Difference between DOMContentLoaded and load Events

Last Updated : 04 Oct, 2024

These two events In web development, the **DOMContentLoaded** and **load events** are used to detect when parts of the webpage are ready. The DOMContentLoaded event fires when the HTML is fully parsed and the DOM is ready, while the load event waits until all resources like images, scripts, and stylesheets are completely loaded.

DOMContentLoaded

The DOMContentLoaded event fires when the HTML document is fully loaded and parsed, meaning the DOM is ready, but before external resources like images and stylesheets are fully loaded. It allows scripts to execute as soon as the DOM is available.

Syntax

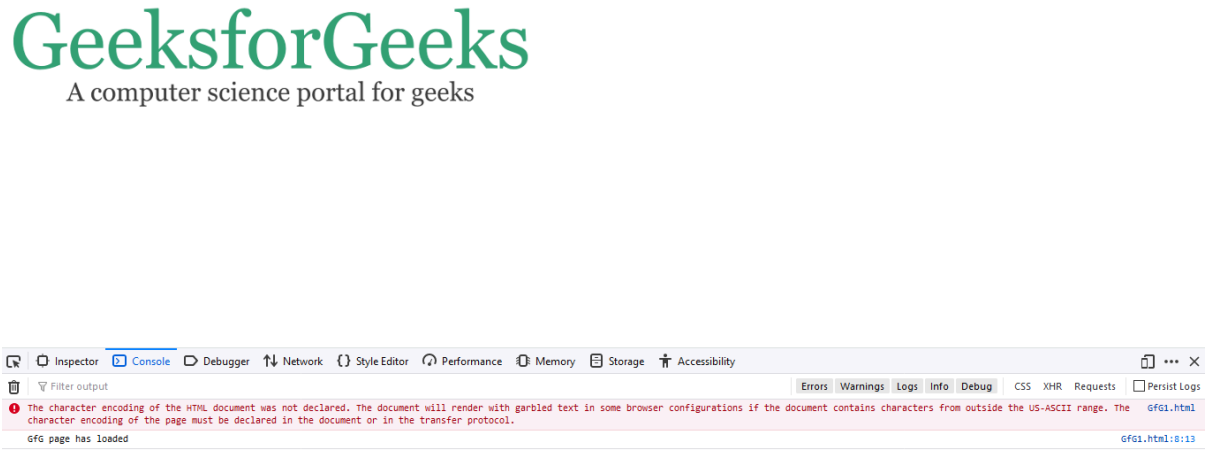
```
document.addEventListener("DOMContentLoaded", function(e) {
  console.log("GfG page has loaded");
});
```

Example : In this example the DOMContentLoaded event, logging a message when the page loads, and displays a centered image using inline CSS on an HTML page.

```
<!DOCTYPE html>
<html>

<head>
    <title>
        Output of DOMContentLoaded and Load events
    </title>

    <script type="text/javascript">
        document.addEventListener("DOMContentLoaded", function (e) {
            console.log("GfG page has loaded");
```

```
        });
    </script>
</head>

<body>
    <img src=
"https://media.geeksforgeeks.org/wp-
content/uploads/20190328185307/gfg28.png"
        style="align-content: center">
</body>

</html>
```

Output:



Advantages of using DOMContentLoaded event

- **Faster execution:** Scripts run as soon as the DOM is ready, without waiting for images or other resources to load.
- **Improved user experience:** Enables faster interaction with the page content.
- **Efficient DOM manipulation:** Ensures that scripts manipulate elements as soon as they are available.
- **Reduces delays:** Helps avoid unnecessary delays caused by waiting for non-critical resources like large images.

load Events

The load event fires when the entire webpage, including all dependent resources such as images, scripts, and stylesheets, is fully loaded. It ensures that everything is completely available before executing associated JavaScript functions or actions.

Syntax

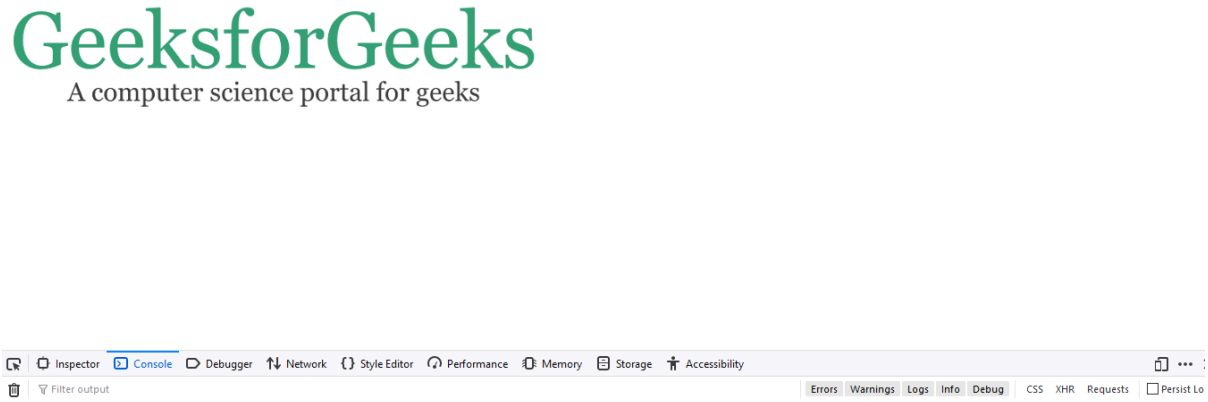
```
document.addEventListener("load", function(e) {
  console.log("The page has completely loaded.");
});
```

Example : In this example the load event, logging a message when the entire page, including images, is fully loaded, displaying a centered image on the HTML page.

```
<!DOCTYPE html>
<html>

<head>
    <title>
        Load events
    </title>

    <script type="text/javascript">
        document.addEventListener("load", function (e) {
            console.log("GfG page has loaded completely");
        });
    </script>
</head>

<body>
    <img src=
"https://media.geeksforgeeks.org/wp-
content/uploads/20190328185307/gfg28.png"
        style="align-content: center">
</body>

</html>
```

Output

:



Advantages of using load event

- **Ensures full content availability:** Executes scripts after all resources are completely loaded.

- **Prevents errors:** Reduces potential errors caused by missing elements or resources.
- **Supports media-heavy pages:** Ideal for pages with large images, videos, or external assets.
- **Improves script reliability:** Ensures that scripts can access and manipulate all fully loaded resources.

Difference table between DOMContentLoaded and load Events

Feature	DOMContentLoaded	load
Firing time	Fires when HTML is fully parsed and DOM is ready	Fires when the entire page, including resources, is fully loaded
External Resources	Doesn't wait for images, stylesheets, or frames	Waits for all external resources to fully load
Performance	Faster execution, improving page interactivity	Slower due to waiting for all resources to finish loading
Use Case	Suitable for early DOM manipulation	Best for handling resource-dependent operations
Common Usage	Interactivity setup, DOM-related scripts	Analytics, loading animations, media handling