upGrad

#RahoAmbitious

Full Stack Software Development

Course: JavaScript and Server-Side Communication

Lecture On: DOM Manipulation and Web Storage

Instructor: Siddhesh Prabhugaonkar

upGrad

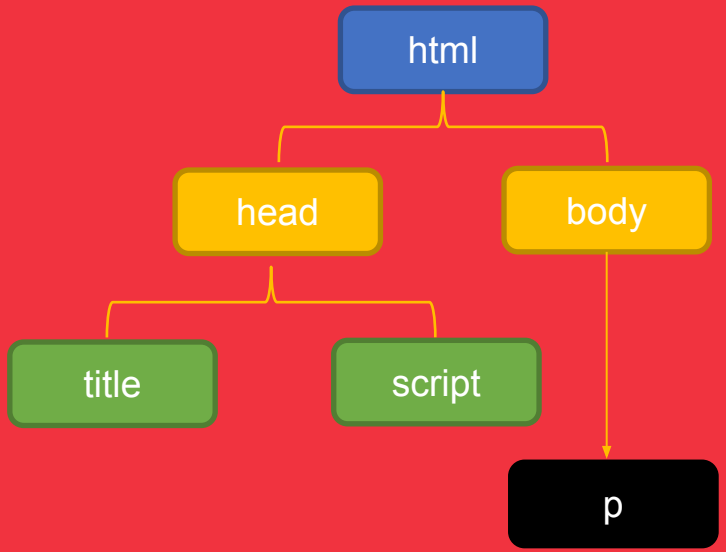
In the previous session, we covered….

- bind(), call() and apply() methods
- Introduction to DOM

Today's Agenda

- DOM manipulation using JavaScript
- Web storage

DOM Manipulation



Creating New Element

To create an element, you use the **document.createElement()** method. Generally, the created element is stored in a variable.

```
<script>
      let para = document.createElement("p");    // Create a <p> element
      para.innerText = "Welcome to upGrad";      // Insert text inside <p> tag
      document.body.appendChild(para);            // Append para to <body>
</script>
```

DOM Manipulation

upGrad

Adding New Element to Parent

Once an element is created, it needs to be appended to either the document or another element. This is achieved using the **appendChild()** method. This method attaches the element to the last node (at the end) of the specified parent node.

```
<!DOCTYPE html>
<head>
    <title>DOM</title>
</head>
<body>
    <script>
      let para = document.createElement("p");    // Create a <p> element
      para.innerText = "Welcome to upGrad";      // Insert text inside <p> tag
      document.body.appendChild(para);            // Append para to <body>
    </script>
</body>
</html>
```

Removing Existing Element

To remove a child element, you should use the **removeChild()** method. You can get an element using ***getElementById***, ***getElementsByClassName*** and ***getElementsByTagName methods***, and then, you can use the removeChild() function to remove the child element.

```
<body>
    <div id="container">
      <p id="para">Hello</p>
    </div>
    <script>
      let container =
document.getElementById("container");
      let para = document.getElementById("para");
      container.removeChild(para);
    </script>
</body>
```

Poll 1 (15 Sec)

What will be the output of the following code?

```
let newNode = document.createElement('h1');
```

1. Creates a new paragraph element named h1
2. Creates a new h1 element for heading
3. Creates a new variable named h1 in JavaScript

Poll 1 (Answer)

What will be the output of the following code?

```
let newNode = document.createElement('h1');
```

1. Creates a new paragraph element named h1
2. **Creates a new h1 element for heading**
3. Creates a new variable named h1 in JavaScript

Poll 2 (15 Sec)

What is the best suited option representing the output of the code given in the following screenshot?

```
<body>
    <div id="mydiv"></div>
    <script>
        let newTextElement = document.createElement('p');
        newTextElement.innerHTML = "<p>New Paragraph</p>";
        document.getElementById('mydiv').appendChild(newText);
    </script>
</body>
```

1. A paragraph is created inside the , and the text “New Paragraph” is appended to the paragraph.
2. A paragraph is created, and the text “New Paragraph” is appended to the paragraph.
3. Code will throw an error.

Poll 2 (Answer)

What is the best suited option representing the output of the code given in the following screenshot?

```
<body>
    <div id="mydiv"></div>
    <script>
        let newTextElement = document.createElement('p');
        newTextElement.innerHTML = "<p>New Paragraph</p>";
        document.getElementById('mydiv').appendChild(newText);
    </script>
</body>
```

1. **A paragraph is created inside the , and the text “New Paragraph” is appended to the paragraph.**
2. A paragraph is created, and the text “New Paragraph” is appended to the paragraph.
3. Code will throw an error.

DOM Manipulation

upGrad

Changing Styles Using JavaScript

- To change the style of an element, you use the **style** property after you fetch that element.
- The syntax to change the style of an element using CSS is as follows:

```
document.getElementById(id).style.property = new value; // syntax
```

You can change the color of a <p> tag with the id “para” to red using the following syntax:

```
document.getElementById(“para”).style.color = “red”; //this will make the para appear in red color
```

You can change the display property of a with the id “container” using the following syntax:

```
document.getElementById(“container”).style.display = “none”; // this will make the div invisible
document.getElementById(“container”).style.display = “block”; // this will make the div visible
```

When the css property is more than one word long and has a hyphen in between. For example, background-color in CSS would be written as backgroundColor in JS.

```
document.getElementById(“para”).style.backgroundColor = “red”; //this will make the background color red color
```

Poll 3 (15 Sec)

What will be the output of the following code?

```
<p id="para" style="font-size: 2px;">Welcome to upGrad</p>
<script>
   document.getElementById('para').style.fontSize = '50px';
</script>
```

1. The font-size of ‘para’ will remain 2px.
2. The font-size of ‘para’ will increase to 50px.

Poll 3 (Answer)

What will be the output of the following code?

```
<p id="para" style="font-size: 2px;">Welcome to upGrad</p>
<script>
   document.getElementById('para').style.fontSize = '50px';
</script>
```

1. The font-size of ‘para’ will remain 2px.
2. **The font-size of ‘para’ will increase to 50px.**

Poll 4 (15 Sec)

What will the output of code given in the adjoining screenshot look like?

```
<p id="para" style="font-size:20px;background-color: purple">Welcome to upGrad</p>
<script>
   document.getElementById('para').style.backgroundColor = 'pink';
</script>
```

1. Welcome to upGrad
2. Welcome to upGrad
3. Welcome to upGrad

Poll 4 (Answer)

What will the output of code given in the adjoining screenshot look like?

```
<p id="para" style="font-size:20px;background-color: purple">Welcome to upGrad</p>
<script>
   document.getElementById('para').style.backgroundColor = 'pink';
</script>
```

1. Welcome to upGrad
2. Welcome to upGrad
3. Welcome to upGrad

Events

JavaScript Events

upGrad

Events Using JavaScript (JS)

- Events are actions that happen to HTML elements. For instance, a user 'clicking' on a button is essentially an **event** that is happening to the button element.

- JavaScript (JS) can react to these events. JS helps you execute your code when these events are detected. These codes are called **event handlers**. The syntax of an event looks as follows:

```
<element event="JavaScript code">
```

For instance, if you are targeting the click event that is happening on a button, then you will write the aforementioned code as given below. If the code is small enough, then it can be included an attribute value; otherwise, you can keep it in a function, and call that function by triggering the **onclick** event.

```
<button onclick="alert('Hi')">Click Me!</button>
```

```
<button onclick="alertText()">Click Me!</button>
```

index.js

```
let alertText = () => {}
```

Common Event Functions

Some common event functions are as follows:

Event	Description
onclick	Occurs when a user clicks on an HTML element
onchange	Occurs when an HTML element, for instance, an input box, changes
onmouseover	Occurs when a user moves the mouse over an element
onmouseout	Occurs when a user moves the mouse away from an HTML element
onblur	Occurs when an element loses focus or when element is no longer selected
onkeyup	Occurs when a user presses a key on a keyboard but has not released it
onkeydown	Occurs when a user releases a key already pressed on the keyboard

Poll 5 (15 Sec)

What will be the output of the following code?

```
<body>
  <button id="submit" onmouseover="alert('Welcome to upGrad')">Click Me</button>
</body>
```

1. An alert box will appear when the button is clicked on.
2. An alert box will appear when the cursor is moved over the button.
3. An alert box will appear when the cursor is taken away from the button.
4. None of the above

Poll 5 (Answer)

What will be the output of the following code?

```
<body>
  <button id="submit" onmouseover="alert('Welcome to upGrad')">Click Me</button>
</body>
```

1. An alert box will appear when the button is clicked on.
2. **An alert box will appear when the cursor is moved over the button.**
3. An alert box will appear when the cursor is taken away from the button.
4. None of the above

Event Listener

Event Listeners

- You can attach an event to any element in the HTML DOM. You can attach multiple events to the same DOM element.
- Instead of adding event listeners in the HTML markup along with the all attributes, you can add them separately using the addEventListener() method.
- The syntax of an *addEventListener* method is as follows:

```
element.addEventListener(event, function [, useCapture]);
```

- The first parameter is the type of event, such as click, focus and blur.
- The second parameter is the function that will be called when an event is triggered. You can either declare and create the entire function as the second parameter or, more preferably, call a function that is declared and created inside the event.
- The third parameter is optional. It is a Boolean value to declare whether to use event bubbling or event capturing.

Event Listeners

```
document.getElementById(“container”).addEventListener(“click”, myFunction());

function myFunction() {
      alert(“Hello World”);
}
```

In the example given above, you set the click event on the element that has the id “container”. Whenever the element with id “container” is clicked on, the function named myFunction() will be invoked, and it will print an alert box with the words “Hello World”.

- The third parameter that you saw in the last example is for **event propagation.** Event Propagation is a category which has two parts - **bubbling** & **capturing**.

- If you have nested elements, for instance, a <p> tag inside a , then **event bubbling** specifies that the click event of <p>, which is the inner element, should be handled first, followed by the click event of the , which is the outer element. Thus, in event bubbling, the event is captured by the innermost element first and then it is propagated to the outer element(s).

- In **event propagation**, the click event of the outer element is handled before the click event of <p>. Thus, in event capturing, the event is captured by the outermost element first and then it is propagated to the inner element(s).

Poll 6 (15 Sec)

What will be the output of the code given in the following screenshot?

1. Three paragraphs with the text “ New paragraph ”
2. Two paragraphs with the text “New paragraph”
3. Three paragraphs with the text “New paragraph”
4. Two paragraphs with the text “ New paragraph ”

```
<body>
   <div id="mydiv"></div>
   <script>
       for(let i=1; i<3; i++)
       {
          let newText = document.createElement('p');
          newText.innerHTML = "<p>New Paragraph</p>";
          document.getElementById('mydiv').appendChild(newText);
       }
   </script>
</body>
```

upGrad

Poll 6 (15 Sec)

What will be the output of the code given in the following screenshot?

1. Three paragraphs with the text “ New paragraph ”
2. **Two paragraphs with the text “New paragraph”**
3. Three paragraphs with the text “New paragraph”
4. Two paragraphs with the text “ New paragraph ”

```
<body>
   <div id="mydiv"></div>
   <script>
       for(let i=1; i<3; i++)
       {
          let newText = document.createElement('p');
          newText.innerHTML = "<p>New Paragraph</p>";
          document.getElementById('mydiv').appendChild(newText);
       }
   </script>
</body>
```

preventDefault()

upGrad

- If an event does not get handled explicitly then, **preventDefault()** will cancel that event if the event is cancellable such that the default action associated with that event won't occur.
- preventDefault() method is useful when we do not want the form to be refreshed after clicking on "submit" button.

```
let submitEvent = e => {
    e.preventDefault();
    alert("Form Submitted");
}
```

Templating

Templating

upGrad

- If you are creating a multiple-page website, in which every page has the same header and footer, then it would be strenuous to repeat the same header and footer HTML code on every page.
- Alternatively, you can simply create a variable that stores the element DOM that you want to replicate, and using JavaScript, you can get the parent elements and either append this variable as its children or add it to the innerHTML. This is called creating **templates**.

```
let headerTemplate = `<div class="header">
    <img src="logo.png" />
    <h4>My Website</h4>
    <img src="user-thumbnail.png" />
    </div>`;

document.getElementById("container").innerHTML = headerTemplate;
```

```
<html>
  <body>
     <div id="container"></div>
  </body>
</html>
```

upGrad

Doubt Clearance (5 mins)

upGrad

Project Work

(Let’s add the required DOM manipulations to our project.)

You can refer to the solution here.

Web Storage

Web Storage

upGrad

- Web applications can store data locally within users' browsers with the help of web storage. All pages from one origin can access the same data.
- The two different ways of storing data on the web are as follows: **localStorage** and **sessionStorage**.
- The **localStorage** object stores data with no expiration date. The data will not be deleted when the browser is closed, and it will be available until its deleted.
- The **sessionStorage** object is similar to the localStorage object, except for the fact that 'the data stored in this object will be deleted once the user closes the specific browser tab. It stores data only for one session; hence, it is called the sessionStorage object.

Web Storage

upGrad

```
// Save data to sessionStorage
sessionStorage.setItem('key', 'value');

// Get saved data from sessionStorage
let data = sessionStorage.getItem('key');

// Remove saved data from sessionStorage
sessionStorage.removeItem('key');

// Remove all saved data from sessionStorage
sessionStorage.clear();
```

Web Storage

upGrad

```
<body>
  <div id="container"></div>

  <script>
      // Set
      sessionStorage.setItem("name", "Prachi Agrawal");
      // Retrieve
      document.getElementById("container").innerText = sessionStorage.getItem("name");
  </script>
</body>
```

Web Storage

upGrad

```
// The following snippet accesses the current domain's local Storage object and adds a data item to it using
Storage.setItem().
localStorage.setItem('myCat', 'Tom');

// The syntax for reading the localStorage item is as follows:
var cat = localStorage.getItem('myCat');

// The syntax for removing the localStorage item is as follows:
localStorage.removeItem('myCat');

// The syntax for removing all the localStorage items is as follows:
localStorage.clear();
```

Web Storage

upGrad

```
<body>
  <div id="container"></div>

  <script>
      // Set
      localStorage.setItem("name", "Prachi Agrawal");
      // Retrieve
      document.getElementById("container").innerText = loaclStorage.getItem("name");
  </script>
</body>
```

upGrad

Project Work

(Let's learn how you can use sessionStorage in our project.)

You can refer to the solution here.

Hands-On Exercise (3 mins)

upGrad

Write the JavaScript for the given HTML form such that:

1. When hovered over the SUBMIT button, the cursor is of the type pointer.

1. When the SUBMIT button is clicked:
   a. The background color of the div changes from white to grey.
   b. The paragraph text changes its color to white.
   c. The text inside the <p> tag changes to “Registered”.

Your page should look like the one depicted in the adjoining image.

The stub code is provided here.

The solution is provided here.

Before button click:

After button click:

Key Takeaways

- You can create elements using createElement() method, append elements as children using appendChild() method and remove children element using removeChild() method.
- You can also change styles using JavaScript (JS) by mentioning the property and value using setAttribute() method.
- Events are actions happening to an element. Events include click, blur, focus and mouseover.
- Additionally, you can create event listeners separately by invoking addEventListener() method.
- To avoid repetition of code, you can write pre-defined code called templates.
- Data can be stored in the browsers using localStorage (data that can be stored persistently until its deleted) and sessionStorage (data that will be deleted once the browser tab is closed).

The following tasks are to be completed after today's session:

MCQs
Coding Questions
Project - Checkpoint 2

In the next class, we will discuss...

- Introduction to ES6
- Introduction to let and const
- Arrow functions of ES6
- map(), filter() and reduce() methods
- Spread operator, ternary operator, temp literals, import and export

upGrad

#RahoAmbitious

Thank you!