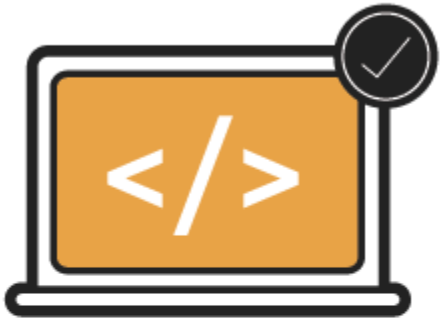
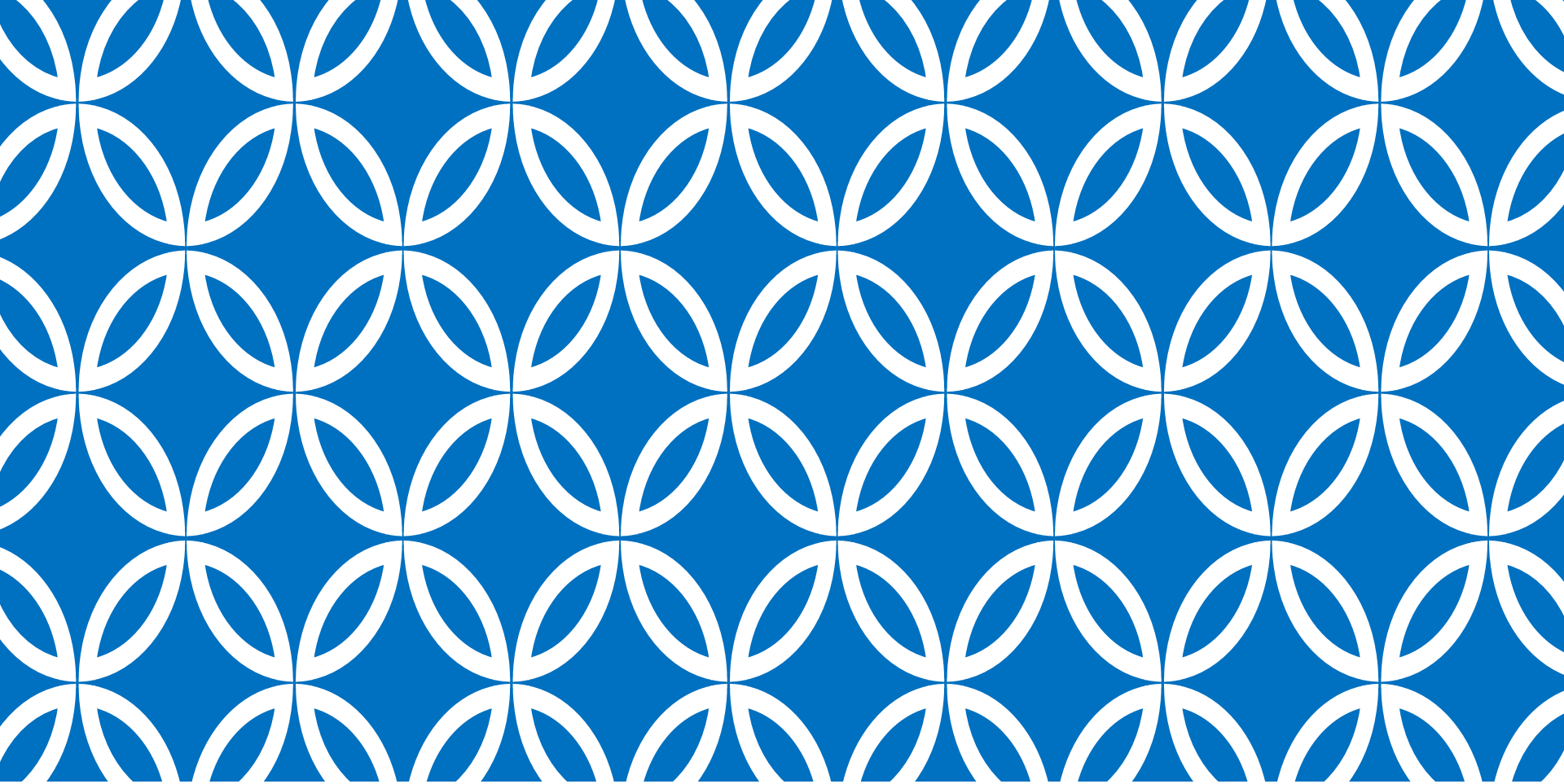
ACADGILD



FRONT END WEB DEVELOPMENT FUNDAMENTALS

Website : http://www.acadgild.com
LinkedIn : https://www.linkedin.com/company/acadgild
Facebook : https://www.facebook.com/acadgild



Session 7 – JavaScript

Agenda – JavaScript

1. DOM
2. How DOM works?
3. DOM Manipulation
4. Modifying HTML Using innerHTML
5. Events
6. Event Types
7. Event Bubbling and Event Capturing
8. Action Dialog
9. Form Validation

DOM (Document Object Model)

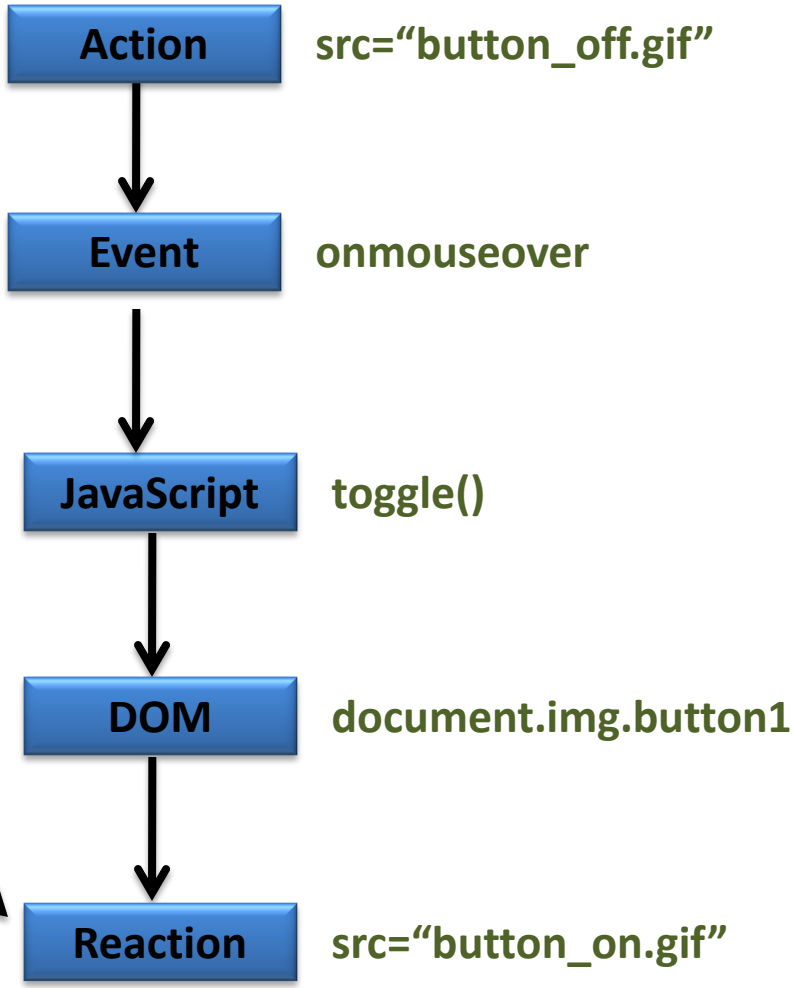
- **DOM** is cross-platform and language-independent programming interface for building, accessing, and manipulating valid HTML and well-formed XML documents.
- Ultimate goal is to make it possible for programmers to write applications that work properly on all browsers and servers, and on all platforms.
- When a web page is loaded, Browser creates a **D**ocument **O**bject **M**odel of the page.

DOM (contd.)

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

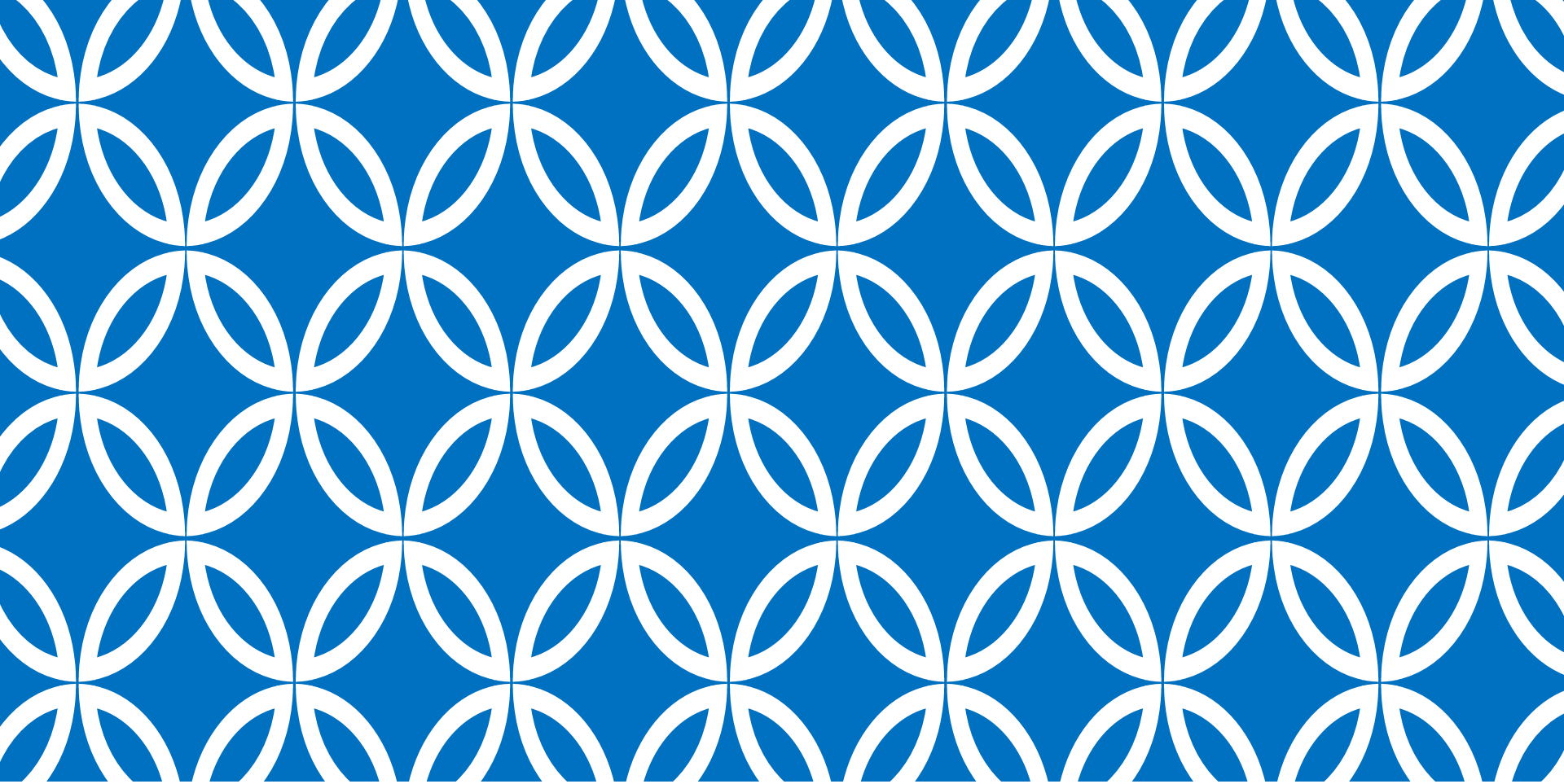
- **document.getElementsByTagName(tagname)** - This method returns a collection of all elements reference in the document with the specified tag name.
- **document.getElementsByClassName(classname)** - This method returns a collection of all elements reference in the document with the specified class 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

Events

- **JavaScript** can respond to events which can also be actions by the user.
  - **Example**, clicking on an 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 as shown below:

  Click

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, mouseover, mousemove, etc.
- **Keyboard Events** – keydown, keypress and keyup
- **Window Events** – load, unload
- **Form events** – focus, change

Event Bubbling and Event Capturing

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

Action Dialog

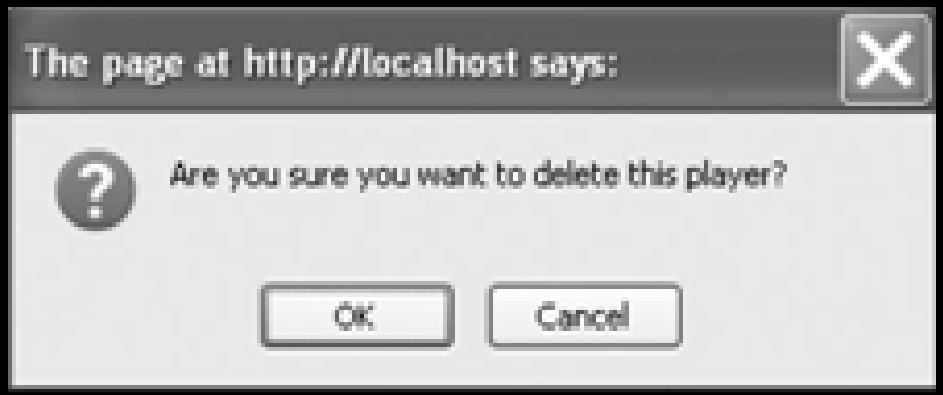
```
<script type="text/javascript">

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




</script>

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



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>
```


```
<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

Assignment