DWA_02.8 Knowledge Check_DWA2

1. What do ES5, ES6 and ES2015 mean - and what are the differences between them?

- ES5 (ECMAScript 5): ES5, released in December 2009, was a significant update to the JavaScript language. It introduced several important features, such as: Strict Mode: A stricter variant of JavaScript that helps catch common coding mistakes and "unsafe" actions. Object Property Accessors, Array Methods:

- New methods like forEach, map, filter, reduce, and more, for easier manipulation of arrays.
- JSON Support: Built-in support for parsing and generating JSON data.
- Function.bind(): A method that allows explicit binding of the “this” value for functions.

- ES6 (ECMAScript 2015) or ES2015: ES6, released in June 2015, brought many significant enhancements to JavaScript. It introduced features that aimed to make the language more modern, expressive, and efficient.

- Arrow Functions: A concise syntax for writing anonymous functions.
- Block-Scoped Variables: Introducing let and const for block-level scoping of variables, reducing issues related to variable hoisting.
- Classes: A more structured and intuitive way to create object-oriented classes in JavaScript.
- Template Literals: Enhanced string interpolation capabilities.
- Spread and Rest Operators: Making it easier to work with arrays and function arguments.
- Destructuring: An elegant way to extract values from objects and arrays.
- Promises: A better approach to handling asynchronous operations compared to callbacks.
- Modules: A standardized way to organize code into separate files with import and export statements.

In summary, ES5, ES6 (or ES2015), and subsequent versions like ES2016, ES2017, and so on, are different iterations of the ECMAScript standard, each bringing new features and improvements to the JavaScript language. Developers often refer to these versions when discussing language features and compatibility.

2. What are JScript, ActionScript and ECMAScript - and how do they relate to JavaScript?

- JScript: JScript is a scripting language developed by Microsoft and is similar to JavaScript. It was originally designed for use in Microsoft's Internet Explorer browser. JScript shares many syntax and features with JavaScript because it's also based on the ECMAScript standard. While JScript is similar to JavaScript, it's primarily associated with older versions of Internet Explorer and is not as widely used or supported today. Modern web development has largely moved away from JScript in favor of JavaScript due to its wider compatibility and better support in modern browsers.
- ActionScript: ActionScript is a scripting language developed by Adobe Systems. It was originally designed for creating interactive content, animations, and applications within Adobe Flash (formerly Macromedia Flash). ActionScript is based on ECMAScript, much like JavaScript and JScript. While it shares some common concepts and syntax with JavaScript, it was specifically tailored for multimedia and interactive content creation. With the decline of Flash and the rise of HTML5, the use of ActionScript has diminished significantly.
- ECMAScript (ES): ECMAScript is a scripting language specification that defines the core features and syntax that scripting languages like JavaScript are based on. It's not an actual programming language itself, but a standardized specification that languages like JavaScript implement. ECMAScript provides rules for how the language should work, including how variables, functions, and objects should behave. JavaScript is one of the most popular implementations of the ECMAScript standard, but other scripting languages like JScript and ActionScript also follow the same standards.

In summary, JavaScript, JScript, ActionScript, and ECMAScript are scripting languages that share a common lineage through the ECMAScript standard. JavaScript is the most widely used and popular implementation of ECMAScript, while JScript and ActionScript are languages developed by Microsoft and Adobe respectively, both based on the same standard.

---

3. What is an example of a JavaScript specification - and where can you find it?

A JavaScript specification is a formal document that defines the rules, syntax, behavior, and features of the JavaScript programming language. It provides a comprehensive description of how the language should work and how its various components interact. One of the most well-known and widely used JavaScript specifications is the ECMAScript specification.

- Website: https://www.ecma-international.org/publications/standards/Ecma-262.htm

On this website, you can access the latest version of the ECMAScript specification, along with previous versions. The specification is presented in a detailed and technical manner, describing the language's syntax, semantics, and features in depth.

---

4. What are v8, SpiderMonkey, Chakra and Tamarin? Do they run JavaScript differently?

- V8 is a JavaScript engine developed by Google. It is primarily used in the Google Chrome browser and is also the engine behind the Node.js runtime.
- SpiderMonkey is a JavaScript engine developed by Mozilla. It is the engine used in the Mozilla Firefox browser. SpiderMonkey was one of the first JavaScript engines and has undergone significant improvements over the years.
- Chakra is a JavaScript engine originally developed by Microsoft. It was used in the Internet Explorer browser and later in the Microsoft Edge browser. Chakra underwent multiple iterations, and the newer versions (ChakraCore) were designed to be modular and embeddable, making them suitable for various platforms. Microsoft has since transitioned to using the Chromium-based engine (which uses V8) for the Edge browser, discontinuing the development of the standalone Chakra engine.
- Tamarin is a JavaScript engine that was initially developed by Adobe Systems and later contributed to the Mozilla project. It was designed to execute JavaScript and ActionScript (the scripting language used in Adobe Flash) in the Tamarin Virtual Machine. Tamarin aimed to improve the performance of ActionScript and JavaScript execution.

While these engines share the common goal of executing JavaScript, they may employ different optimization techniques, memory management strategies, and internal structures. Over time, many of them have converged in adopting similar approaches to improve performance and compatibility, driven by the competitive landscape of web browsers and the need for efficient execution of modern web applications.

---

5. Show a practical example using **caniuse.com** and the MDN compatibility table.

Can I Use (caniuse.com)

Step 1 : Go to caniuse.com
Step 2 : In the search bar, type "fetch" and select "Fetch API" from the dropdown.
Step 3: You'll see a detailed compatibility table for the Fetch API across various web browsers. The table shows the versions of browsers that support the feature and any specific notes about compatibility.

MDN Web Docs Compatibility Table:

Step 1: Go to the MDN Web Docs (https://developer.mozilla.org/).
Step 2: In the search bar, type "fetch" and select "Fetch API" from the search results.
Step 3: On the Fetch API page, scroll down to the "Browser compatibility" section. Here, you'll find a compatibility table similar to the one on Can I Use.

By using both “Can I Use” and the “MDN Web Docs”, you can get a comprehensive understanding of the browser compatibility of a particular feature. “Can I Use” provides an interactive way to quickly see compatibility across a wide range of features, while the MDN Web Docs offer detailed documentation and additional context on how to use the feature.
