JAVASCRIPT OVERVIEW AND UNDERSTANDING JAVASCRIPT

JavaScript is a versatile, high-level programming language that is primarily known for its use in web development. It was initially created to enhance web pages by enabling interactive and dynamic content, but its capabilities have expanded significantly over the years. History and Evolution

- Creation:

  JavaScript was created by Brendan Eich in 1995 while he was working at Netscape Communications. It was originally called "Mocha," then "LiveScript" and finally “JavaScript”

- Standardization:

  JavaScript was standardized under the name ECMAScript (ES) by the European Computer Manufacturers Association (ECMA) in 1997. Since then, multiple versions of ECMAScript have been released, with ECMAScript 6 (ES6) in 2015 being a significant update.

Core Features:

- Dynamic Typing: JavaScript is dynamically typed, meaning variable types are determined at runtime.
- First-Class Functions: Functions in JavaScript are first-class citizens, meaning they can be assigned to variables, passed as arguments, and returned from other functions.
- Prototype-Based Inheritance: JavaScript uses prototype-based inheritance, which is different from classical inheritance models found in languages like Java or C++.
- Event-Driven: JavaScript is event-driven, which makes it suitable for handling events like user inputs, clicks, and other interactions in web applications.

Execution Environment

- Client-Side (Browser): JavaScript is most commonly run in the browser, where it interacts with the Document Object Model (DOM) to dynamically update web pages. Popular browsers like Chrome, Firefox, and Safari have JavaScript engines (e.g., V8 in Chrome) to execute JavaScript code.

- Server-Side (Node.js): With the advent of Node.js, JavaScript can also be run on the server side, allowing for the creation of full-stack applications using a single language.

Key Concepts

- Variables and Data Types: JavaScript supports primitive data types like strings, numbers, booleans, null, undefined, and the more complex types like objects and arrays.
- Functions and Scope: Functions can be defined using function declarations, expressions, or arrow functions. JavaScript also has lexical (block) scoping and closures.
- Error Handling: Error handling in JavaScript is typically done using try...catch blocksLibraries and Frameworks
- Frontend Frameworks: Libraries and frameworks like React, Angular, and Vue.js are built on JavaScript to create complex user interfaces.

- Backend Frameworks: On the server side, frameworks like Express.js (for Node.js) are commonly used to build APIs and handle server-side logic.

Ecosystem and Tools

- Package Management: NPM (Node Package Manager) is the default package manager for JavaScript, providing access to thousands of libraries and tools.
- Build Tools: Tools like Webpack, Babel, and Parcel are used to bundle and transpile JavaScript code, making it compatible with various browsers.
- Testing: JavaScript has a rich ecosystem of testing libraries like Jest, Mocha, and Jasmine for unit and integration testing.

Modern JavaScript (ES6 and Beyond)

- ES6 Features: Includes significant updates like arrow functions, template literals, classes, modules, let and const for variable declarations and destructuring.
- Newer Features: Recent versions of ECMAScript have introduced features like optional chaining, nullish coalescing, and advancements in asynchronous programming with async/await.

Popular Use Cases

- Web Development: JavaScript is essential for building interactive and responsive web pages.
- Mobile App Development: With frameworks like React Native, JavaScript can be used to develop cross-platform mobile apps.
- Server-Side Applications: Node.js allows JavaScript to be used for server-side scripting and backend development.
- Game Development: JavaScript is used in game development, particularly for web-based games.

Future of JavaScript

JavaScript continues to evolve with regular updates to the ECMAScript standard. The language's versatility and widespread use ensure that it remains a key technology in web and software development for the foreseeable future.

Challenges

- Browser Compatibility: Although modern browsers are more aligned, developers still need to consider compatibility issues.
- Security: JavaScript's ubiquity makes it a common target for security vulnerabilities, requiring best practices in coding and security measures.

JavaScript is a foundational technology in modern web development and beyond, with a constantly evolving ecosystem that supports a wide range of applications.

UNDERSTANDING DYNAMIC WEBSITES AND HTML 5 APPLICATIONS

The term dynamic refers to something active or something that motivates another person to become active. A dynamic website is one that incorporates interactivity into its functionality and design and that also motivates a user to take an action.

In web development, two types of scripting exist: server side and client side. Both types of scripting which is, in fact, a form of computer programming are beyond the scope of these lessons.

Some popular (and relatively easy-to-learn) server-side scripting languages include the following (to learn more, visit the websites listed):

- Perl—www.perl.org
- PHP (PHP: Hypertext Preprocessor)—www.php.net
- Python—www.python.org
- Ruby—www.ruby-lang.org

Including JavaScript in HTML

JavaScript code can live in one of two places in your files:

- In its own file with a .js extension
- Directly in your HTML files

Regardless of where your JavaScript lives, your browser learns of its existence through the use of the <script></script> tag pair.

When you store your JavaScript in external files, it is referenced in this manner:
<script src="/path/to/script.js"></script>

These <script></script> tags are typically placed between the <head></head> tags because, strictly speaking, they are not content that belongs in the <body> of the page.

Example of Using javascript to print some text

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>JavaScript Example</title>
    <meta name="viewport"
          content="width=device-width, initial-scale=1">
  </head>
  <body>
    <h1>JavaScript Example</h1>
    <p>This text is HTML.</p>
    <script>
      <!-- Hide the script from old browsers
      document.write('<p>This text comes from JavaScript.</p>');
      // Stop hiding the script -->
    </script>
    <p>And this text is HTML again.</p>
  </body>
</html>
```

Output of Using javascript to print some text

Thinking Ahead to Developing HTML5 Applications

Although HTML5 is incredibly rich, the creation of highly interactive HTML5 websites and applications including mobile applications doesn't happen in isolation. Interactivity comes when HTML5 is paired with a client-side language such as JavaScript, which then reaches back into the server and talks to a server-side language (PHP, Ruby, Python, and so on) through a persistent connection called a web socket.

With this connection open and talking to some server-side code that is (for example) talking to a database or performing some calculation, the browser can relay a bundle of information that is additionally processed by JavaScript and finally rendered in HTML5.

The depth of the technologies involved in HTML5 application creation is beyond the scope of these lessons, but the foundation you should have in standards-compliant HTML5, CSS3, and JavaScript will serve you well if you begin to think outside the box of a basic website.