Persistent

JavaScript: Events

Persistent Interactive | Persistent University

Key learning points :

- JavaScript Events & Event handlers
- Different event handling approaches along with Pros & Cons
- Event propagation – event capturing & event bubbling
- How to capture more details of an Event
- Event Object, its properties along with examples

Events & Event Handlers

- Event is an action that occurs when a user interacts with a web page.

- Every element on a web page has certain events which can trigger invocation of event handlers.

- Examples of user events
  - A mouse click
  - Moving mouse over a link
  - Selecting an item from combo box
  - Typing into a textbox

- Event handlers will contain the functionality that must occur when the respective event is triggered.

Events in JavaScript

Event	Description
onclick	Mouse clicks an object
onfocus	An element gets focus
onblur	An element loses focus
onchange	The content of a field changes
onkeydown	A keyboard key is pressed
onkeypress	A keyboard key is pressed or held down
onkeyup	A keyboard key is released

Events in JavaScript

Event	Description
onmousemove	When mouse moves over an element
onmouseover	When mouse is on the element
onmouseout	When mouse is moved out of the element
onsubmit	The submit button is clicked
onload	When an element is loaded
onunload	When an element is unloaded

Event Handling Approaches

- Inline Event Handlers
- Binding events from script
- Binding using addEventListener API

Inline Event Handler

```
<script type="text/JavaScript">

        function display(){
            alert("Welcome to PSL");
        }

</script>

<body>

 <input type="button" text="click me"
onclick="display()"      />

</body>
```

Setting the name of handler to be invoked on click.

Pros and Cons

- Most easiest and oldest approach
- Not recommended approach as embedding JavaScript code into HTML.
- Event handler is hardcoded in tag which is not changeable dynamically.

Binding from Script

Event binding to be done on window.onload

```
<script type="text/JavaScript">
        window.onload = function() {
        document.getElementById('btn1').onclick = display;
        }
    function display(){
        alert("Welcome to PSL");
    }
</script>
<body>
   <input type="button" size id=" btn1 "  text="click me" />
  </body>
```

Removing a handler

Assign the handler to null.

```
<head>
      <script type=“text/JavaScript”>
                  function removeHandler()  {

document.getElementById(‘btn1').onclick = null;
                  }
      </script>
</head>
<body>
       <input type="button" size id=" btn1 "  text="click me"
/>
 </body>
```

Pros and Cons

- Better approach over first
- Cannot add more than one handler

Here we attach, 2 handlers for same button. But when button is clicked, ONLY greet will be called.

```
window.onload = function() {

    document.getElementById('btn1').onclick = display;

    document.getElementById('btn1').onclick = greet;
}

<input type="button" size id=" btn1 "  text="click me" />
```

addEventListener(eventType, eventHandler, true\false)

When button is clicked, display() will be called

```
<body>
    <script type="text/JavaScript">

        window.onload = function(){

document.getElementById('btn1').addEventListener(

            'click', display, false);
        }
    </script>

    <input type="button" size id=" btn1 "  text="click me" />
  </body>
```

Multiple Handlers

```
<body>
    <script type="text/JavaScript">
        window.onload = function(){
    document.getElementById('btn1').addEventListener(
                'click', display, false);
        }
    </script>
    <input type="button" size id=" btn1 "
        text="click me" />
</body>
```

removeEventListener(eventType, eventHandler, true \false)

```
<body>
        <script type=“text/JavaScript”>
         window.onload = function(){

        document.getElementById(‘btn1').removeEventListener('click', display, false);

         }
        </script>

<input type="button" size id=" btn1 "  text="click me" />
</body>
```

Pros and Cons

- Dynamically bind Handlers from script
- Can add multiple event handlers for same event
- Order of execution of multiple handlers not guaranteed
- Problem of Cross browser support

Capturing details of an Event

- User actions are captured by JavaScript events for example when a button is clicked, the event is ‘onclick’

- More details required such as :-
  - Which HTML element was clicked ?
  - Which mouse button was pressed during the click event ?
  - What was the mouse position during the event ?

Event Object

Event Property	Description
event.type	What type of event occurred: onclick,onchange,onmouseover
event.target	Reference of current element on which user has performed action. Eg: button, image, div, combo box
event.keyCode	KeyCode of the key pressed
event.which	Which mouse button was clicked. Eg:Left or Right
event.clientX/clientY	Coordinates on screen where mouse cursor is pointing to.

Example to use event object

Event Object has to be passed

```
<body>
        <script type="text/JavaScript">

        document.getElementById('btn1').onclick =
    setValue;

        function setValue(event) {

        var target  =  event.target;
        var id = target.id;

        }
        </script>

        <button id="btn1">Submit</button>
</body>
```

IE support to be handled

Used when browser is IE

```
function setValue(event) {
    var target = event.target ? event.target :
event.srcElement;
    var id = target.id;
 }
```

Summary : Session

With this we have come to an end of our session, where we discussed :

- JavaScript Events & Event handlers
- Different Event handling approaches, their pros & cons.
- Event object

At the end of this session, we expect you to :

- Understand concepts related to entire event handling
- mechanism.
- Apply & implement all these concepts whenever required

Appendix

- References
- Key Contacts

Reference Material : Books

- **Head First JavaScript Programming**
  - By: Eric T. Freeman; Elisabeth Robson
  - Publisher: O'Reilly Media, Inc.
- **Professional: JavaScript® for Web Developers**
  - By: Nicholas C. Zakas
  - Publisher: Wrox

Persistent University

Tarun Kr. Joshi

tarun_joshi@persistent.co.in

Priya Singh

priya_singh@persistent.co.in

Shubhangi Kelkar

shubhangi_kelkar@persistent.co.in

Persistent

Thank you!

Persistent Interactive | Persistent University