Background Story

Odessa's journey into the world of web development takes an exciting turn as she learns to connect her JavaScript logic to a webpage. By integrating her script with the Document Object Model (DOM), Odessa can now manipulate the content live on the screen, bringing her applications to life.

One of the first tasks Odessa takes on is updating her organization's online bulletin board. With the power of JavaScript and the DOM, she can ensure that the bulletin board updates automatically with new information, making the website more dynamic and engaging for users.

As Odessa witnesses her script interact with the browser, changing text, styles, and even adding new elements to the webpage, she is amazed by the potential of this technology. This hands-on experience of bridging the gap between code and visual output empowers her to create impactful web solutions that resonate with users.

With topics like `document.querySelector`, `innerText`, and DOM events under her belt, Odessa is ready to DOM-inate the web and unlock a new level of creativity in her web development journey.

Theory & Lecture Content

The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of a document as a tree of objects that can be modified with scripting languages like JavaScript.

document.querySelector

`document.querySelector` is a method in JavaScript that allows us to select an element in the DOM using a CSS selector.

```
const heading = document.querySelector("h1");
```

innerText

The `innerText` property allows us to set or return the text content of an element and all its descendants.

```
heading.innerText = "Hello, World!";
```

DOM Events

DOM events are actions that occur as a result of user interaction or other events in the browser. We can add event listeners to elements to respond to these events.

```
button.addEventListener("click", function () {
  console.log("Button Clicked!");
});
```

For more information, you can visit the Mozilla Developer Network (MDN) documentation on Document Object Model (DOM).

Exercises

Exercise 1

Problem Statement

Select the element with the id `username` and change its text content to 'Welcome, Odessa!'

Starter Code

```
const usernameElement = document.querySelector("#username");
// Your code here
```

Full Solution Code

```
const usernameElement = document.querySelector("#username");
usernameElement.innerText = "Welcome, Odessa!";
```

Exercise 2

Problem Statement

Create a click event listener for the button with id `submitBtn`. When the button is clicked, change the text content of the element with id `message` to 'Form Submitted!'.

Starter Code

```
const submitBtn = document.querySelector("#submitBtn");
const messageElement = document.querySelector("#message");
// Your code here
```

Full Solution Code

```
const submitBtn = document.querySelector("#submitBtn");
const messageElement = document.querySelector("#message");

submitBtn.addEventListener("click", function () {
  messageElement.innerText = "Form Submitted!";
});
```

Exercise 3

Problem Statement

Create a function `toggleVisibility` that toggles the visibility of an element with id `toggleElement` when a button with id `toggleBtn` is clicked.

Starter Code

```
const toggleBtn = document.querySelector("#toggleBtn");
const toggleElement = document.querySelector("#toggleElement");

function toggleVisibility() {
  // Your code here
}
toggleBtn.addEventListener("click", toggleVisibility);
```

Full Solution Code

```
const toggleBtn = document.querySelector("#toggleBtn");
const toggleElement = document.querySelector("#toggleElement");

function toggleVisibility() {
  if (toggleElement.style.display === "none") {
    toggleElement.style.display = "block";
  } else {
    toggleElement.style.display = "none";
  }
}

toggleBtn.addEventListener("click", toggleVisibility);
```

Test Cases

Below are the Jest test cases, which you can run in the browser console to interact with the DOM.

```
test("Changing text content of username element", () => {
  usernameElement.innerText = "Welcome, Odessa!";
  // Verify manually on the webpage
});

test("Simulating click event on submit button", () => {
  submitBtn.click();
  expect(messageElement.innerText).toBe("Form Submitted!");
  // Verify manually on the webpage
});
```

```
// No test case for Exercise 3 as it involves visual changes
```

Closing Story

Odessa's exploration of the Document Object Model (DOM) opens up a whole new dimension in her web development journey. By connecting her JavaScript logic to the webpage, she can now create dynamic and interactive experiences for users, like the automatic update feature on her organization's bulletin board.

With the ability to target and modify elements on the webpage using methods like `document.querySelector` and properties like `innerText`, Odessa realizes the power of code in shaping the user experience. The thrill of seeing her script come to life on the screen fuels her passion for web development even more.

Join Odessa in the next lesson as she dives into more advanced topics like handling form submissions, manipulating styles with JavaScript, and implementing AJAX requests. As she continues to hone her skills, the possibilities of what she can achieve on the web become even more exciting.