Debugging

HyperionDev

Muhammad Zahir
Junejo

Lecture - Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
  - Please review Code of Conduct (in Student Undertaking Agreement) if unsure
- No question is daft or silly - **ask them!**
- Q&A session at the end of the lesson, should you wish to ask any follow-up questions.
- Should you have any questions after the lecture, please schedule a mentor session.
- For all non-academic questions, please submit a query: www.hyperiondev.com/support

Lecture Objectives

1. Role of debugging in ensuring robust and reliable code.
2. Harnessing stack traces, console debugging and browser developer tools for diagnosis.
3. Skillfully debugging JavaScript using breakpoints, step controls and variable inspection.

Importance of Debugging

- Debugging ensures code correctness, functionality, and user experience.
- It saves time by identifying issues early in the development process.
- Effective debugging skills are fundamental to becoming a proficient developer.

Stack Traces

- A stack trace is a detailed report of function calls leading to an error.
- Analyzing the stack trace helps pinpoint the exact location of the error.
- JavaScript automatically generates stack traces for unhandled exceptions.
- Interpreting Stack Traces:
  - Identify the error type and its description.
  - Review the line number and file name where the error occurred.
  - Ascertain the sequence of function calls that led to the error.

Using Console for Debugging

- ❏ console.log(): Print variables, objects, and messages for inspection.
- ❏ console.error(): Highlight errors in red to draw immediate attention.
- ❏ console.warn(): Emit warnings for potential issues that don't halt execution.

Browser Developer Tools

- Access developer tools with F12 or Ctrl+Shift+I in most browsers.
- Gain insights into the Document Object Model (DOM), network requests, and JavaScript execution.
- DevTools empower real-time analysis, performance profiling, and responsive design testing.

Inspecting Elements

- "Elements" panel: Examine and manipulate the HTML and CSS of your page.
- Hover over elements to visualize dimensions, positioning, and styles.
- Live-edit HTML and CSS for rapid iteration and experimentation.
- Right-click on an element and select "Break on" to pause execution when the element changes.

Debugging JavaScript

- Set breakpoints at specific lines to pause code execution for inspection.
- "Step Over," "Step Into," and "Step Out" buttons navigate through code execution.
- Inspect variables, watches, and the call stack for a deeper understanding of the code's behavior.
- "Network" panel: Monitor API requests, responses, and performance.
- "Console" panel: Execute JavaScript commands and review logs and errors.

QnA

Thank You!