DWA_02.8 Knowledge Check_DWA2

---

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

ES5 and ES6 refer to different versions of ECMAScript, the standardized scripting language that JavaScript is based on.ES6 is often used interchangeably with ES2015 since it was the major release that brought significant changes to the language. However, it's worth noting that ECMAScript has continued to evolve since ES6, with newer versions like ES2016, ES2017, and so on, introducing additional features and improvements like introduced features like arrow functions, classes, modules, and template literals.

---

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

JScript, ActionScript, and ECMAScript are all scripting languages with varying degrees of relationship to JavaScript.

> **JScrip**t: JScript is a dialect of ECMAScript developed by Microsoft. It is primarily used in Microsoft's Internet Explorer web browser.
> **ActionScript**: ActionScript is a scripting language derived from ECMAScript. It was primarily used for developing applications and games in Adobe Flash and Adobe AIR.
> **ECMAScript**: ECMAScript is the standard scripting language specification upon which JavaScript is based. JavaScript adheres to the ECMAScript specification, and new versions of JavaScript often align with updates to ECMAScript.

JavaScript is the most widely known and used implementation of ECMAScript, and it is the de facto scripting language for 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 specification, which defines the rules, syntax, and behavior of the JavaScript language. It provides a detailed documentation of how JavaScript should work, including its features, built-in objects, and methods.

https://262.ecma-international.org/13.0/

---

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

V8, SpiderMonkey, Chakra, and Tamarin are JavaScript engines, each with its own implementation of the JavaScript language and unique characteristics. They are responsible for executing JavaScript code in different environments.

V8: V8 is the JavaScript engine developed by Google. It is known for its high-performance execution and is used in the Google Chrome browser and the Node.js runtime environment. V8 uses just-in-time (JIT) compilation to optimize JavaScript code and provides efficient memory management.

SpiderMonkey: SpiderMonkey is the JavaScript engine developed by Mozilla. It was one of the first JavaScript engines ever created and is used in the Firefox web browser. SpiderMonkey also utilizes JIT compilation techniques to execute JavaScript code efficiently.

Chakra: Chakra is the JavaScript engine developed by Microsoft. It was used in Internet Explorer and later in the Microsoft Edge browser. Chakra employed a combination of both JIT compilation and interpreter-based execution to optimize JavaScript performance.

Tamarin: Tamarin is a JavaScript engine developed by Adobe. It was specifically designed for executing ActionScript (a language similar to ECMAScript) in Adobe Flash

Player. Tamarin utilized a just-in-time (JIT) compiler called NanoJIT to optimize and execute ActionScript code.

While all these JavaScript engines run JavaScript code, they have different implementations and optimizations strategies. Each engine employs techniques such as JIT compilation, interpreter-based execution, garbage collection, and memory management to improve performance and provide a runtime environment for executing JavaScript code. However, the specific optimizations and internal workings may vary between these engines, resulting in performance differences and unique capabilities in running JavaScript.

---

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

JavaScript operator: import

The `import` statement is not an operator in JavaScript; it is a language feature used for importing modules. However, the `import` statement is not supported in the browser's JavaScript environment directly. Instead, browsers use a different mechanism called `<script>` tags to load and execute JavaScript files.

To use modules in the browser, you can use the `<script>` tag with the `type="module"` attribute. Here's an example:

```html
<!DOCTYPE html>
<html>
<head>
  <title>Module Example</title>
</head>
<body>
  <script type="module">
    import { myFunction } from './myModule.js';

    myFunction();
  </script>
</body>
</html>
```

In the above example, the `myModule.js` file is a separate JavaScript module that exports a function `myFunction`. The `import` statement is used to import that function into the main script and then call it.

Most modern browsers support JavaScript modules. However, it's always a good practice to check the compatibility table for the specific features you plan to use, as browser support may vary.

JavaScript built-in: Array: pop

The `Array.prototype.pop()` method is a built-in JavaScript function that removes the last element from an array and returns that element. It is a standard part of the JavaScript language and is supported by all modern web browsers.

You can use the `pop()` method on arrays in any browser that has JavaScript support, including popular browsers like Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, and others. This method is part of the ECMAScript standard, which defines the JavaScript language, and is widely supported across different environments, including web browsers.

Here's an example of using the `pop()` method in JavaScript:

```javascript
const myArray = [1, 2, 3, 4, 5];
const poppedElement = myArray.pop();

console.log(myArray);        // Output: [1, 2, 3, 4]
console.log(poppedElement);  // Output: 5
```

In the above example, `pop()` is called on the `myArray` variable, and the last element (5) is removed from the array. The `pop()` method also returns the popped element, which is stored in the `poppedElement` variable and printed to the console.