JavaScript onload Event

Summary: in this tutorial, you will learn how to handle the load event that fires on the document, image, and script elements in JavaScript.

The window's load event

For the `window` object, the `load` event is fired when the whole webpage (HTML) has fully loaded, including all resources such as JavaScript files, CSS files, and images.

To handle the `load` event, you register an event listener using the `addEventListener()` method:

```
window.addEventListener('load', (event) => {
    console.log('The page has fully loaded');
});
```

Alternatively, you can use the `onload` property of the `window` object:

```
window.onload = (event) => {
    console.log('The page has fully loaded');
};
```

If you maintain a legacy system, you may find that the `load` event handler is registered in the body element of the HTML document, like this:

```
<!DOCTYPE html>
<html>
<head>
    <title>JS load Event Demo</title>
</head>
<body onload="console.log('Loaded!')">
```

```
</body>
</html>
```

It's a good practice to use the `addEventListener()` method to assign the `onload` event handler whenever possible.

The image's load event

The `load` event also fires on images. To handle the `load` event on images, you use the `addEventListener()` method of the image elements.

The following example uses the `load` event handler to determine if an image, which exists in the DOM tree, has been completely loaded:

```
<!DOCTYPE html>
<html>
<head>
    <title>Image load Event Demo</title>
</head>
<body>
    <img id="logo">
    <script>
        let logo = document.querySelector('#logo');

        logo.addEventListener('load', (event) => {
            console.log('Logo has been loaded!');
        });

        logo.src = "logo.png";
    </script>
</body>
</html>
```

You can assign an `onload` event handler directly using the `onload` attribute of the `<img>` element, like this:

```
<img id="logo"
     src="logo.png"
     onload="console.log('Logo loaded!')">
```

If you create an image element dynamically, you can assign an `onload` event handler before setting the `src` property as follows:

```
window.addEventListener('load' () => {
    let logo = document.createElement('img');
    // assign and onload event handler
    logo.addEventListener('load', (event) => {
        console.log('The logo has been loaded');
    });
    // add logo to the document
    document.body.appendChild(logo);
    logo.src = 'logo.png';
});
```

How it works:

- First, create an image element after the document has been fully loaded by placing the code inside the event handler of the window's load event.
- Second, assign the `onload` event handler to the image.
- Third, add the image to the document.
- Finally, assign an image URL to the `src` attribute. The image will be downloaded to the element as soon as the `src` property is set.

The script's load event

The `<script>` element also supports the `load` event slightly different from standard ways. The script's `load` event allows you to check if a JavaScript file has been completely loaded.

Unlike images, the web browser starts downloading JavaScript files only after the `src` property has been assigned and the `<script>` element has been added to the document.

The following code loads the `app.js` file after the page has been completely loaded. It assigns an `onload` event handler to check if the `app.js` has been fully loaded.

```
window.addEventListener('load', checkJSLoaded)

function checkJSLoaded() {
    // create the script element
    let script = document.createElement('script');

    // assign an onload event handler
    script.addEventListener('load', (event) => {
        console.log('app.js file has been loaded');
    });

    // load the script file
    script.src = 'app.js';
    document.body.appendChild(script);
}
```

Summary

- The `load` event occurs when the document has been completely loaded, including dependent resources like JavaScript files, CSS files, and images.
- The `<img>` and `<script>` elements also support the `load` event.
- Use the `addEventListener()` method to register an `onload` event handler.