# DWA\_02.8 Knowledge Check\_DWA2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

1. ES5 (ECMAScript 5) was released in December 2009.
2. ES6 (ECMAScript 6) / ES2015 was released in June 2015.
3. ES2015 is simply another name for ES6. The name reflects the new naming convention adopted by the ECMAScript committee to indicate the year of the release. So, ES6 and ES2015 are the same version of ECMAScript.
4. Differences Between ES5 and ES6/ES2015
   1. ES6/ES2015 introduced new syntax that makes the code more concise and readable, such as arrow functions, classes, and template literals.
   2. ES6/ES2015 provided better-scoping rules with 'let' and 'const', whereas ES5 primarily relied on 'var'.
   3. ES6/ES2015 introduced a native module system, while ES5 did not have built-in support for modules.
   4. ES6/ES2015 included Promises for better handling of asynchronous code, whereas ES5 primarily used callbacks.
   5. ES6/ES2015 introduced new data structures like Maps and Sets, as well as new utility methods for arrays and objects.

ES5 laid the foundation with essential features and improvements, while ES6/ES2015 introduced significant syntax enhancements and new capabilities that modernized JavaScript, making it more powerful and easier to use.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

1. ECMAScript: is a scripting language specification standardized by Ecma International in the ECMA-262 document. It defines the core language features and syntax. JavaScript is an implementation of the ECMAScript standard. This means JavaScript follows the rules and guidelines set by ECMAScript, adding additional features like the DOM (Document Object Model) to interact with web pages.
2. JScript: is Microsoft's proprietary implementation of the ECMAScript standard. It was created to be used in Microsoft's Internet Explorer browser and other Microsoft products like Active Server Pages (ASP). While JScript is mostly compatible with ECMAScript, it includes additional features and extensions specific to the Windows environment and Internet Explorer.
3. ActionScript: is an implementation of the ECMAScript standard used primarily for Adobe Flash applications and Adobe AIR. Designed for controlling animations, multimedia, and other interactive elements in Flash-based applications. Although based on ECMAScript, ActionScript includes features tailored to the Flash environment, such as more robust graphics and animation handling capabilities.

ECMAScript is the standardized language specification that serves as the foundation for various scripting languages, including JavaScript, JScript, and ActionScript. JavaScript is the most widely used implementation, evolving continuously with the ECMAScript updates. JScript and ActionScript are specialized implementations created for specific environments (Microsoft and Adobe Flash, respectively) but are less common in modern web development.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

The ECMAScript 2015 (ES6) specification is a crucial update to JavaScript, introducing features like block-scoped variables, arrow functions, classes, template literals, destructuring, modules, and promises. These enhancements make JavaScript more robust, easier to use, and maintainable. You can find detailed ECMAScript specifications also on the Ecma International website, which serves as the authoritative source for all versions of the ECMAScript standard.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

V8: Was developed by Google. Compiles JavaScript to native machine code at runtime, which significantly improves performance. It includes a garbage collector to manage memory efficiently, cleaning up unused objects to free memory. Uses advanced optimization techniques, such as inline caching and hidden classes, to speed up JavaScript execution. Known for its high performance and is widely used in both browsers and server-side environments (Node.js). It has a strong focus on speed and efficiency, making it a popular choice for performance-critical applications.

SpiderMonkey: This was developed by Mozilla. Combines an interpreter for quick startup and a JIT compiler for faster execution of frequently run code. Implements incremental and generational garbage collection to manage memory efficiently. Known for closely following ECMAScript standards and incorporating new features early.  Is the first-ever JavaScript engine, developed for the Netscape browser and later adopted by Mozilla for Firefox. It focuses on maintaining a balance between performance and compatibility with web standards.

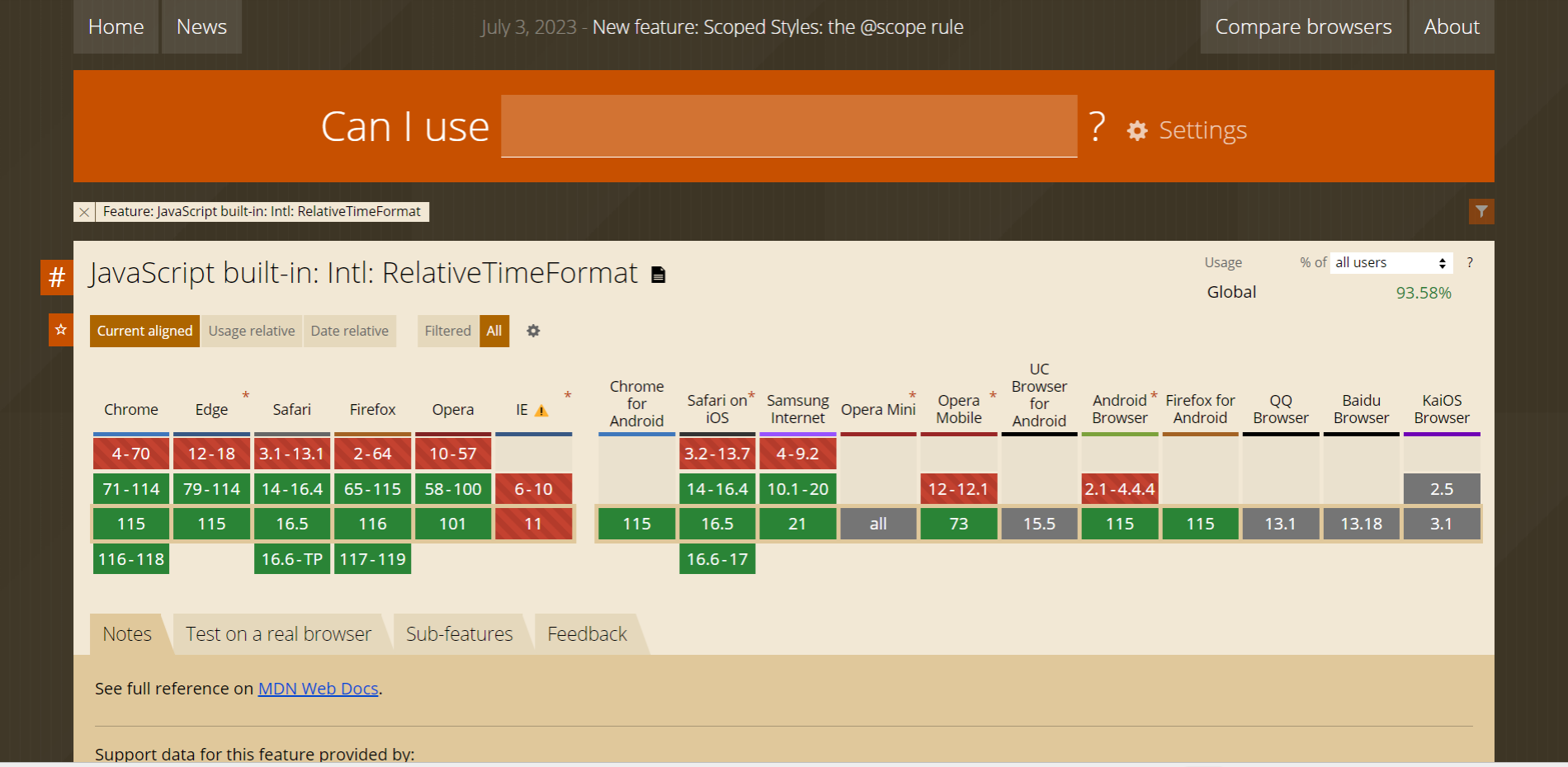
Chakra: This was developed by Microsoft. Utilizes JIT compilation to convert JavaScript into optimized machine code at runtime. Chakra can use multiple cores for JIT compilation, which can improve performance on modern multi-core processors. Supports AOT compilation for faster startup times. Was developed to improve JavaScript performance in Internet Explorer and later in the legacy version of Microsoft Edge. It introduced several innovations, like multi-core JIT compilation, to leverage the full potential of modern hardware.

Tamarin: Was developed by Adobe (in collaboration with Mozilla). Primarily designed to execute ActionScript, a language used for scripting in Adobe Flash. Includes various performance optimizations tailored for rich media applications. It is less relevant today due to the decline of Adobe Flash, which has been deprecated and is no longer supported in modern web browsers. It was used to power the scripting in Flash, focusing on executing ActionScript efficiently rather than general JavaScript.

V8 focuses heavily on performance and uses advanced optimization techniques to ensure JavaScript runs as fast as possible. SpiderMonkey balances performance with strict adherence to ECMAScript standards. Chakra introduced innovative features like multi-core JIT compilation to enhance performance. Tamarin was optimized for running ActionScript in multimedia applications. Each engine uses different garbage collection strategies tailored to their performance and memory usage goals. V8 and SpiderMonkey employ sophisticated garbage collection algorithms to handle memory efficiently during JavaScript execution. V8 is optimized for both browser (Chrome) and server-side (Node.js) environments, supporting a wide range of JavaScript applications. SpiderMonkey is integrated into Firefox and focuses on web standards and browser-specific optimizations. Chakra was designed for Microsoft's browsers and integrates deeply with the Windows operating system and Tamarin specialized for the Adobe Flash environment, focusing on the needs of rich multimedia content.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Show a practical example using **[caniuse.com](http://caniuse.com/)** and the MDN compatibility table.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_