ACADGILD

Presents
Front End Web Development Basics

Session 7 – JavaScript

Agenda – JavaScript

1. Introduction to DOM
2. DOM Manipulation
3. Event
4. Event Types
5. Event Bubbling or Event Capturing
6. Action Dialog
7. Form validation

DOM (Document Object Model)

- A standard platform- and language-neutral programming interface **for building, accessing, and manipulating valid HTML and well-formed XML documents.**

- Ultimate goal is to make it possible for programmers to write applications that **work properly on all browsers and servers, and on all platforms.**

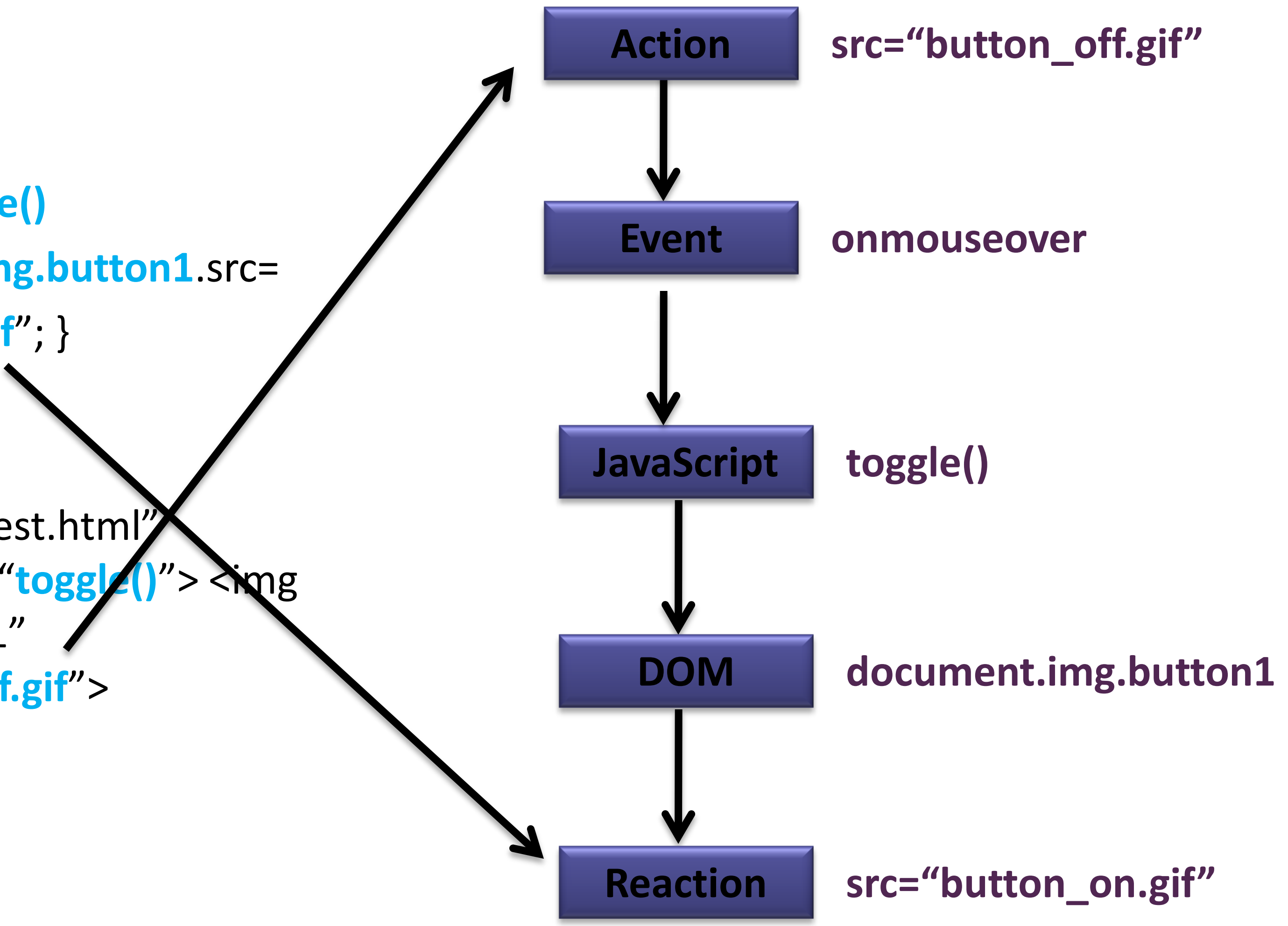
- When a web page is loaded, Browser creates a **D**ocument **O**bject **M**odel of the page

DOM (Document Object Model)

- With Object model, JavaScript gets power it need to create dynamic HTML
  - JavaScript can change the HTML elements in the page
  - JavaScript can change the HTML attributes in the page
  - JavaScript can change the CSS styles in the page
  - JavaScript can remove existing HTML elements and attributes
  - JavaScript can add new HTML elements and attributes
  - JavaScript can react to all existing HTML events in the page
  - JavaScript can create new HTML events in the page

How DOM Works ?

```
<head>
            <script>
      function toggle()
      { document.img.button1.src=
      "button_on.gif"; }
      </script>
 </head>
<body>  <a href="test.html"
     onmouseover="toggle()"> <img
     name="button1"
     src="button_off.gif">
</a>  </body>
```



DOM Manipulation

- **document.getElementsByTagName(tagname)**

  This method returns a collection of all elements reference in the document with the specified tag name.

- **document.getElementsByClassName(classname)**

  This method returns a collection of all elements reference in the document with the specified class name.

- **document.getElementById(id)**

  This method returns a element reference in the document with the specified id.

Modifying HTML Using innerHTML

InnerHTML is the property of DOM object nodes. Using this property we can get/set the html inside a tag.

Example :

```
<head> <script type = "text/javascript">
        function addHeading(){
        var ref = document.getElementById("container");
        var htmlToInsert= '<h3> This is the Heading</h3>';
        ref.innerHTML = htmlToInsert;
}  </script>
</head>
<body>   <button  onclick = "addHeading()">Add Heading</button>
                <div id="container"></div>    </body>
```

Events

- JavaScript can also respond to events which can also be actions by the user.

- Example clicking on a element, hovering over an element are all actions by user and JavaScript uses events which can react to these actions.

- JavaScript attaches a function called an event listener or event handler to a specific event and the function invokes when the event occurs.

Events can be attached in the following ways

1)Inline HTML attributes

2)Adding to element properties with JavaScript

3)Using DOM Event Listeners

Inline HTML Elements

- Events can be attached as attributes to the elements like this

```
<div onclick = “showMsg()”>Click<div>
```

Adding to Element Properties

We can also assign a function to the onclick property of a DOM node element. Have a look at the code snippet below

```
<div id = “container”>click here</div>
<script type = “text/javascript”>var ref =
    document.getElementById('container');
ref.onclick = function () {
alert('The div area is clicked');
};
</script>
```

Using DOM Event Listener

- The best way to handle events is to use the event listener approach.We can assign listeners to the click event using the addEventListener() method.
  - ref.addEventListener(event,function)

- addEventListener() method attaches an event handler to the specified element.
- You can add event listeners to any DOM object

- **The removeEventListener() method**
- removes event handlers that have been attached with the addEventListener() method
- ***Syntax :*** *element*.removeEventListener(“Event Name", function);

Event Types

- **Mouse Events** - mouseup , mousedown
- **Keyboard events** – keydown , keyup
- **window events** -load, unload
- **Form events** – focus ,change

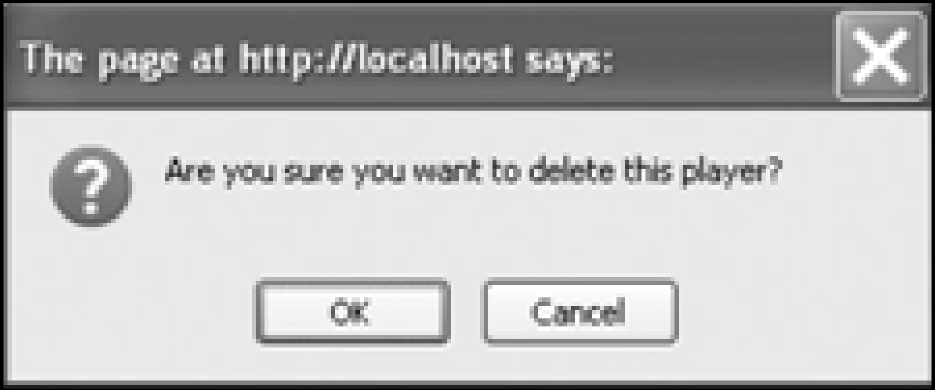
Event Bubbling or Event Capturing

- **Event Propagation :** way of defining the element order when an event occurs
- Two ways of event propagation in the HTML DOM
  - Bubbling and Capturing

- **Event Bubbling:** inner most element's event is handled first and then the outer
- **Event Capturing:** outer most element's event is handled first and then the inner

Action Dialog

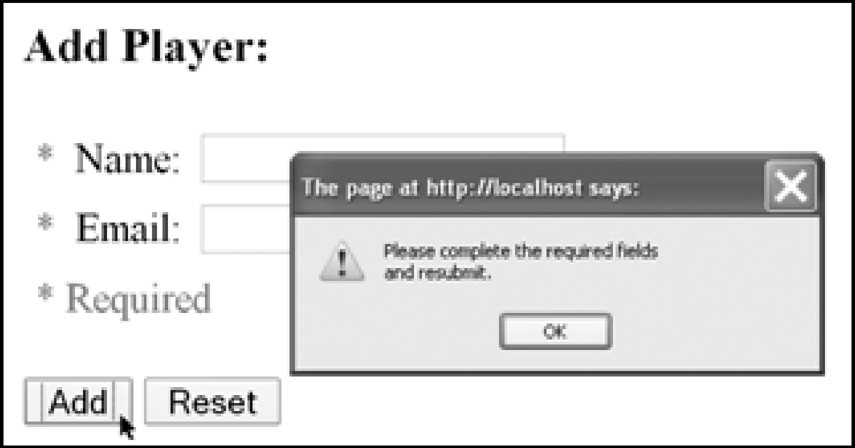
```
<script type="text/javascript">

        function confirmDelete() {
          var answer = confirm("Are you sure you want"
            + "to delete this player?");
          return answer
        }

</script>

<form method="post" action="/delete">
                <p>
                <input type="submit" value="Delete“  onclick="return confirmDelete()" />
                </p>
</form>
```


The page at http://localhost says:

Are you sure you want to delete this player?

Form Validation

```
<script>
 function validate() {
  if (document.getElementById("name").value.length == 0) {
    alert("Please complete the required fields\n" +
      "and resubmit.");
    return false;
  }
  return true;
 }
</script>
```



Add Player:

* Name:

* Email:

* Required

The page at http://localhost says:

Please complete the required fields and resubmit.

```
<h3>Add Player:</h3>
<form id="form1" action="addplayer“ onsubmit="return  validate()" >
 <p>Name: <input type="text" id="name" /></p>
  ...
 <p><input type="submit" value="Register" /></p>
</form>
```

Lets Discuss Assignments