TODO App
DOM Manipulation, Event Handlers

Relevel
by Unacademy

List of Concepts Involved

In this session, we are going to learn about basic concepts of DOM using TODO app and javascript. We will see detailed examples of below concepts.

- DOM Manipulation
- Event Listener

CodeSandbox & Vanilla JS Template

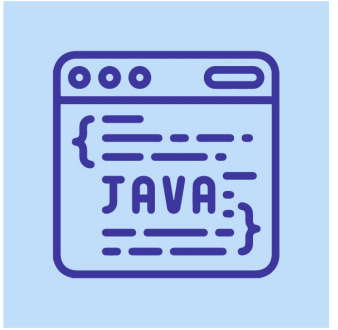
CodeSandbox is online IDE which can be used to create JavaScript based web applications. We can also share our projects easily using it. It's easier to code on this IDE since we don't need to do setup and infrastructure configuration

Introduction and setting up the Coding Environment

- Introduce to students the coding environment that we will be using - codesandbox.io
- Click on Create Sandbox in the top right corner.
- Select the Vanilla JS template.

How does JavaScript work?

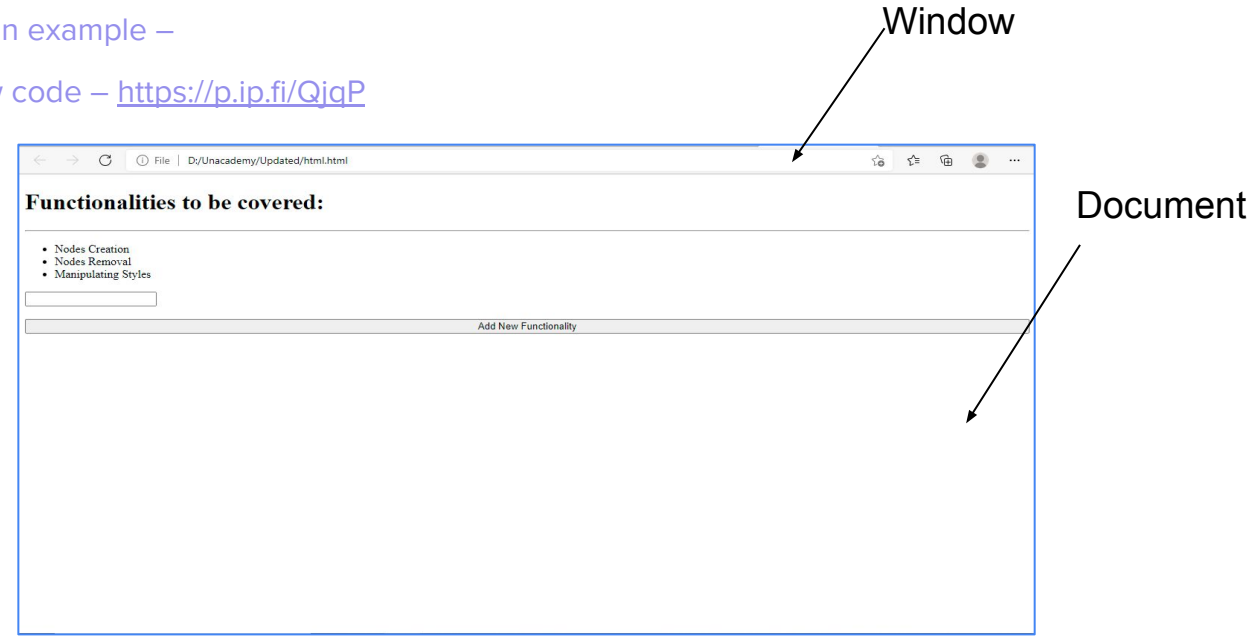
When you open any website on the Internet, your computer receives a lot of files, including at least one HTML file. This HTML file may contain a <script> tag. The <script> tag is used to embed or reference executable code; this is typically used to embed or refer to JavaScript code. In the current case, the JavaScript is the client-side language, which means it waits to execute code until it's present on your computer, using a front-end program called the web browser. Each browser has its own rendering and JavaScript engines, which actually execute the code.

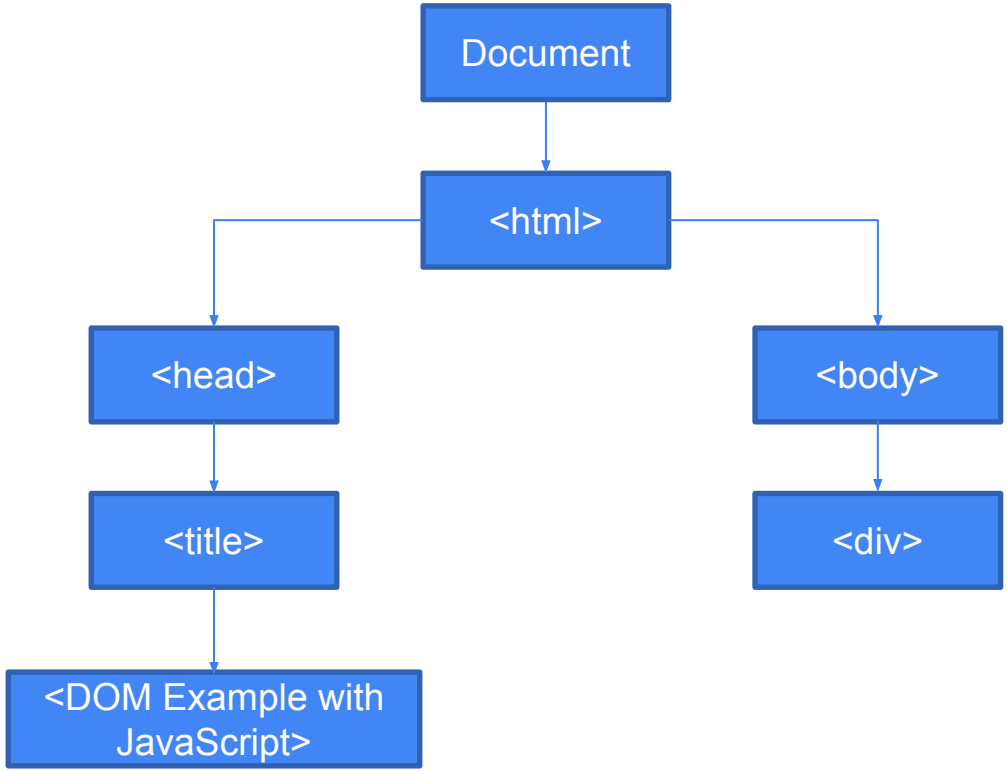


DOM (Document Object Model)

DOM is tree-like structure which represents hierarchical relationship between HTML elements. Let's take an example –

Have a look on below code – https://p.ip.fi/QjqP




Document
<html>
<head>
<body>
<title>

<DOM Example with JavaScript>

Commonly used terminologies in DOM

Element node: An element, as it exists in the DOM.

Root node: The top node in the tree, which in the case of HTML is always the HTML node (other markup vocabularies like SVG and custom XML will have different root elements).
In our example, <html> is the root node.

Child node: A node directly inside another node.<title> is the child node of <head> node

Descendant node: A node anywhere inside another node. node is a descendant of the <html> node.

Parent node: A node which has another node inside it. <head> node is parent node of <title> node.

Sibling nodes: Nodes that sit on the same level in the DOM tree. Inside the node, we have <h1> node and <li> node. These both nodes are at the same level and hence they are sibling nodes.

Text node: A node containing a text string. <input> node is text node.

DOM Manipulations

Selecting Node

To select node, we have to use querySelector function.

Example –

let functionality = document.querySelector("#userInput").value;

As argument, we are giving textbox and when user will enter any value in textbox, that value can be fetched in functionality variable.

We can also use below functions to select the node.

- const target = document.getElementById("dateTime").value // It will select all elements with ID attribute as “dateTime”
- const target = document.getElementByTagName("li") // It will select all elements with node

DOM Manipulations

Creating Element

To create element, we have to use createElement function.

Example –

let li = document.createElement("li");

As argument, we are giving element tag and this will create new element in our webpage.

DOM Manipulations

Creating Node

To create node, we have to use createTextNode.

Example –

let node = document.createTextNode(functionalityInput);

Here we are giving node tag to create new node.

DOM Manipulations

Adding Node

To add node, we have to use appendChild function.

Example –

```
let node = document.createTextNode(functionalityInput);
li.appendChild(node);
```

As argument, we are giving node tag and this will add newly created node to existing node.

DOM Manipulations

Removing Node

To remove node, we have to use removeChild function.

Example –

```
let node = document.createTextNode(functionalityInput);
li.appendChild(node);
li.removeChild(node);
```

As argument, we are giving node tag and this will remove newly added node again from existing node.

DOM Manipulations

Removing Element

To remove elements, we have to use remove function.
Example –

```
var elementObj = document.getElementById("dateTime");
elementObj.remove();
```

As an argument, we are giving ID attribute value and hence this will remove elements with id value as “dateTime”

Replacing Node

To replace nodes, we have to use the replaceChild() function.
Example –

```
let newNode = document.createTextNode(functionalityInput); // Create new text node
var itemNode = document.getElementById("dateTime"); // Get itemNode
itemNode.replaceChild(newNode , itemNode); // Replace itemNode with newNode
```

As an argument, we are giving 2 nodes, first is a new node and second is an existing node which we want to replace.

DOM Manipulations

Access ParentNode

To get the parent node of a node,we can use the parentNode.
Example – *var parentNodeName = document.getElementById("li").parentNode.nodeName;*

Window Element

Apart from functions we have seen to manipulate DOM elements and nodes, we also have a Window interface which represents a window having DOM. We can get different attributes of a window using the Window interface of javascript. Let's have a look on few examples -

1) *var width = window.innerWidth; // Fetch Width of Window*
2) *var height = window.innerHeight; // Fetch Height of window*

Change HTML Content

We can change the HTML content of elements.. Let's have a look on below example -
document.getElementById("dateTime").innerHTML = "dateTime changed!";
In the above example, the content of the elements having tag "dateTime" changed.

DOM Manipulations

Manipulating Styles

To set attribute of CSS properties or style, we can use setAttribute function.

It will take 2 arguments –

1) Attribute

2) Value

Example –

```
li.setAttribute('class', 'highlight');
```

As argument, we are giving ‘class’ attribute of li node to be set as highlight

DOM Event Listener

Sometimes, we need to trigger an event when there is some action. Event is output of action. Event can be navigation, addition of new data and so on. EventListener handle any event. It will execute some logic which we have defined in eventListener whenever event will trigger.

We will see below examples of eventListener –

1) When button got clicked by user
   When user will click on any button, we want to execute some logic. That can be done using eventListener. Let's take an example

```
document.getElementById("submitButton").addEventListener("click", submit);
function submit() {
  console.log ("SUBMIT Button clicked!");
}
```

   In above example, when user click on submitButton, it will call submit function. We can write logic in our submit function and that will be executed whenever user clicks on submit button.

DOM Event Listener

When user enter some key

When user enter some key, we want to execute some logic. That can also be done using eventListener. Let’s take an example

```
window.addEventListener('keypress', function (e) {
    if (e.keyCode !== 13) {
        console.log(“Key Pressed”);
    }
}, false);
```

In above example, when user press any key except “Enter” key, it will print “Key Pressed” on console. Code for Enter Key is 13 and hence based on our logic, it will print output only when key pressed is different from “Enter” key.

TODO App Implementation Code

Example Code

Practice Code

- Create Sample Tree structure of given TODO App
- Create a new TODO App which will have an input field to add tasks. We will have 2 different lists - “To be Done” and “Completed” List
    - When user enter task in the input field, it should be added to “To Be Done” list
    - When a user clicks on the “Mark as Done” button on a task present in the “To Be Done” list, it should get added to the “Completed” list and removed from the “To Be Done” list.

Thank you!