr1); // Output: [1, 2, 3]
console.log(arr2); // Output: [4, 5, 6]

arr1 = arr2;

console.log(arr1); // Output: [4, 5, 6]
console.log(arr2); // Output: [4, 5, 6]
```

In this example, arr1 and arr2 are both variables that refer to separate arrays in memory. When we assign arr2 to arr1, the value of arr1 is updated to refer to the same array as arr2. However, the original array that arr1 was referring to is not affected and remains in memory.

What happens when I update an argument array in a function in JavaScript?

In JavaScript, when you pass an array as an argument to a function and modify the array inside the function, the changes will be reflected in the original array outside the function. This is because arrays in JavaScript are passed by reference, rather than by value.

For example:

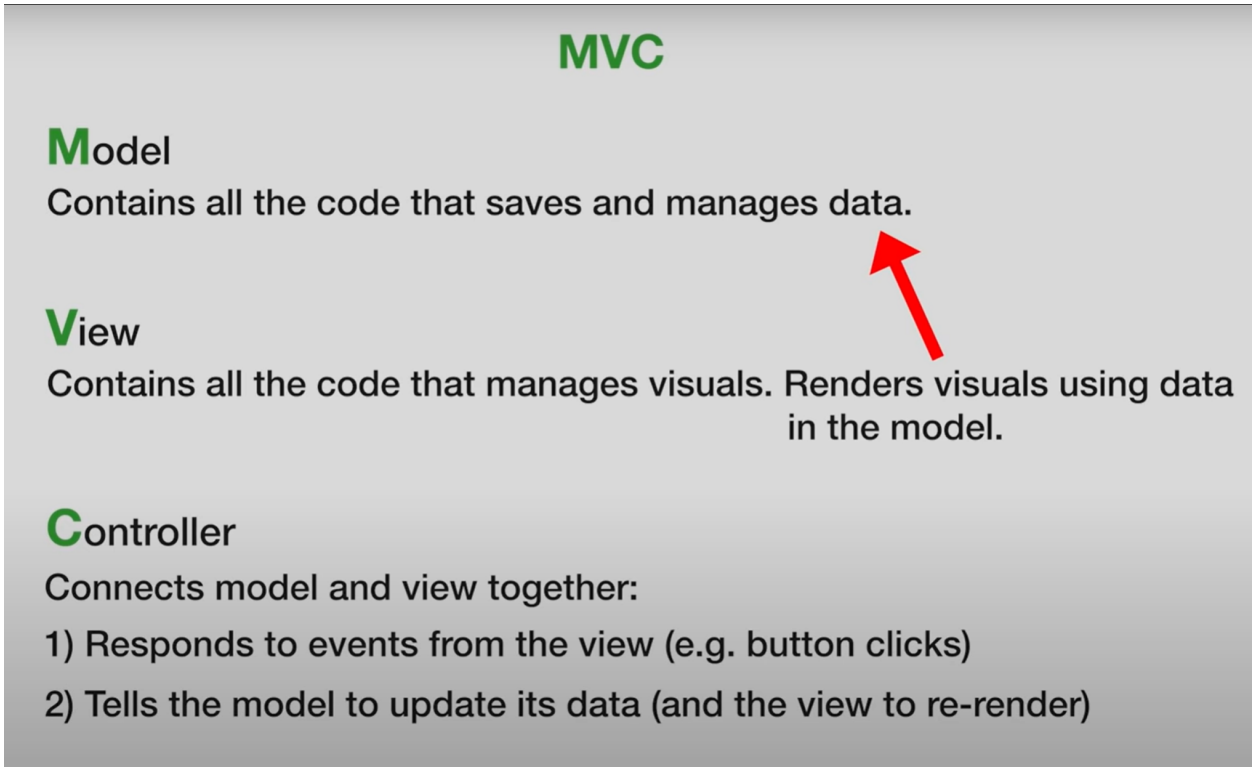
```
let arr = [1, 2, 3];

function modifyArray(arr) {
  arr.push(4);
}

console.log(arr); // Output: [1, 2, 3]

modifyArray(arr);

console.log(arr); // Output: [1, 2, 3, 4]
```

In this example, the modifyArray function takes an array as an argument and adds a new element to the array using the push method. When we call the function and pass in the arr array, the arr array is modified inside the function, and the changes are reflected in the original array outside the function.

It's important to note that this behavior only applies to arrays and objects, which are passed by reference in JavaScript. Primitive data types such as numbers, strings, and booleans are passed by value, which means that modifying them inside a function will not affect the original values outside the function.

MVC

- Way to structure our data.



Why is it important?

- Each section can handle a very specific task.

Use functions (minimum amount of code) to connect these sections and keep scopes separate.
Best practice in programming:
We want our functions to be doing one specific thing only.
Ex. addTodo function in the controller (refer video code - todo.html) deals only with the button click of ‘Add Todo’ button and nothing else. All the details about creating Todo and rendering are handled in different functions. Since they are not part of the controller, we keep them in their respective sections.

How does MVC help you in finding a job?

1. You can build bigger and more impressive projects.
   (For your portfolio that you share with the company)

2. Most Javascript frameworks are built on MVC.

What is a Javascript framework? → provides common functions, methods, and objects for rendering visuals and managing data

```
function render() {
  // 1. Blow away (erase) the list
  // 2. Re-render the list

  Another developer:
  // 1. Keep the list the same
  // 2. Look at what changed and only re-render what's necessary
}
```

How does MVC help you in finding a job?

1. You can build bigger and more impressive projects.
   (For your portfolio that you share with the company)

2. Most Javascript frameworks are built on MVC.

What is a Javascript framework? → provides common functions, methods, and objects for rendering visuals and managing data

```
function render() {

  // The framework can provide
  // this render function.

}
```

→ makes code more consistent

How does MVC help you in finding a job?

1. You can build bigger and more impressive projects.
   (For your portfolio that you share with the company)

2. Most Javascript frameworks are built on MVC.

   Helps you learn frameworks more easily.

   Helps you get a job because you're using the same tools
   that professionals use on the job.

V What is the dataset property of HTML elements?

The `dataset` property of HTML elements is an object that allows you to store custom data private to the page or application on HTML elements. It is part of the HTML5 dataset API, which provides a way to store data on HTML elements in a structured way.

The `dataset` property allows you to set and retrieve data on an HTML element using the `data-*` attributes. The `data-*` attributes are used to store data on an HTML element, where `*` is a name that you can choose. These attributes are used to store data that is private to the page or application and is not intended to be displayed to the user.

For example:

Copy code

```
<div id="myDiv" data-user-id="123" data-user-name="John">
  Hello, my name is John.
</div>
```

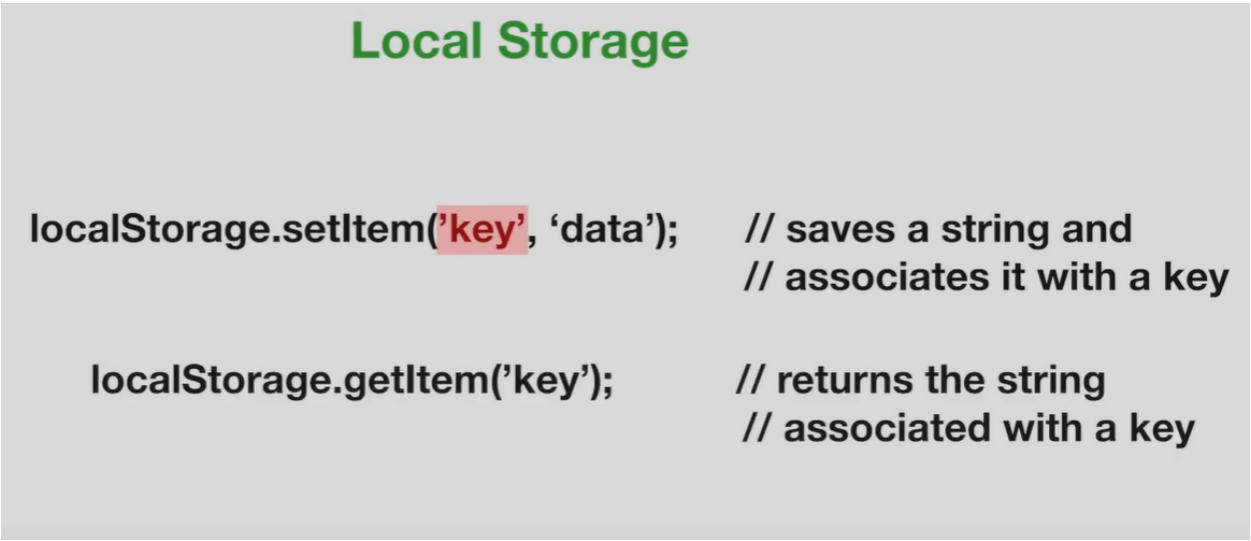
In this example, the `div` element has two `data-*` attributes: `data-user-id` and `data-user-name`. You can access these attributes using the `dataset` property of the `div` element like this:

```
let div = document.getElementById('myDiv');
console.log(div.dataset.userId); // Output: "123"
console.log(div.dataset.userName); // Output: "John"
```

You can also use the `dataset` property to set the values of the `data-*` attributes:

```
div.dataset.userId = 456;
div.dataset.userName = "Jane";

console.log(div.dataset.userId); // Output: "456"
console.log(div.dataset.userName); // Output: "Jane"
```

Save and retrieve data



Local Storage can only store strings.

V How long does a localStorage value last?

In JavaScript, the `localStorage` object provides a way to store data in the browser on a per-origin basis. This means that the data stored using `localStorage` is specific to the domain of the web page, and is available to any page on that domain.

The data stored in `localStorage` persists until it is explicitly removed. This means that the data remains available even if the user closes the browser or shuts down the computer. The data is also available to any page on the same domain that the user visits in the future, until it is removed.

You can remove a value from `localStorage` using the `removeItem` method:

Copy code

```
localStorage.removeItem('key');
```

You can also clear all values from `localStorage` using the `clear` method:

Copy code

```
localStorage.clear();
```

It's important to note that the `localStorage` object has a limited size, typically around 5 MB. If you try to store more data than the available space, an exception will be thrown.

Defining functions differently

Old way of defining function:

```
function addTodo() {
  const textbox = document.getElementById('todo-title');
  const title = textbox.value;

  const datePicker = document.getElementById('date-picker');
  const dueDate = datePicker.value;

  createTodo(title, dueDate);
  render();
}
```

Defining as a value assigned to a variable. The variable name is the function name. It can be used to call the function.

```
const addTodo2 = function () {
  return 'string';
};
```

Arrow functions: Easy way of defining a function.

```
const addTodo2 = (todo, title) => {
  return 'string';
};
```

When only one parameter is present, () are optional and return is hidden.

```
const result = words.filter(word => word.length > 6);
```

JavaScript.info

Fetch: Cross-Origin Requests

- Both client and server should consent for requests.
- To ensure this, CORS is implemented by browsers.
- There are safe cross-origin requests that include GET, POST and HEAD methods and some limited headers. The rest are unsafe.
- For both types, the response headers should include ‘Access-Control-Allow-Headers’: origin-url or *. If not, then it means that the recipient is not willing to accept the request, so an error is generated.
- For unsafe requests, a preliminary preflight request is sent. (Only for unsafe requests?)
- A cross-origin request initiated by JavaScript code by default does not bring any credentials (cookies or HTTP authentication). By adding credentials: “include”; we allow credentials to be included.

Document and resource loading