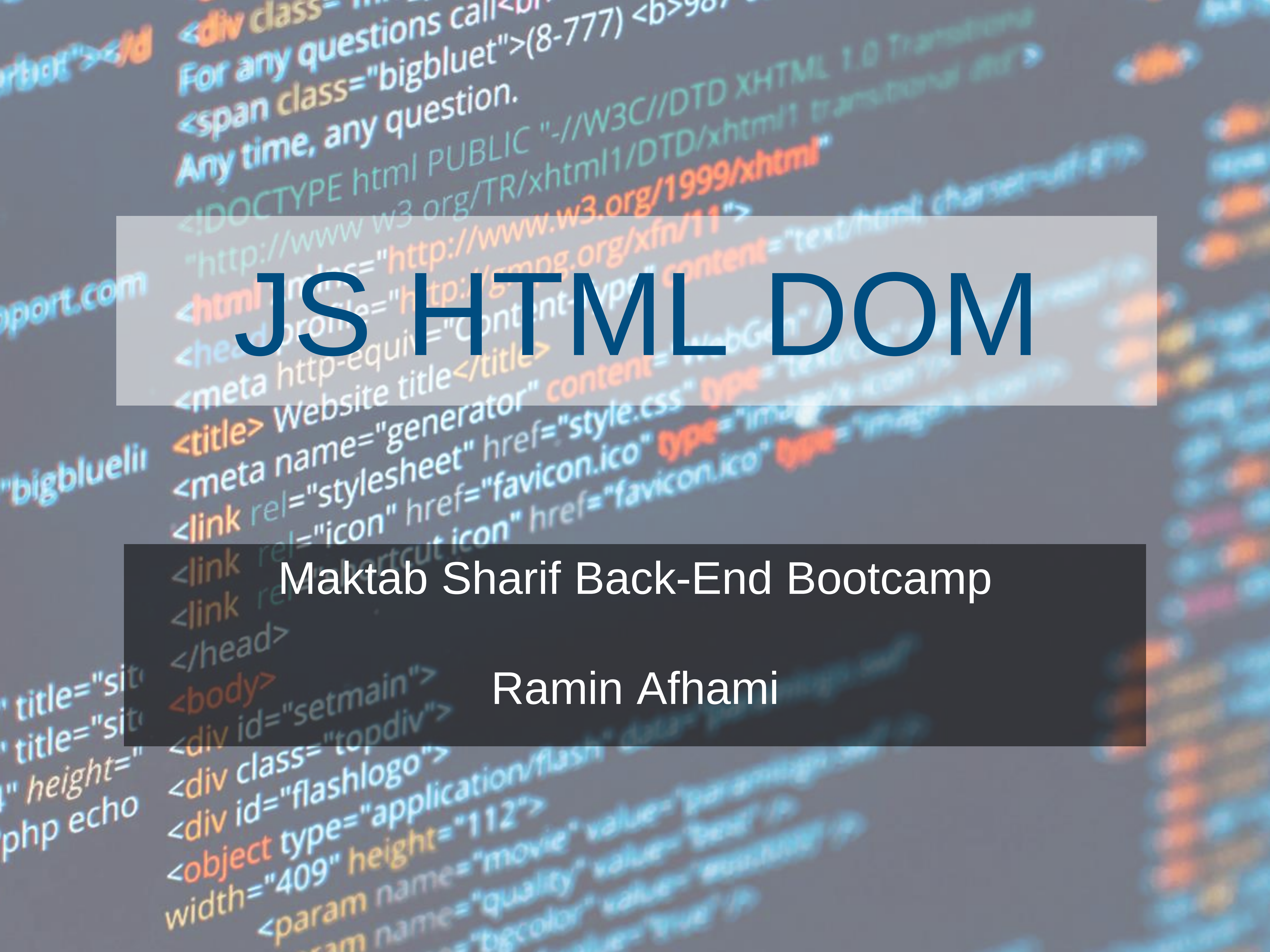


JS HTML DOM

Maktab Sharif Back-End Bootcamp

Ramin Afhami

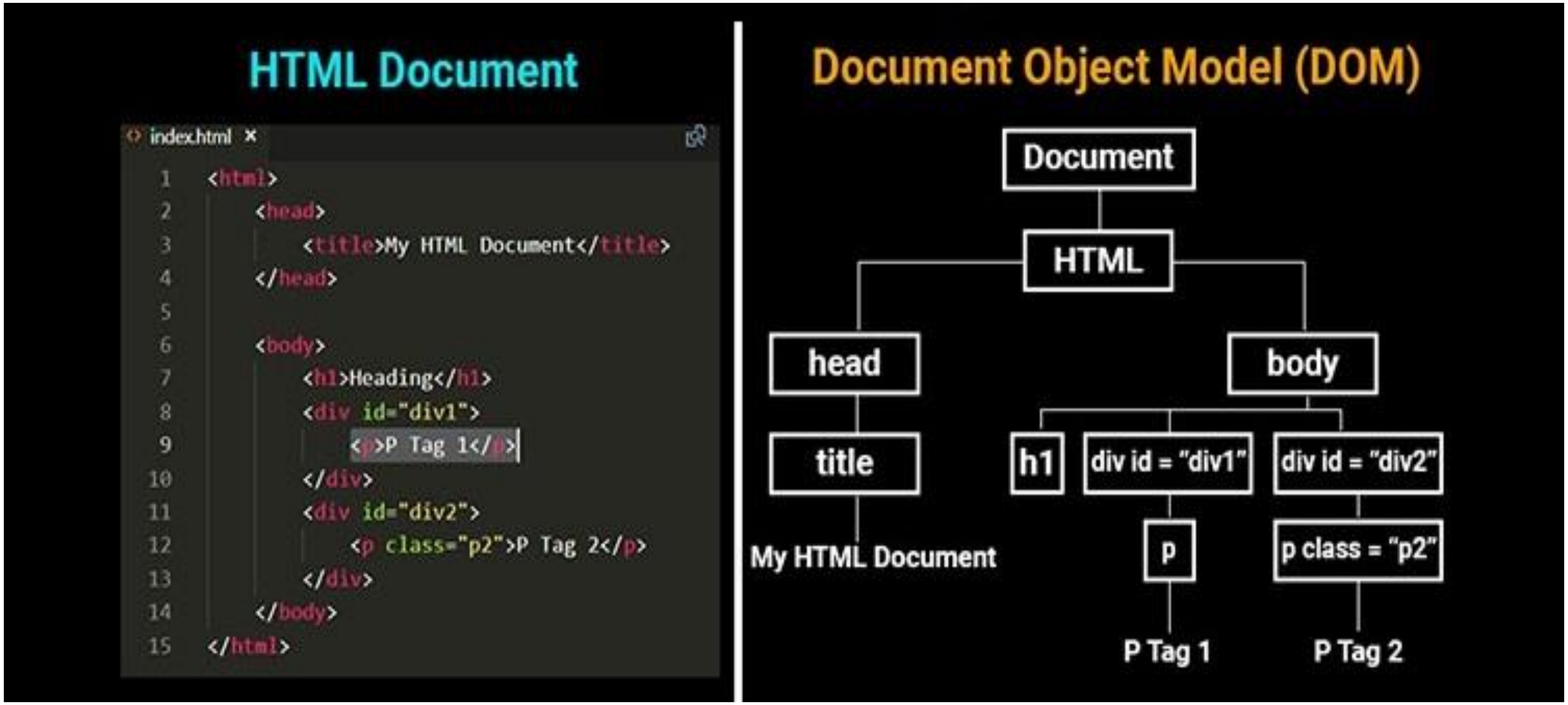
Document Object Model

The Document Object Model (DOM) is a programming interface for HTML documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects. That way, programming languages can connect to the page.

- A Web page is a document.
- The Document Object Model (DOM) represents that same document so it can be manipulated.
- The DOM is an object-oriented representation of the web page, which can be modified with JavaScript.
- You don't have to do anything special to begin using the DOM.

Document Object Model

- A Web page is a document.



```html
<html>
    <head>
        <title>My HTML Document</title>
    </head>

    <body>
        <h1>Heading</h1>
        <div id="div1">
            <p>P Tag 1</p>
        </div>
        <div id="div2">
            <p class="p2">P Tag 2</p>
        </div>
    </body>
</html>
```

Document Object Model

- You don't have to do anything special to begin using the DOM.

.

```
project/
 |-- css/
 |     |-- style.css
 |-- js/
 |     |-- scripts.js
 |-- index.html
```

JavaScript can …

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

The DOM Programming Interface

The HTML DOM can be accessed with JavaScript.
In the DOM, all HTML elements are defined as objects.
The programming interface is the properties and methods of each object.

- A property is a value that you can get or set (like changing the content of an HTML element).
- A method is an action you can do (like add or deleting an HTML element).

```
<body>
<p id="demo"></p>
...
```

Index.html

```
document.getElementById("demo").innerHTML = "Hello World!";
```

Script.js

Finding HTML Elements

Often, with JavaScript, you want to manipulate HTML elements.

1. elements by id
2. elements by tag name
3. elements by class name
4. elements by CSS selectors
5. elements by HTML object collections

Finding HTML Elements

1. elements by id

Index.html
```
<p id="demo"></p>
```

Script.js
```
let myElement = document.getElementById("demo");
```

2. elements by tag name

Index.html
```
<p></p>
```

Script.js
```
let x = document.getElementsByTagName("p");
```

Finding HTML Elements

3. elements by class name

```html
<p class="intro">The DOM is very useful.</p>
```

Index.html

```js
let x = document.getElementsByClassName("intro");
```

Script.js

4. elements by CSS selectors

```html
<p class="intro">The DOM is very useful.</p>
```

Index.html

```js
let x = document.querySelectorAll("p.intro");
```

Script.js

Finding HTML Elements

5. elements by HTML object collections

```html
<a name="html">HTML Tutorial</a>
<a name="css">CSS Tutorial</a>
<a name="xml">XML Tutorial</a>
```

Index.html

```javascript
let x = document.anchors;
```

Script.js

Finding HTML Elements

5. elements by HTML object collections

Method	Collections
`document.anchors`	Anchor tags
`document.body`	Body tag
`document.forms`	Form tags
`document.images`	Image tags
`document.head`	Head tag
`document.title`	Title tag
`document.scripts`	Script tags
`document.links`	Link tags
`document.embeds`	Embed tags

Manipulating DOM Elements

Now that you've learnt how to select elements. we will learn how to add or remove DOM elements dynamically, get their contents, and so on.

1. Adding New Elements to DOM
2. Getting HTML Contents to DOM
3. Setting HTML Contents to DOM
4. Removing Existing Elements from DOM
5. Replacing Existing Elements in DOM

Manipulating DOM Elements

1. Adding New Elements to DOM

Index.html

```html
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>
```

Script.js

```javascript
let newDiv = document.createElement("div");

let newContent = document.createTextNode("Hi!");

newDiv.appendChild(newContent);

let currentDiv = document.getElementById("main");

currentDiv.appendChild(newDiv);
```

Manipulating DOM Elements

2. Getting HTML Contents to DOM

Index.html

```
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>
```

Script.js

```
let contents = document.getElementById("main").innerHTM
alert(contents); // Outputs inner html contents
```

Manipulating DOM Elements

3. Setting HTML Contents to DOM

```html
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>
```

Index.html

```javascript
let mainDiv = document.getElementById("main");
mainDiv.innerHTML = "<p>This is aparagraph.</p>";
```

Script.js

Manipulating DOM Elements

4. Removing Existing Elements from DOM

Index.html

```html
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>
```

Script.js

```javascript
let parentElem = document.getElementById("main");
let childElem = document.getElementById("hint");
parentElem.removeChild(childElem);
```

Manipulating DOM Elements

5. Replacing Existing Elements in DOM

Index.html

```html
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>
```

Script.js

```javascript
let parentElem = document.getElementById("main");
let oldPara = document.getElementById("hint");

let newPara = document.createElement("p");
let newContent = document.createTextNode("This is a new
paragraph.");
newPara.appendChild(newContent);

parentElem.replaceChild(newPara, oldPara);x`
```

Styling DOM Elements

Getting Style Information from Elements

Similarly, you get the styles applied on the HTML elements using the style property.

The following example will get the style information from the element having id="intro".

Index.html

```html
<p id="intro">This is a paragraph.</p>
<p>This is another paragraph.</p>
```

Script.js

```javascript
let elem = document.getElementById("intro");
alert(elem.style.color);
alert(elem.style.fontSize);
alert(elem.style.fontStyle);
```

Styling DOM Elements

Setting Inline Styles on Elements

Inline styles are applied directly to the specific HTML element using the style attribute. In JavaScript the style property is used to get or set the inline style of an element**.**

Index.html

```html
<p id="intro">This is a paragraph.</p>
<p>This is another paragraph.</p>
```

Script.js

```javascript
let elem = document.getElementById("intro");
elem.style.color = "blue";
elem.style.fontSize = "18px";
elem.style.fontWeight = "bold";
```

Styling DOM Elements

Naming Conventions of CSS Properties in JavaScript

Many CSS properties, such as font-size, background-image, text-decoration, etc. contain hyphens (-) in their names. Since, in JavaScript hyphen is a reserved operator and it is interpreted as a minus sign, so it is not possible to write an expression, like: elem.style.font-size

Therefore, in JavaScript, the CSS property names that contain one or more hyphens are converted to intercapitalized style word. It is done by removing the hyphens and capitalizing the letter immediately following each hyphen, thus the CSS property font-size becomes the DOM property fontSize, border-left-style becomes borderLeftStyle, and so on.

Styling DOM Elements

Adding CSS Classes to Elements

You can also get or set CSS classes to the HTML elements using the className property.

Style.css

```
.highlight {
    background: yellow;
 }
```

Index.html

```
<div id="info" class="disabled">Something very important!</div>
```

Script.js

```
let elem = document.getElementById("info");
elem.className = "note";
elem.className += " highlight";
```

Working with Attributes

Getting Element's Attribute Value

The getAttribute() method is used to get the current value of a attribute on the element.

```html
<button type="button" id="myBtn">Click Me</button>
```

Index.html

```javascript
// Selecting the element by ID attribute
let link = document.getElementById("myLink");
// Getting the attributes values
let href = link.getAttribute("href");
alert(href); // Outputs: https://www.google.com/
```

Script.js

Working with Attributes

Setting Attributes on Elements

The setAttribute() method is used to get the current value of a attribute on the element.

```html
<a href="https://www.google.com/" id="myLink">Google</a>
```

Index.html

```javascript
// Selecting the element
let btn = document.getElementById("myBtn");

// Setting new attributes
btn.setAttribute("class", "click-btn");
btn.setAttribute("disabled", "");
```

Script.js

Working with Attributes

Removing Attributes from Elements

The removeAttribute() method is used to remove an attribute from the specified element.

```html
<a href="https://www.google.com/" id="myLink">Google</a>
```

Index.html

```javascript
// Selecting the element
let link = document.getElementById("myLink");

// Removing the href attribute
link.removeAttribute("href");
```

Script.js

DOM Events

HTML DOM allows JavaScript to react to HTML events:

```html
<h1 onclick="this.innerHTML='Ooops!'">Click on this text!</h1>
```

Index.html

```html
<h1 onclick="changeText(this)">Click on this text!</h1>
```

Index.html

```javascript
function changeText(id) {
console.log(id)
id.innerHTML = "Ooops!";
};
```

Script.js

DOM Events

Assign Events Using the HTML DOM

Index.html

```html
<button id="myBtn">Try it</button>

<p id="demo"></p>
```

Script.js

```javascript
document.getElementById("myBtn").onclick = displayDate;

function displayDate() {
document.getElementById("demo").innerHTML = Date();
}
```

DOM Events

Mouse events

- mousedown
- mouseup
- mouseover
- mouseout
- mousemove
- click
- dblclick

Keyboard events

- onkeydown
- onkeypress
- onkeyup

Finished

Ramin Afhami
afhami.ramin@yahoo.com