DWA_02.8 Knowledge Check_DWA2

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

ES5, ES6, and ES2015 are all references to different versions of the ECMAScript (ES) specification, which is the standard for the JavaScript language. Each version introduces new features, syntax improvements, and functionality to enhance the language.

ES5 (ECMAScript 5):

ES5 was released in December 2009. It brought significant improvements to the language, introducing features like strict mode, array methods (e.g., forEach, map, filter, etc.), JSON support, and various new methods for string manipulation.
Notable features in ES5 include the bind, call, and apply methods for function context manipulation, and the Object.create method for object creation with custom prototypes.

ES6 (ECMAScript 2015):

ES6, also known as ECMAScript 2015, was released in June 2015. This was a major update to the language, introducing many new features and syntax enhancements, making JavaScript more expressive and powerful.
Some notable additions in ES6 include the let and const keywords for block-scoped variable declarations, arrow functions, classes, template literals, destructuring assignments, spread and rest operators, and enhanced object literals.
ES6 also introduced Promises for improved asynchronous programming and introduced new array methods such as `find`, `findIndex`, and `entries`.

ES2015 (ECMAScript 2015): ES2015 is simply another name for ES6. It is the same release as ECMAScript 2015, but the naming convention was changed to reflect the yearly release cycle starting from ES2015. After ES2015, ECMAScript started adopting a new release process, with new versions being named after the year of their release (e.g., ES2016, ES2017, etc.).

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

JScript: JScript is a Microsoft implementation of the ECMAScript language specification. It was Microsoft's version of JavaScript and was used primarily in Internet Explorer web browsers. JScript was introduced in the mid-1990s as a scripting language for web development and is similar to JavaScript in terms of syntax and functionality. However, there were some differences in implementation and additional features specific to JScript. With the decline of Internet Explorer and the rise of other modern browsers, JScript has become less relevant in modern web development, and developers now focus more on using standard JavaScript.

ActionScript: ActionScript is a scripting language developed by Adobe Systems and used primarily in Adobe Flash applications. While ActionScript has its roots in ECMAScript (specifically ECMAScript 3), it was a separate language with additional features and capabilities beyond the core ECMAScript specifications. ActionScript was widely used for creating interactive content, animations, and games within Flash applications. However, with the decline of Flash and the rise of HTML5 and modern web technologies, ActionScript has lost much of its popularity and support.

ECMAScript: ECMAScript is a standardized scripting language specification that defines the features and behavior of JavaScript. JavaScript is the most well-known and widely used implementation of the ECMAScript language. It is the client-side scripting language for web browsers and is widely used for web development. ECMAScript is not tied to any specific platform or implementation. Instead, it serves as a language specification that browser vendors and other environments (like Node.js) can use to create their JavaScript engines. The ECMAScript specification is maintained by Ecma International (formerly known as the European Computer Manufacturers Association), and it is regularly updated to introduce new features and improvements to the language.

In summary, JScript, ActionScript, and ECMAScript are all related to JavaScript, but they represent different implementations or variations of the same language. JavaScript is the most popular and widely used implementation of ECMAScript, while JScript and ActionScript were specific implementations used in Internet Explorer and Adobe Flash, respectively. As JavaScript evolved and became the de facto language for web development, JScript and ActionScript gradually lost relevance, and developers now mainly focus on ECMAScript and standard JavaScript for modern web development

---

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

One example of a JavaScript specification is the **ECMAScript** Language Specification. The ECMAScript Language Specification defines the syntax, semantics, and standard library for the JavaScript language. It serves as the official reference for JavaScript and its various implementations. You can find the ECMAScript Language Specification online on the Ecma International website. The organization responsible for maintaining the ECMAScript specification is Ecma International, which is a standards organization that develops and publishes international standards. The ECMAScript specification is regularly updated with new features and improvements to the language.

---

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

V8, SpiderMonkey, Chakra, and Tamarin are JavaScript engines used in various web browsers and platforms to execute JavaScript code. They are responsible for parsing and executing JavaScript programs, and each engine has its unique characteristics and performance optimizations.

V8:

V8 is the JavaScript engine developed by Google and used in the Chrome web browser and Node.js runtime. It is written in C++ and is highly optimized for performance.
V8 uses Just-In-Time (JIT) compilation, where JavaScript code is first translated into intermediate machine code and then executed directly by the CPU. This approach enables V8 to execute JavaScript at near-native speeds.
V8 incorporates various optimization techniques, such as inline caching, hidden classes, and adaptive compilation, to further improve performance and reduce execution time.

SpiderMonkey:

SpiderMonkey is the JavaScript engine developed by Mozilla and used in the Firefox web browser. It was the first-ever JavaScript engine, created by Brendan Eich in 1995.

Like V8, SpiderMonkey also uses Just-In-Time compilation, which translates JavaScript code into machine code for faster execution.
SpiderMonkey has evolved over the years and includes various optimizations, such as JIT warm-up, type inference, and parallel compilation, to enhance performance.

Chakra:

Chakra was the JavaScript engine developed by Microsoft for Internet Explorer and later Microsoft Edge browsers. It has been replaced by the V8-based engine in the Chromium-based Microsoft Edge browser.
Chakra used a combination of JIT and pre-compilation (interpreting frequently used code paths) to execute JavaScript code.

Tamarin:

Tamarin was a JavaScript engine developed by Adobe Systems for the Adobe Flash Player and the ActionScript language (a variant of ECMAScript used in Flash). Tamarin used a unique tracing JIT compilation technique, which provided high-performance execution for ActionScript code.

Each JavaScript engine implements the ECMAScript specification and is optimized to run JavaScript code efficiently. While the core functionality and behavior are the same, the performance and execution speed can vary between different engines due to their various optimization strategies. Additionally, as technology evolves and new standards are introduced (such as ECMAScript 6 and later versions), JavaScript engines need to keep pace with these changes to ensure compatibility and performance improvements.

---

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

Caniuse.com :

Step 1: Open your web browser and go to caniuse.com.
Step 2: In the search bar, type "fetch" and select the "Fetch API" from the suggestions.
Step 3: The caniuse.com page for the Fetch API will display detailed information about browser support for this feature.

Step 4: You'll see a table showing the support for fetch in various web browsers, including Chrome, Firefox, Safari, Edge, and others. The table will indicate whether the feature is fully supported, partially supported, or not supported at all in each browser version.
Step 5: The table will also show data for different versions of the browser, and you can click on specific versions to see more details about their support.

MDN compatibility table:

Step 1: Open your web browser and go to the MDN web docs (developer.mozilla.org).
Step 2: In the search bar, type "fetch" and select the "Fetch API" from the suggestions.
Step 3: The MDN documentation page for the Fetch API will provide information about the feature, including its usage, syntax, and compatibility.
Step 4: Scroll down to the "Browser compatibility" section. The compatibility table will display support information for the Fetch API across various web browsers.
Step 5: The table will show the level of support for each browser version, indicating whether the feature is supported, partially supported, or not supported.
