DWA_02.8 Knowledge Check_DWA2

---

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

The European Computer Manufacturers Association(ECMA) created a standardized specification for Netscape's JavaScript and all its variants. So ES5, ES6 and ECMAScript 2015 are the names used for each variant or edition of the ECMA-262 standard and each edition features major changes and improvements to the ECMAScript Specification.

ES5 is the fifth edition to the ECMA-262 Standard which includes significant improvements and new features such as strict mode, JSON and array methods.

ES6 which is also known as ES2015 or ECMAScript 2015 which refers to the year when ES6 was finalized. This is the sixth edition to the ECMA-262 Standard and it includes major changes, new features and syntax improvement such as arrow functions, classes, template literal, block-scoped variable, modules and more.

---

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

- JScript was the scripting programming language created by Microsoft. It was introduced as Microsoft's implementation of JavaScript for their Internet Explorer browser. JScript was similar to JavaScript but some differences were proprietary features and syntax differences and it was mainly used for client-side scripting within the Internet Explorer browser.
- ActionScript was a scripting language originally created by Macromedia. It was mainly used for the development of websites and software focused on the Adobe Flash platform. ActionScripts shared similarities with JavaScript mainly being its syntax and Object-Oriented Programming features but it had distinct features and functionality specific to the Flash platform.
- ECMAScript is the standardized scripting language on which JavaScript is based. It provides the rules, details and guidelines that a scripting language must observe for it to be considered ECMAScript compliant.

JavaScript, JScript and ActionScript are the implementations of the ECMAScript scripting language.

---

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

An example of a JavaScript Specification is the ECMAScript2021 or ES12 language which is the 12th edition of the ECMAScript Language Specification. This edition introduced several new features and improvements including:

- String.prototype.replaceAll() method which is used to replace all occurrences of a substring in a string with another substring.
- Promise.any() method which takes an iterable of promises as input and returns a single promise.
- Logical assignment operators that combine logical operators with an assignment. (eg. ??=, &&=, ||=)

The official JavaScript Specification can be found on the ECMA Internal Website which is titled ECMA-262 (technical name for ECMAScript Standard) under the Standards tab. There you will find all the specifications including ECMA-262 12th edition. These specifications can be used as a comprehensive and detailed reference for JavaScript Developers.

---

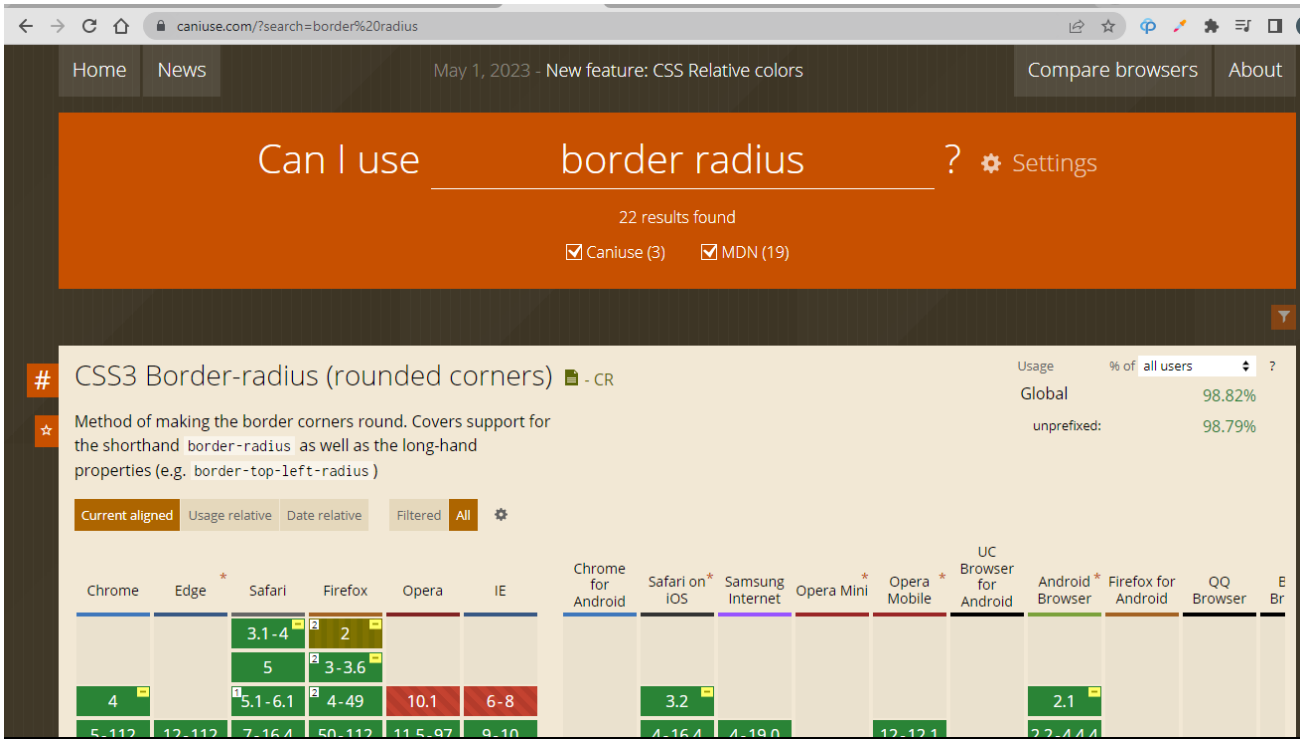
4. What are v8, SpiderMonkey, Chakra and Tamarin? Do they run JavaScript differently?

V8, spiderMonkey, Chakra and Tamrin are JavaScript engines or Javascript Implementations which is basically a program or interpreter that understands and executes JavaScript. They are commonly found on web browsers: v8 will be found in Google Chrome, spiderMonkey and Tamarin will be found in Mozilla Firefox and Chakra will be found in Microsoft Edge. Each JavaScript engine is a language module for its application which allows it to support a certain subset of the JavaScript language. The engines implement JavaScript in slightly different ways. A few ways in which they can differ is:

- Understanding code. JavaScript engines analyze structure and meaning of JavaScript Code and doing this it may be interpreted slightly differently and result in small differences in behavior and error handling.
- Management memory. JavaScript engines handle memory management meaning they keep track of assigned memory and freeing up memory that’s no longer needed, the strategies differ which impacts the memory usage and performance.
- Supporting JavaScript. JavaScript engines may support different JavaScript features and versions and some engines adopt new features sooner than others, Which impacts compatibility and availability of certain features.

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

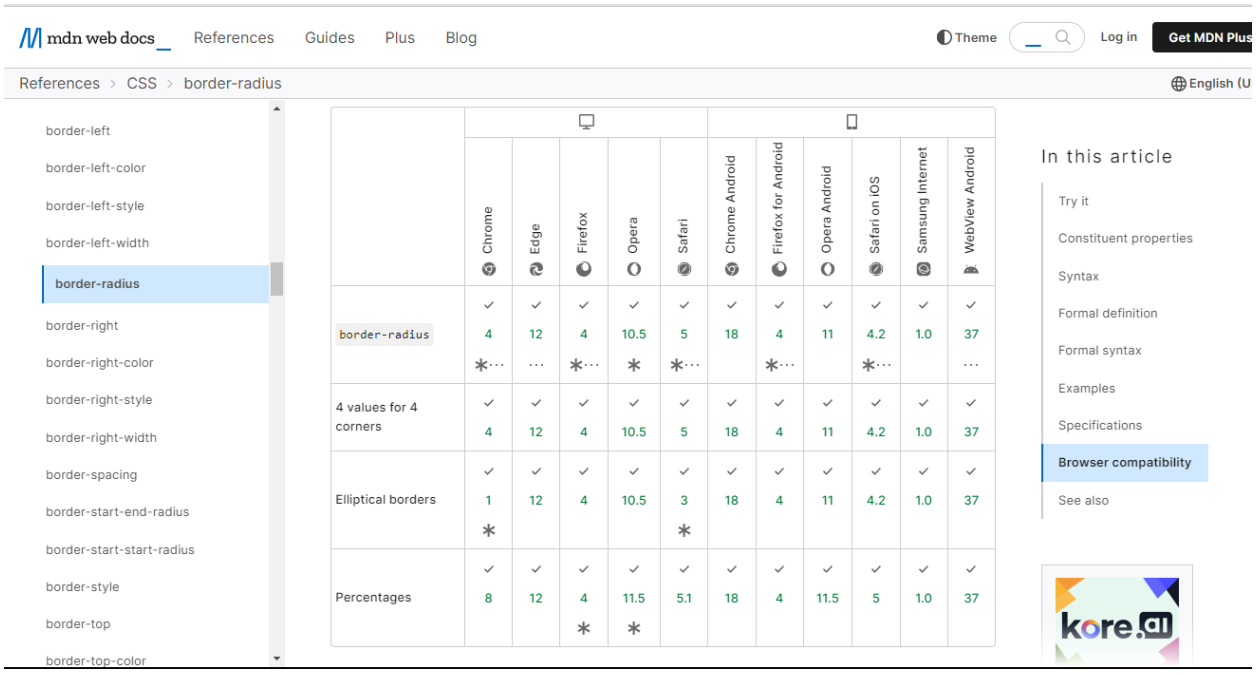
For example, if I am working on a project that involves using the border-radius css property. I want to ensure that the border-radius property is supported across different browsers before using it in my project. Using the canisue.com website, I will start by typing in the search “border-radius” which will take me to the border-radius feature page.



Once I search for the border-radius property, I will see a detailed overview of the feature’s support across different browsers. It will provide a compatibility table showing the percentage of global browser usage that supports “border-radius”. It also provides information about specific browser versions and the compatibility with the border-radius property

On the screenshot above, there is a section at the bottom called “known issues”. This highlights any specific bugs or issues related to the border-radius in certain browsers which can be useful for troubleshooting and looking for workarounds if needed.

Now if I move to the MDN compatibility table for the border-radius, I will see a visual representation of the browser support for border-radius



So the MDN compatibility table is almost similar to the one on caniuse.com. Some differences is that it shows if the property is supported by using a “tick” symbol and includes additional information and details about each version’s browser support when you click on it. Although I used the external MDN website, you will also find the MDN compatibility data on the caniuse.com website as it has been integrated in the website. Here is an screenshot of the MDN compatibility data:

Overall, using both the caniuse and the MDN compatibility table, I was able to get a comprehensive view of the border-radius property support across different browsers. This information can help make informed decisions about using border-radius in my project and see if I need to consider fallback options.
