Background Story

In the midst of maintaining her organization's online bulletin board, Odessa encounters a critical issue—a broken update that crashes the site. This unexpected setback teaches Odessa a valuable lesson in debugging, a skill that every developer must master to tackle unforeseen errors effectively.

Eager to restore the functionality of the bulletin board, Odessa dives into the world of debugging. She learns how to leverage the browser developer tools, specifically the console, to identify, track, and fix errors in her code. Through this experience, she discovers that errors are not indicative of failure but rather valuable feedback that can lead to growth and improvement.

The journey of debugging opens up a new perspective for Odessa, highlighting the importance of resilience and problem-solving in the world of web development.

Theory & Lecture Content

Mastering the art of debugging is essential for any developer. Here are some key topics that Odessa will explore:

try...catch Statement

The `try...catch` statement in JavaScript allows developers to handle exceptions (errors) gracefully. Code within the `try` block is executed, and if an error occurs, the `catch` block can handle the error.

Console

The console in browser developer tools is a powerful tool for debugging JavaScript code. Developers can log messages, variables, and errors to the console to track the flow of the program.

Error Messages

Understanding and deciphering error messages is crucial in debugging. Error messages provide valuable insights into what went wrong in the code and where the issue might be located.

Exercises

Exercise 1

Problem Statement

Create a function that attempts to parse a JSON string. Use a `try...catch` block to handle any parsing errors and log the error message to the console.

Starter Code

```javascript
const jsonString = '{"name": "Odessa", "age": 25}';
```

Full Solution Code

```
function parseJSON(jsonString) {
  try {
    const data = JSON.parse(jsonString);
    console.log("Parsed JSON:", data);
  } catch (error) {
    console.error("Error parsing JSON:", error.message);
  }
}

parseJSON(jsonString);
```

Exercise 2

Problem Statement

Debug the code snippet below by using the console to identify and fix the error.

```
let num1 = 10;
let num2 = 0;
let result = num1 / num2;
console.log("Result:", result);
```

Exercise 3

Problem Statement

Create a function that simulates an API call. Use a `try...catch` block to handle a potential error (e.g., network failure) and log an appropriate message to the console.

Starter Code

```
function fetchData() {
  // Simulating API call
  throw new Error("Network failure");
}

// Calling the function
fetchData();
```

Full Solution Code

```
function fetchData() {
  try {
    // Simulating API call
```

```
      throw new Error("Network failure");
    } catch (error) {
      console.error("API Call Error:", error.message);
    }
  }

  // Calling the function
  fetchData();
```

Test Cases

Due to the nature of debugging exercises, manual testing and verification using the browser console are recommended for these exercises.

Closing Story

Odessa's encounter with debugging not only helps her overcome the challenges in her bulletin board site but also provides her with a deeper understanding of error handling and problem-solving in programming. By embracing errors as feedback and learning opportunities, Odessa grows more resilient and adaptable in the face of technical challenges.

As she continues her web development journey, the skills and mindset she has developed through debugging pave the way for her to become a proficient developer who can navigate the complex world of software development with confidence and expertise.