JavaScript Popover API

Summary: in this tutorial, you will learn how to use the JavaScript Popover API to show popover content on top of other elements on a webpage.

Introduction to JavaScript Popover API

The Popover API is a standard way to show popover content on top of other page elements. There are two ways to control the popover content:

- Declare HTML attributes.
- Use JavaScript.

Declarative popovers

To make popover content out of an HTML element, you add the `popover` and `id` attribute to it:

```
<div id="popover" popover>This is a popover</div>
```

In this syntax:

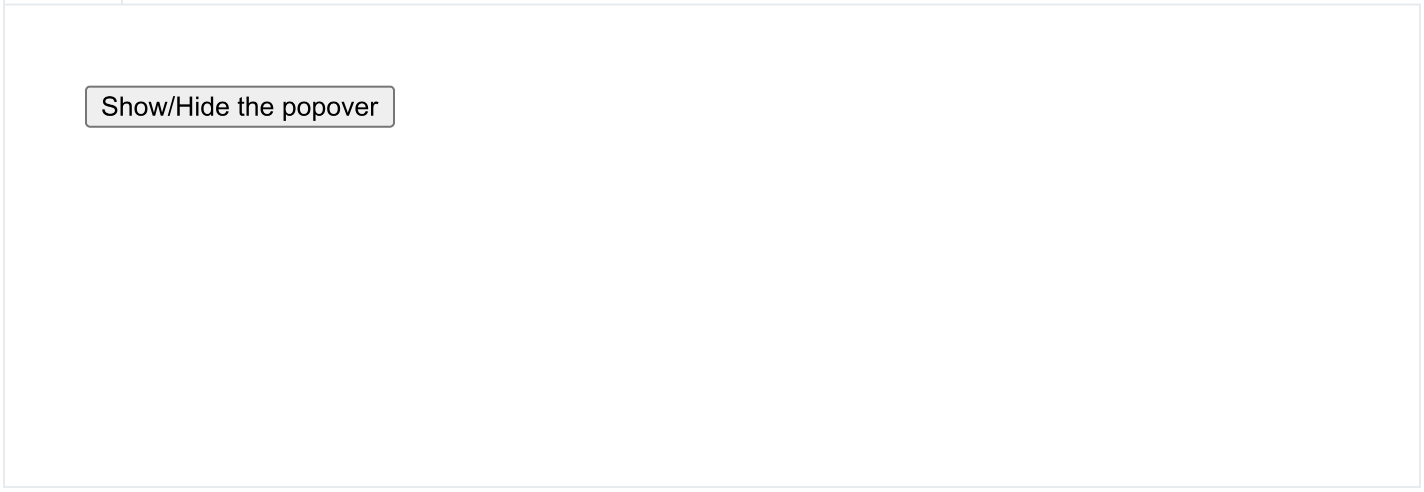
- The `popover` attribute implicitly sets the `display` property to `none` , which causes the element to be hidden on page load.
- The `id` is used to associate the popover element with its control, which can be a button.

To show or hide a popover, you need another element which can be a `button` or an `input` with the type `button` . Additionally, you need to set the `popovertarget` attribute of the button to the `id` of the popover. For example:

```
<div id="popover" popover>This is a popover</div>
<button popovertarget="popover">Show/Hide the popover</button>
```

Now, if you click the button, it will show the popover, if you click the button again, it will hide the popover. The default behavior of the popover is a **toggle**:

Output



You can change the behavior of the popover element by setting the `popovertargetaction` attribute of the button to `hide`, `show`, or `toggle`. For example:

```html
<div id="popover" popover>This is a popover</div>
<button popovertarget="popover" popovertargetaction="show">Show popover</button>
<button popovertarget="popover" popovertargetaction="hide">Hide popover</button>
```

Output

Note that if you don't specify the `popovertargetaction` attribute, its default value is `toggle`, which shows and hides the popover element when you click the button repeatedly.

Popover states

When you add the `popover` attribute to an HTML element, its state defaults to `auto`. It means that the following code snippets are equivalent and have the same effect:

```
<div id="popover" popover>This is a popover</div>
```

and

```
<div id="popover" popover="auto">This is a popover</div>
```

If a popover is in the auto state, it behaves as follows:

- Clicking outside the popover element will hide it. This is referred to as “light dismissed”.
- Pressing the `ESC` key will also hide the popover.
- Only one popover can be displayed at a time, showing a second popover will hide the first one.

An auto-state popover can be useful when showing a single popover at a time. For example, you may want to display a message indicating the result of an API call.

A popover has another state which is the `manual` state. The following sets the `manual` state for a popover element:

```
<div id="popover" popover="manual">This is a popover</div>
```

In this `manual` state:

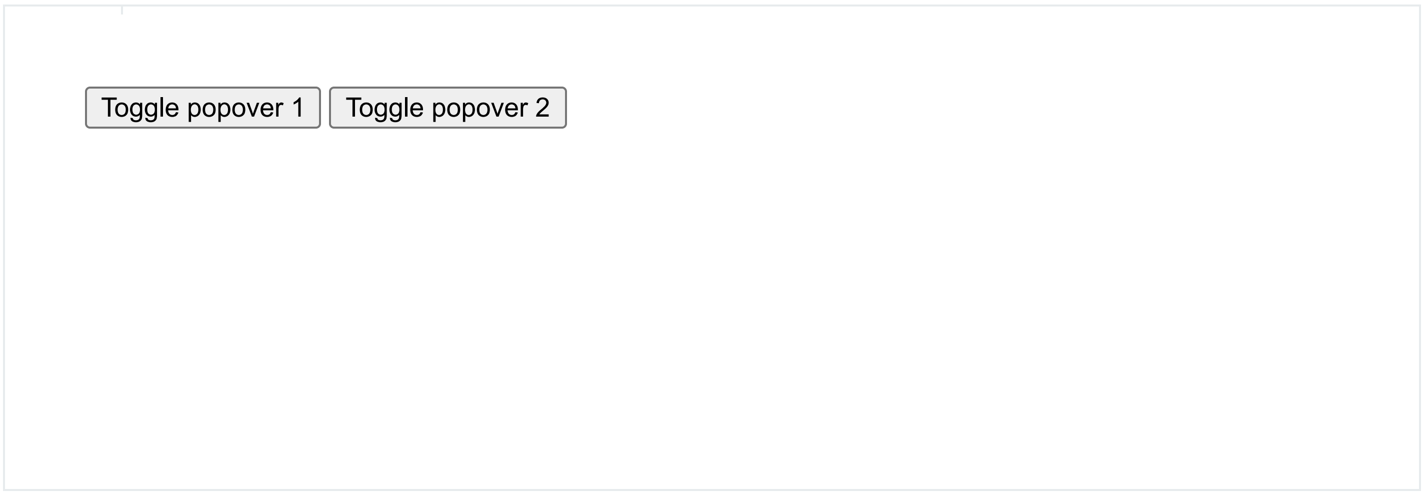
- The popover cannot be “light dismissed”.
- Multiple independent popovers can be displayed at a time.

The following page shows how to display two popovers at the same time:

```
<!DOCTYPE html>
<html lang="en">

    <head>
```

```html
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Popover</title>
        <style>
            :popover-open {
                position: absolute;
                inset: unset;
                top: 50px;
            }

            #popover1 {
                left: 10px;
            }

            #popover2 {
                left: 150px;
            }
        </style>
    </head>

    <body>

        <button type="button" popovertarget="popover1">Toggle popover 1</button>
        <button type="button" popovertarget="popover2">Toggle popover 2</button>

        <div id="popover1" popover="manual">This is popover 1</div>
        <div id="popover2" popover="manual">This is popover 2</div>


    </body>

</html>
```

Output



Controlling popovers via JavaScript

To check if the web browser supports the popover API, you can use the `popover` attribute of the `HtmlElement` as follows:

```
const isPopoverSupported = () => HTMLElement.prototype.hasOwnProperty("popover");
```

The `HtmlElement` object has the following method that controls the popover:

Popover method	Description
showPopover()	Show a popover
hidePopover()	Hide a popover
togglePopover()	Toggle a popover

The following page shows how to toggle a popover in JavaScript by pressing the `h` keyboard:

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>JavaScript Popover API example</title>
        <style>
            :popover-open {
                position: absolute;
                inset: unset;
                top: 5px;
                right: 5px;
            }

            @media all and (max-width: 450px) {
                :popover-open {
                    left: 5px;
                }
            }
        </style>
    </head>

    <body>
        <p>Press the letter <strong>h</strong> to toggle the popover.</p>
        <div id="message" popover>This is a popover</div>
        <script>
            const isPopoverSupported = () => HTMLElement.prototype.hasOwnProperty("popover")

            if (isPopoverSupported()) {
                document.addEventListener("keydown", (event) => {
                    if (event.key === "h") {
                        message.togglePopover();
                    }
                });
            }

        </script>

    </body>

</html>
```

Output

(Click inside this output area)

Press the letter **h** to toggle the popover.

How it works.

First, declare a popover element with the id `message` and attribute `popover` :

```
<div id="message" popover>This is a popover</div>
```

Second, define an arrow function that returns true if the Popover API is supported:

```
const isPopoverSupported = () => HTMLElement.prototype.hasOwnProperty("popover");
```

Third, check if the Popover API is supported:

```
if (isPopoverSupported()) {
```

Finally, add the event listener that toggles the popover when users press the `h` key:

```
document.addEventListener("keydown", (event) => {
       if (event.key === "h") {
               message.togglePopover();
       }
});
```

Popover CSS

The example page shows the popover on the top right corner of the screen. This is because of the CSS class that we use.

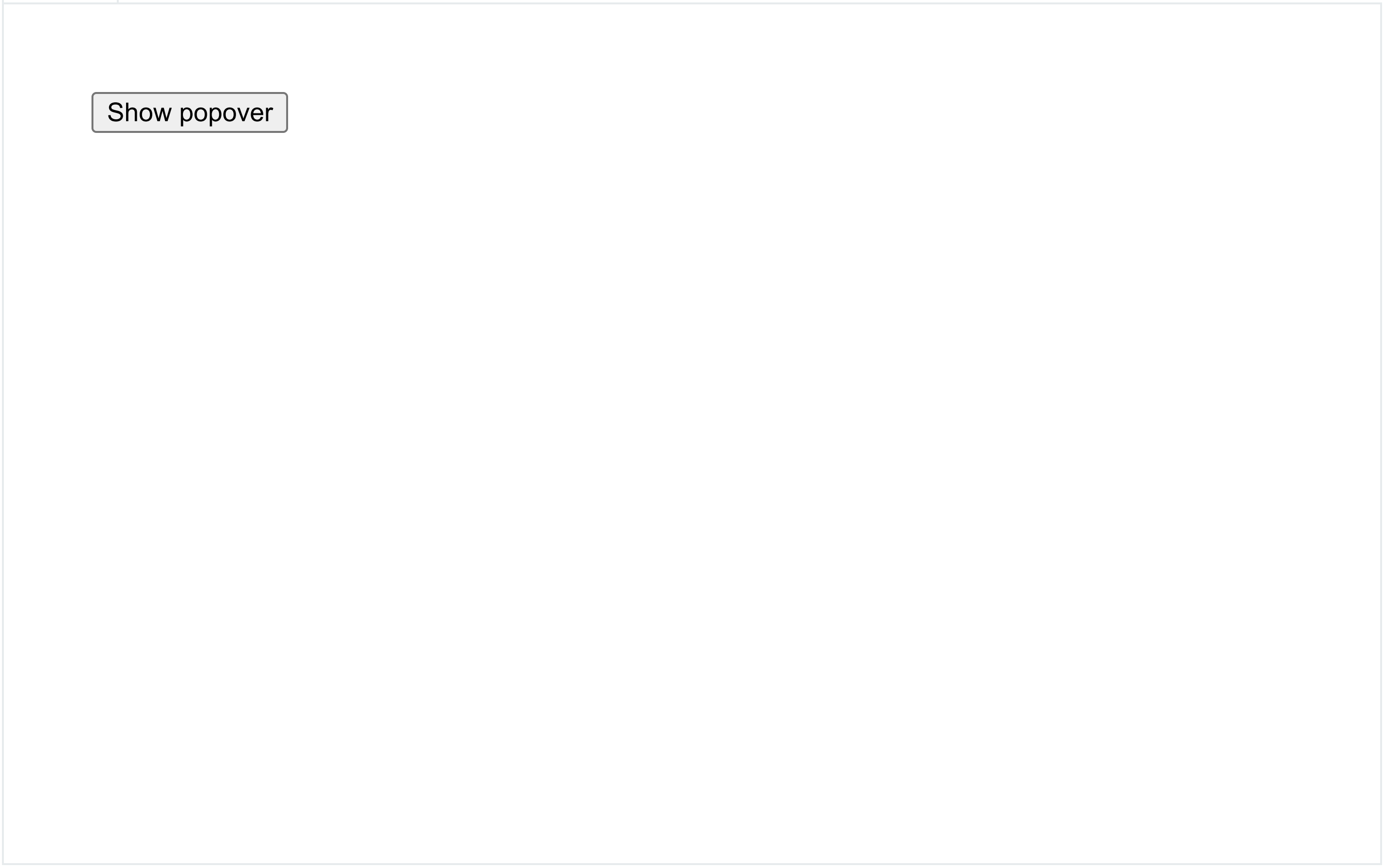
When a popover shows, it has a `:popover-open` pseudo-class, which you can use to style popover elements.

Additionally, you can use the `::backdrop` psuedo-element which is a full-screen element placed directly behind popover elements. This allows you to add effect to the page content behind the popover.

The following page illustrates how to make the background blur using the `::backdrop` pseudo-element:

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>JavaScript Popover API background example</title>
        <style>
            body {
                background-image: url("data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A//www.w3
                background-repeat: repeat;
                font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto
                line-height: 1.8;
            }

            button {
                margin: 5px 0 0 5px;
            }

            :popover-open {
                width: 300px;
                height: 200px;
                padding: 0 10px;
                border-radius: 10px;
            }

            ::backdrop {
                backdrop-filter: blur(3px);
            }
        </style>
    </head>

    <body>
        <button popovertarget="message" popovertargetaction="show">
            Show popover
        </button>
        <div id="message" popover>
            <h2>Popover heading</h2>
            <p>
                This is a popover with a blur page content.
            </p>
        </div>
    </body>

</html>
```

Output



Summary

- Add a `popover` attribute to an element to make it a popover.
- Use the `popovertargetaction` attribute to define actions such as `toggle` , `show` , and `hide` the target popover element.
- Use the `popovertarget` to specify the id of the popover element for the control elements such as buttons.
- Use the `showPopover()` , `hidePopover()` , and `togglePopover()` method to show, hide, and toggle the popover.