

Week 3

❖ Topic 1: Introduction to JavaScript Modules

JavaScript modules are standalone units of code that can be reused in different applications.

They enable code organization and reusability, similar to camera lenses for different shooting situations in photography.

❖ Topic 2: Module Necessity

Before modules, all functions defined on the window object in JavaScript were global, leading to potential conflicts in larger projects.

The global scope could result in variable name clashes when multiple developers or third-party libraries were involved.

❖ Topic 3: History of Modules

The lack of built-in module functionality led to the development of CommonJS by Mozilla engineer Kevin Bangor.

CommonJS was designed for server-side JavaScript, primarily used in Node.js.

Browsers don't understand CommonJS syntax, causing issues for client-side code.

❖ Topic 4: ES6 Modules

ES6 (ECMAScript 2015) introduced native module support for browsers.

ES6 modules are encapsulated units of code that can be imported and used in other scripts.

They use import and export statements to specify dependencies and expose functionality.

❖ Topic 5: Working with ES6 Modules in HTML

HTML `<script>` tags can include JavaScript code.

The type attribute specifies the content type; for JavaScript, it's `type="text/javascript"`.

ES6 modules are identified with `type="module"`.

❖ Topic 6: Importing Modules

Modules can be imported using the import statement.

Example: `import greeting from './greeting.js';`

❖ Topic 7: Running HTML Files Over a Server

When running locally, browser security features like Cross-Origin Resource Sharing (CORS) can block access to modules.

Setting up a local server is recommended for running HTML files that import modules.

Week 3

❖Topic 8: Summary

JavaScript modules provide a way to organize and reuse code.

ES6 modules are the modern standard for organizing code in both browser and Node.js environments.

They use import and export statements to specify dependencies and expose functionality.

This summary covers the main topics and key information related to JavaScript modules, their history, and how to use ES6 modules in a browser environment.

JavaScript's Role: JavaScript was originally designed to give web developers control over webpage behavior and browser interactions. This purpose remains relevant today.

Evolution of JavaScript: JavaScript has evolved from plain JavaScript in the late 1990s to the introduction of libraries like jQuery in the mid-2000s. These libraries aimed to simplify web development by enabling developers to write less code while achieving more functionality.

Modern JavaScript Ecosystem: The reading mentions modern JavaScript frameworks and libraries like React, Vue, Angular, D3, and others that have further expanded the capabilities of web development. npm and Node.js are also highlighted as important components of the JavaScript ecosystem.

JavaScript's Versatility: JavaScript is praised for its versatility in enabling a wide range of interactive web experiences. It allows users to access geolocation, interact with maps, play browser-based games, handle user-triggered events across various devices, validate form input, and much more.

Conclusion: The reading concludes by reaffirming JavaScript's central role in creating rich and interactive web experiences. While CSS has evolved, it is JavaScript that remains essential for achieving dynamic and engaging web applications.

Week 3

1. Introduction to JSON:

JSON is a data interchange format based on JavaScript objects.

It is a lightweight and easily readable format used for data transfer between web services, websites, and databases.

2. JSON vs. XML:

JSON replaced XML as the preferred data interchange format due to its simplicity and efficiency.

JSON is more compact and easier to work with in JavaScript, making it a better fit for web development.

3. JSON as JavaScript:

JSON is often referred to as a subset of JavaScript because it adheres to JavaScript's syntax rules.

JSON data can be parsed into JavaScript objects, making it accessible and manipulable in JavaScript code.

4. JSON File Format:

JSON is not only used for data interchange but also as a file format.

Third-party data, such as stock prices, can be accessed in the form of JSON files and converted into JavaScript objects for further processing.

5. Data Access and Manipulation:

JSON allows developers to access and manipulate data in JavaScript code.

JSON data can be parsed into JavaScript objects, and specific information can be extracted based on criteria.

6. Proper JSON Format:

JSON must adhere to specific rules, including double-quoting keys, using commas to separate properties, and surrounding string properties in double quotes.

Accepted JSON values include strings, numbers, booleans, null, objects, and arrays (excluding functions).

Week 3

7. Limitations:

JSON cannot stringify certain data types, such as functions or BigInt numbers, and may result in errors or silent failures.

8. JSON Examples:

The reading provides examples of JSON strings, including simple key-value pairs, nested objects, and arrays.

In conclusion, the reading emphasizes JSON's importance as a data interchange format in web development. It explains why JSON has gained popularity over XML and how it seamlessly integrates with JavaScript, enabling developers to work with data efficiently in web applications.
