Window.getComputedStyle()

The **`Window.getComputedStyle()`** method returns an object containing the values of all CSS properties of an element, after applying active stylesheets and resolving any basic computation those values may contain. Individual CSS property values are accessed through APIs provided by the object, or by indexing with CSS property names.

Syntax

```
var style = window.getComputedStyle(element [, pseudoElt]);
```

`element`
: The `Element` for which to get the computed style.

*`pseudoElt`*Optional
: A string specifying the pseudo-element to match. Omitted (or `null`) for real elements.

The returned `style` is a *live* `CSSStyleDeclaration` object, which updates automatically when the element's styles are changed.

Throws

`TypeError`
: If the passed object is not an `Element` or the *`pseudoElt`* is not a valid pseudo-element selector or is `::part()` or `::slotted()`.

: **Note:** Valid pseudo-element selector refers to syntactic validity, e.g. `::unsupported` is considered valid, even though the pseudo-element itself is not supported.

Examples

In this example we style a `<p>` element, then retrieve those styles using `getComputedStyle()`, and print them into the text content of the `<p>`.

HTML

```
<p>Hello</p>
```

CSS

```
p {
  width: 400px;
  margin: 0 auto;
  padding: 20px;
  font: 2rem/2 sans-serif;
  text-align: center;
  background: purple;
  color: white;
}
```

JavaScript

```
let para = document.querySelector('p');
let compStyles = window.getComputedStyle(para);
para.textContent = 'My computed font-size is ' +
    compStyles.getPropertyValue('font-size') +
    ',\nand my computed line-height is ' +
    compStyles.getPropertyValue('line-height') +
    '.';
```

Result

Description

The returned object is the same `CSSStyleDeclaration` type as the object returned from the element's `style` property. However, the two objects have different purposes:

- The object from `getComputedStyle` is read-only, and should be used to inspect the element's style — including those set by a `<style>` element or an external stylesheet.
- The *`element`*`.style` object should be used to **set** styles on that element, or inspect styles directly added to it from JavaScript manipulation or the global `style` attribute.

The first argument must be an `Element`. non-elements, like a `Text` node, will throw an error.

defaultView

In many code samples, `getComputedStyle` is used from the `document.defaultView` object. In nearly all cases, this is needless, as `getComputedStyle` exists on the `window` object as well. It's likely the `defaultView` pattern was a combination of folks not wanting to write a testing spec for `window` and making an API that was also usable in Java.

Use with pseudo-elements

`getComputedStyle` can pull style info from pseudo-elements (such as `::after`, `::before`, `::marker`, `::line-marker` — see the pseudo-element spec).

```
<style>
  h3::after {
    content: ' rocks!';
  }
</style>

<h3>Generated content</h3>

<script>
  var h3 = document.querySelector('h3');
  var result = getComputedStyle(h3, ':after').content;

  console.log('the generated content is: ', result);
</script>
```

Notes

- The returned `CSSStyleDeclaration` object contains active values for CSS property ***longhand*** names. For example, `border-bottom-width` instead of the `border-width` and `border` shorthand property names. It is safest to query values with only longhand names like `font-size`. Shorthand names like `font` will not work with most browsers.
- CSS property values may be accessed using the `getPropertyValue(propName)` API or by indexing directly into the object such as `obj['z-index']` or `obj.zIndex`.
- The values returned by `getComputedStyle` are `resolved values`. These are usually the same as CSS 2.1's `computed values`, but for some older properties like `width`, `height`, or `padding`, they are instead the same as `used values`. Originally, CSS 2.0 defined the *computed values* as the "ready to be used" final values of properties after cascading and inheritance, but CSS 2.1 redefined them as pre-layout, and *used values* as post-layout. For CSS 2.0 properties, `getComputedStyle` returns the old meaning of computed values, now called **used values**. An example difference between pre- and post-layout values includes the resolution of percentages for `width` or `height`, as those will be replaced by their pixel equivalent only for *used values*.
- Returned values are sometimes deliberately inaccurate. To avoid the "CSS History Leak" security issue, browsers may lie about the computed styles for a visited link, returning values as if the user never visited the linked URL. See Plugging the CSS History Leak and Privacy-related changes coming to CSS :visited for examples of how this is implemented.
- During CSS transitions, `getComputedStyle` returns the original property value in Firefox, but the final property value in WebKit.
- In Firefox, properties with the value `auto` return the used value, not the value `auto`. So if you apply `top:auto` and `bottom:0` on an element with `height:30px` and a containing block of `height:100px`, Firefox's computed style for `top` returns `70px`, as 100 − 30 = 70.

Specifications

Specification	Status	Comment
CSS Object Model (CSSOM) The definition of 'getComputedStyle()' in that specification.	Working Draft	
Document Object Model (DOM) Level 2 Style Specification The definition of 'getComputedStyle()' in that specification.	Obsolete	Initial definition

Browser compatibility

The compatibility table in this page is generated from structured data. If you'd like to contribute to the data, please check out https://github.com/mdn/browser-compat-data and send us a pull request.

Update compatibility data on GitHub

	Desktop						Mobile			
	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Android webview	Chrome for Android	Firefox for Android	Opera for Andro
`getComputedStyle`	Chrome Full support 1	Edge Full support 12	Firefox Full support 1 Notes Full support 1 Notes Notes Before version 62 this function returned `null` when called on a Window with no presentation (e.g. an iframe with `display: none;` set). Since 62 it returns a `CSSStyleDeclaration` object with `length` 0, containing empty strings (bug 1467722; also see bug 1471231 for further work).	IE Full support 9	Opera Full support 7.2	Safari Full support 3	WebView Android Full support 1	Chrome Android Full support 18	Firefox Android Full support 4 Notes Full support 4 Notes Notes Before version 62 this function returned `null` when called on a Window with no presentation (e.g. an iframe with `display: none;` set). Since 62 it returns a `CSSStyleDeclaration` object with `length` 0, containing empty strings (bug 1467722; also see bug 1471231 for further work).	Opera Androi Full support 10.1
Pseudo-element support	Chrome Full support Yes	Edge Full support 12	Firefox Full support Yes	IE Full support 9	Opera Full support 15	Safari Full support Yes	WebView Android Full support Yes	Chrome Android Full support Yes	Firefox Android ?	Opera Androi ?

What happens next?

Our team will review your report. Once we verify the information you have supplied we will update this browser compatability table accordingly.

Can I keep track of my report?

You can join the GitHub repository to see updates and commits for this table data:

https://github.com/mdn/browser-compat-data

Our goal is to provide accurate, real values for all our compatibility data tables. Notifying MDN of inaccurate data or supplying new data pushes us further towards our goal of providing 100% real values to the developer community.
Thank you for helping.

Please select the browser or browsers which are affected.

Briefly outline the issue you are highlighting. Minimum 10 and maximum 1,000 characters.

Browser documentation and release notes are good supporting items to accompany your message. A demo hosted on services like Codepen or JSBin are perfect for providing real examples of your findings.

Connection error:Sorry, we can't seem to reach the server. We are working to fix the problem. Please try again later.

Legend

Full support
    Full support
Compatibility unknown
    Compatibility unknown
See implementation notes.
    See implementation notes.

See also

- `window.getDefaultComputedStyle`
- `resolved value`