OBJECT™ TECHNOLOGIES

Dynamic HTML

What will be covered

OBJECT TECHNOLOGIES

- Events
- Event handlers
- HTML event handler attributes
- Disadvantages of event handler attributes
- Assigning Event handlers
- Generating dynamic HTML
- JS popup boxes

Events

OBJECT

- Events are actions or occurrences that happen in the system you are programming, which the system tells you about so you can respond to them in some way if desired.
- In the case of the Web, events are fired inside the browser window, and tend to be attached to a specific item that resides in it — this might be a single element, set of elements, the HTML document loaded in the current tab, or the entire browser window. There are many different types of events that can occur.
  - The user selects a certain element or hovers the cursor over a certain element.
  - The user chooses a key on the keyboard.
  - The user resizes or closes the browser window.
  - A web page finishes loading.
  - A form is submitted.
  - A video is played, paused, or finishes.

Event Handlers

OBJECT TECHNOLOGIES

- Each available event has an event handler, which is a block of code (usually a JavaScript function that you as a programmer create) that runs when the event fires.
- When such a block of code is defined to run in response to an event, is called registering an event handler.
- Mouse events :

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Event Handlers

OBJECT TECHNOLOGIES

- Keyboard Events :

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

- Form Events :

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

Event Handlers

OBJECT TECHNOLOGIES

- Keyboard Events :

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

- Form Events :

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

Event Handlers

OBJECT TECHNOLOGIES

- Window events :

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

- All these event handlers are actually used as attributes to a respective element.
- For this attribute, value is given as in the form of java script code

HTML event handler attributes

OBJECT TECHNOLOGIES

- Event handlers typically have names that begin with on, for example, the event handler for the click event is onclick.
- To assign an event handler to an event associated with an HTML element, you can use an HTML attribute with the name of the event handler. For example, to execute some code when a button is clicked, you use the following:

```
<input type="button" value="Save" onclick="alert('Clicked!')">
```

- When you assign JavaScript code as the value of the onclick attribute, you need to escape the HTML characters such as ampersand (&), double quotes ("), less than (<), etc., or generates syntax error.

HTML event handler attributes

OBJECT

```
<script>
    function showAlert() {
        alert('Clicked!');
    }
</script>
<input type="button" value="Save" onclick="showAlert()">
```

- In this example, the button calls the showAlert() function when it is clicked.
- The showAlert() is a function defined in a separate <script> element, and could be placed in an external JavaScript file.

HTML event handler attributes

OBJECT TECHNOLOGIES

- The code in the event handler can access the event object without explicitly defining it.

```
<input type="button" value="Save" onclick="alert(event.type)">
```

- The **this** value inside the event handler is equivalent to the event's target element:

```
<input type="button" value="Save" onclick="alert(this.value)">
```

- the event handler can access the element's properties,

```
<input type="button" value="Save" onclick="alert(value)">
```

Disadvantage of event handler attributes

OBJECT TECHNOLOGIES

- the event handler code is mixed with the HTML code, which will make the code more difficult to maintain and extend.
- If the element is loaded fully before the JavaScript code, users can start interacting with the element on the webpage which will cause an error. It may happen that javascript code is not loaded by that time.

Assigning Event handlers

OBJECT TECHNOLOGIES

- Each element has event handler properties such as onclick. To assign an event handler, you set the property to a function

```
let btn = document.querySelector('#btn');

btn.onclick = function() {
    alert('Clicked!');
};
```

- To remove the event handler, you set the value of the event handler property to null
- The addEventListener() method accepts three arguments: an event name, an event handler function, and a Boolean value that instructs the method to call the event handler during the capture phase (true) or during the bubble phase (false).

Assigning Event handlers

OBJECT

```
let btn = document.querySelector('#btn');
btn.addEventListener('click',function(event) {
    alert(event.type); // click
});

btn.addEventListener('click',function(event) {
    alert('Clicked!');
});
```

- The removeEventListener() removes an event listener that was added.

Generating dynamic content

OBJECT TECHNOLOGIES

```
</head>
<body bgcolor="pink">
    <form>
        <input type="text" id="num" />
        <input type="button" onclick="findFactoial()" value="Click me" />
    </form>
    <p id="result"></p>
</body>
</html>
```

HTML page

JS Code In the form of function

```
<script>
    function findFactoial(){
        var num = document.getElementById("num").value;
        var i,fact;
        fact = 1;
        for (i = 1; i <= num; i++)
        {
            fact = fact*i;
        }
        alert("Factorial : "+fact);
        document.getElementById("result").innerHTML = "Factorial is : "+fact;
    }

</script>
```

Generating dynamic content

OBJECT

- In the above example, HTML form is used to accept the input from the user.
- After inserting the number when user clicks on the button, JS function is called.
- It calculates factorial and displays in the element of HTML page having id as result. Function used is :

  document.getElementById("result")

- innerHTML is a property of HTML element node

JS Popup boxes

OBJECT TECHNOLOGIES

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

- **Alert Box**

  An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed.

- **Confirm Box**

  A confirm box is often used if you want the user to verify or accept something.

  When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

- **Prompt Box**

  A prompt box is often used if you want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

JS Popup boxes

OBJECT

Examples :

```
alert("I am an alert box!");
```

```
var r = confirm("Press a button");
if (r == true) {
    x = "You pressed OK!";
} else {
    x = "You pressed Cancel!";
}
```

```
var person = prompt("Please enter your name", "Harry Potter");
if (person != null) {
    document.getElementById("demo").innerHTML =
    "Hello " + person + "! How are you today?";
}
```

Dynamic HTML

Java script is mainly used for generating dynamic HTML based on user's input. Generating new content or seletively displaying some part of existing content based on some condition can be easily done by event handling in java script. In this chapter, we will learn about this dynamic HTML.

Events

An event is an action that occurs as per the user's instruction as input . Different types of events get generated when user performs some action like mouse clicks, button presses, press tab and text box change.

Here's a list of the most useful DOM events, just to take a look at:

Mouse events:

click – when the mouse clicks on an element (touchscreen devices generate it on a tap).

contextmenu – when the mouse right-clicks on an element.

mouseover / mouseout – when the mouse cursor comes over / leaves an element.

mousedown / mouseup – when the mouse button is pressed / released over an element.

mousemove – when the mouse is moved.

Keyboard events:

keydown and keyup – when a keyboard key is pressed and released.

Form element events:

submit – when the visitor submits a <form>.

focus – when the visitor focuses on an element, e.g. on an <input>.

Event handlers

JavaScript implements a component called an event handler that helps you acknowledge the certain action to the events. An event handler is a section of code that can be considered as a user-defined JavaScript function that operates when a particular event fires. We can define it as the registration of an event handler and can consider it as an event listener that performs and listens as an event and returns the result.

Following is the list of JavaScript listeners' definitions which have specific types of methods.

Onload: When your page loads, it performs accordingly.

Onclick: When a user clicks on a button or inputs it occurs.

Onmouseover: When a user mouses over on the button.

Onload: When your page loads, it performs accordingly.

Onclick: When a user clicks on a button or inputs it occurs.

Onmouseover: When a user mouses over on the button.

Onfocus: Certain scenarios when a user keeps the cursor in a form field.

Onblur: If a particular form field leaves within it.

Event handlers typically have names that begin with on, for example, the event handler for the click event is onclick. To assign an event handler to an event associated with an HTML element, you can use an HTML attribute with the name of the event handler. For example, to execute some code when a button is clicked, you use the following:

```
<input type="button" value="Save" onclick="alert('Clicked!')">
```

When you assign JavaScript code as the value of the onclick attribute, you need to escape the HTML characters such as ampersand (&), double quotes ("), less than (<), etc., or generates syntax error.

JS Popup Boxes

JavaScript provides different built-in functions to display popup messages for different purposes e.g. to display a simple message or display a message and take user's confirmation on it or display a popup to take a user's input value.

Alert Box

Use alert() function to display a popup message to the user. Examples :

```
alert("This is alert box!");  // display string message
alert(100); // display number
alert(true); // display boolean
```

Alert popup displays message and ok button. This ok button can be clicked to close the alert box.

The alert function can display message of any data type e.g. string, number, boolean etc. There is no need to convert a message to string type.

Confirm Box

Sometimes you need to take the user's confirmation to proceed. For example, you want to take user's confirmation before saving updated data or deleting existing data. In this scenario, use JavaScript built-in function confirm(). The confirm() function displays a popup message to the user with two buttons, OK and Cancel. You can check which button the user has clicked and proceed accordingly.

Confirm Box

Sometimes you need to take the user's confirmation to proceed. For example, you want to take user's confirmation before saving updated data or deleting existing data. In this scenario, use JavaScript built-in function confirm(). The confirm() function displays a popup message to the user with two buttons, OK and Cancel. You can check which button the user has clicked and proceed accordingly.

The following example demonstrates how to display a confirm box and then checks which button the user has clicked. confirm() will return true if user clicks ok button and false if user clicks cancel button

Example: Confirm Box

```
if (confirm("Do you want to save changes?") == true) {
   alert("Data saved successfully!");
} else {
   alert("Save Cancelled!");
}
```

Prompt Box

Sometimes you may need to take the user's input to do further actions in a web page. For example, you want to calculate EMI based on users' preferred tenure of loan. For this kind of scenario, use JavaScript built-in function prompt().

Prompt function takes two string parameters. First parameter is the message to be displayed and second parameter is the default value which will be in input text when the message is displayed.

Syntax:

```
prompt([string message], [string defaultValue]);
```

Example: prompt Box

```
var tenure = prompt("Please enter preferred tenure in years", "15");
if (tenure != null) {
   alert("You have entered " + tenure + " years" );
}
```

As you can see in the above example, we have specified a message as first parameter and default value "15" as second parameter. The prompt function returns a user entered value. If user has not entered anything then it returns null. So it is recommended to check null before proceeding.

```
var tenure = prompt("Please enter preferred tenure in years", "15");
if (tenure != null) {
   alert("You have entered " + tenure + " years" );
}
```

As you can see in the above example, we have specified a message as first parameter and default value "15" as second parameter. The prompt function returns a user entered value. If user has not entered anything then it returns null. So it is recommended to check null before proceeding.

Note:

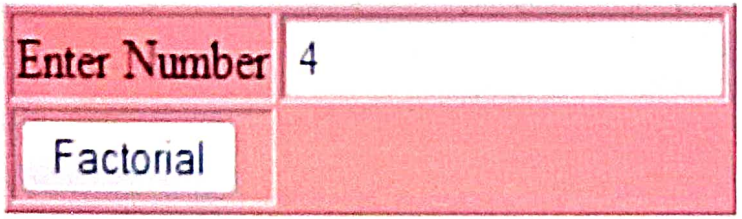
The alert, confirm and prompt functions are global functions. So it can be called using window object like window.alert(), window.confirm() and window.prompt().

Points to Remember :

1. Popup message can be shown using global functions - alert(), confirm() and prompt().
2. alert() function displays popup message with 'Ok' button.
3. confirm() function display popup message with 'Ok' and 'Cancel' buttons. Use confirm() function to take user's confirmation to proceed.
4. prompt() function enables you to take user's input with 'Ok' and 'Cancel' buttons. prompt() function returns value entered by the user. It returns null if the user does not provide any input value.

Assignments

1. Write a simple Html page "fact.html" with following features:
   It should contain a form to accept a number from user. Design the form as per following format. On clicking the Factorial Button, factorial of the number should be displayed in alert window as well as on same page.
   Expected Output:

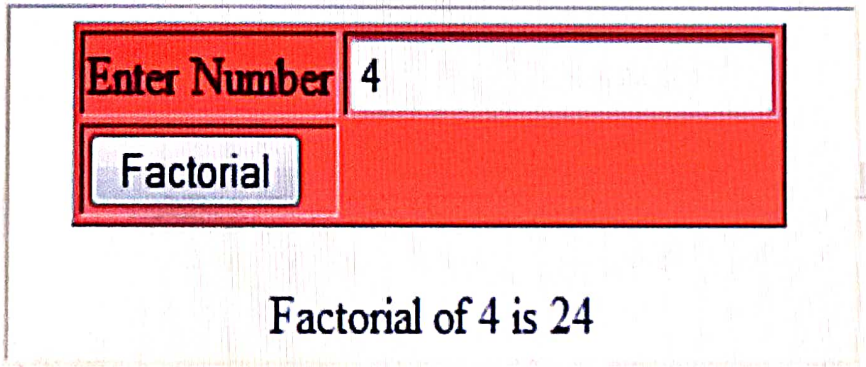


Assignments

1. Write a simple Html page "fact.html" with following features:

   It should contain a form to accept a number from user. Design the form as per following format. On clicking the Factorial Button, factorial of the number should be displayed in alert window as well as on same page.

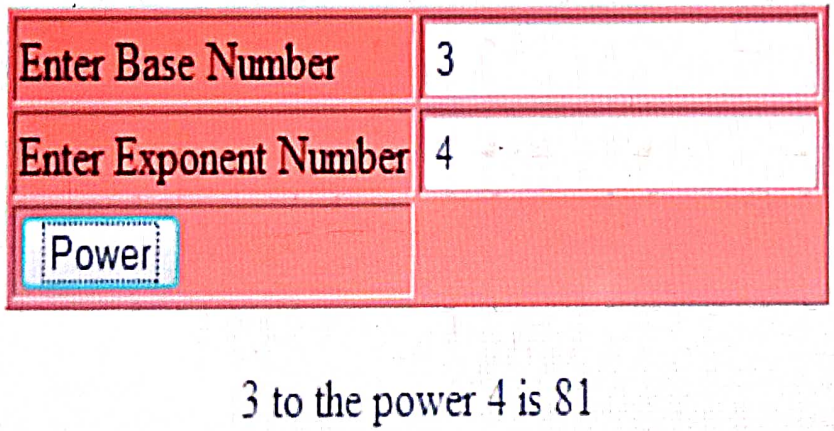
   Expected Output:



2. Write a simple Html page "power.html" with following features:

   It should contain a form to accept two numbers from user one for base and other for exponent. Design the form as per following format. On clicking the Power Button, Result should be displayed in alert window as well as on same page.

   Expected Output:



3. Create an HTML page factTable.html:

   It should have a button "Factorial Table". When we click on the button, the following output should appear on same page:

Number	Factorial

It should contain a form to accept two numbers from user one for base and other for exponent. Design the form as per following format. On clicking the Power Button, Result should be displayed in alert window as well as on same page.

Expected Output:

Enter Base Number	3
Enter Exponent Number	4
Power	

3 to the power 4 is 81

3. Create an HTML page factTable.html:

It should have a button "Factorial Table". When we click on the button, the following output should appear on same page:

Factorial Table

Number	Factorial
1	1
2	2
3	6
4	24

4. Accept a number from the user using HTML form and check whether number is pallindrome and display the message accordingly on the web page
5. Accept the number from the user and check whether number is prime and display the message accordingly on the web page. Give proper styling to the web page
6. Accept the number from the user and display the sum of digits of a number on the web page.