JavaScript - addEventListener()

The JavaScript **addEventListener()** method is used to attach an event handler function to an HTML element. This allows you to specify a function that will be executed when a particular event occurs on the specified element.

Events are a specific occurrence or action like user clicks, keypress or page loads. The browser detects these events and triggers associated JavaScript functions known as event handlers to respond accordingly.

Developers employ the 'addEventListener()' method for linking particular HTML elements with specific function behaviours when those events occur. Examples of events include clicks, mouse movements, keyboard inputs, and document loading.

Syntax

The basic syntax for addEventListener() is as follows −

```
element.addEventListener(event, function, options);
```

Here **element** is an HTML element, such as a button, input or div - can be selected using methods like getElementById, getElementsByClassName, getElementsByTagName and querySelector; these are just a few examples. The event listener attaches to this particular element.

Parameters

The addEventListener() method accepts the following parameters −

event − a string that embodies the type of action − for instance, "click", "mouseover", or "keydown" among others; will serve as our trigger to execute the given function.

function − a named, anonymous or reference to an existing function is called when the specified event occurs; it's essentially the operation that facilitates execution at predetermined instances.

options (optional) − it allows for the specification of additional settings particularly capturing or once behaviours related to the event listener.

Examples

Example: Alert on Clicking Button

In this example, we will have a simple button being displayed which upon clicking shows an alert on the screen. The addeventlistener will be responsible for handling the event “click” which means it will call a function handleClick which throws an alert to the screen when the button is clicked. We make use of the getElementById to fetch the button we want to bind the event listener to.

This is a commonly used event when it comes to submitting on forms, login, signup etc.

Open Compiler

```
<html>
<head>
   <title>Click Event Example</title>
</head>
<body>
   <p> Click the button below to perform an event </p>
   <button id="myButton">Click Me</button>

   <script>
      // Get the button element
      const button = document.getElementById("myButton");
      // Define the event handler function
      function handleClick() {
         alert("Button clicked!");
      }
      // Attach the event listener to the button
      button.addEventListener("click", handleClick);
   </script>

</body>
</html>
```

Example: Colour Change on Mouse Over

In this example we have a dig tag which will initially be of light blue colour. Upon hovering the mouse on this div tag, it will change to red and back to blue if we hover out.

There are two events in this case, **mouseover** and **mouseout**. The mouseover means the mouse moves onto an element mouseout means the mouse movies out of an

element.

There are two functions here, one for mouseover and one for mouseout. Upon mouseover, the background colour property is set to light coral (a shade of red) and upon mouseout, the background colour is set to light blue.

These types of mouse hover events are commonly seen when hovering over the navbar of a lot of websites.

Open Compiler

```
<html>
<head>
   <title>Mouseover Event Example</title>
   <style>
      #myDiv {
         width: 600px;
         height: 200px;
         background-color: lightblue;
      }
   </style>
</head>
<body>
   <div id="myDiv">Hover over me</div>
   <script>
      // Get the div element
      const myDiv = document.getElementById("myDiv");

      // Define the event handler function
      function handleMouseover() {
      myDiv.style.backgroundColor = "lightcoral";
      }

      // Attach the event listener to the div
      myDiv.addEventListener("mouseover", handleMouseover);

      // Additional example: Change color back on mouseout
      function handleMouseout() {
         myDiv.style.backgroundColor = "lightblue";
      }

      myDiv.addEventListener("mouseout", handleMouseout);
   </script>
```

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

There can be multiple event listeners for the same elements like in the case of 2nd example which has two event listeners (for mouseover and mouseout). Event listeners can be removed using the removeEventListener function. By passing a parameter in the options as once:true, we can ensure that the event listener is removed after being invoked once and this is important in certain case scenarios like payments.

It is important to note that one should never use the "on" prefix for specifying events, this simply means for a click event, we should specify it as "click" and not "onclick".