UNIT 4 – DIALOG BOX AND EVENTS

1. Dialog Boxes

In JavaScript, dialog boxes are commonly used to interact with users, display messages, or prompt for input within a web application. There are several types of dialog boxes available in JavaScript:

1.2 Alert Dialog 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.

Syntax

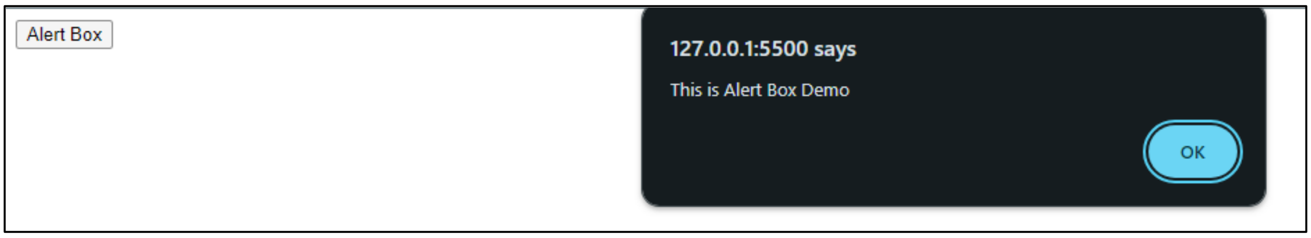
```
window.alert("sometext");
```

The window.alert() method can be written without the window prefix.

Example

```
<html>
<head>
    <script>
        function alertdemo() {
            alert("This is Alert Box Demo");
        }
    </script>
</head>
<body>
    <input type="button" value="Alert Box" onClick="alertdemo()" />
</body>
</html>
```

Output



```
127.0.0.1:5500 says
This is Alert Box Demo
```

1.3 Confirm Dialog Box

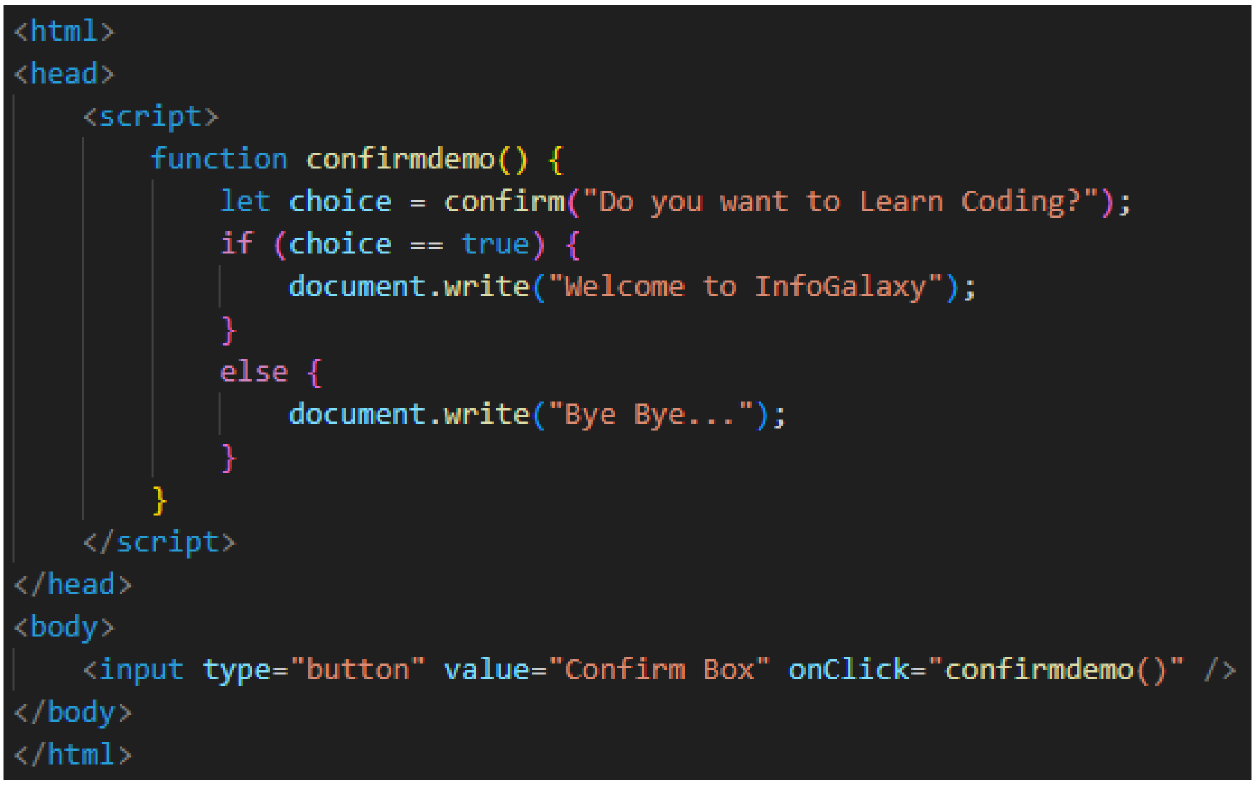
- A confirm box is often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

```
window.confirm("sometext");
```

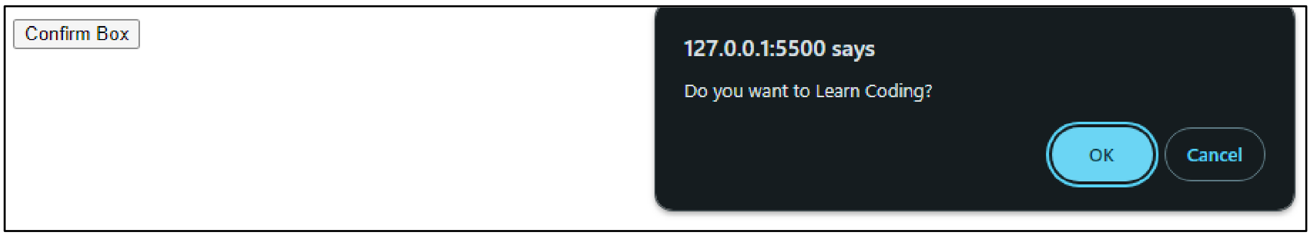
- The window.confirm() method can be written without the window prefix.

Example



```
<html>
<head>
    <script>
        function confirmdemo() {
            let choice = confirm("Do you want to Learn Coding?");
            if (choice == true) {
                document.write("Welcome to InfoGalaxy");
            }
            else {
                document.write("Bye Bye...");
            }
        }
    </script>
</head>
<body>
    <input type="button" value="Confirm Box" onClick="confirmdemo()" />
</body>
</html>
```

Output

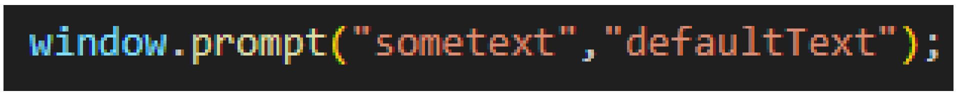


Do you want to Learn Coding?

1.4 Prompt Dialog 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.

Syntax



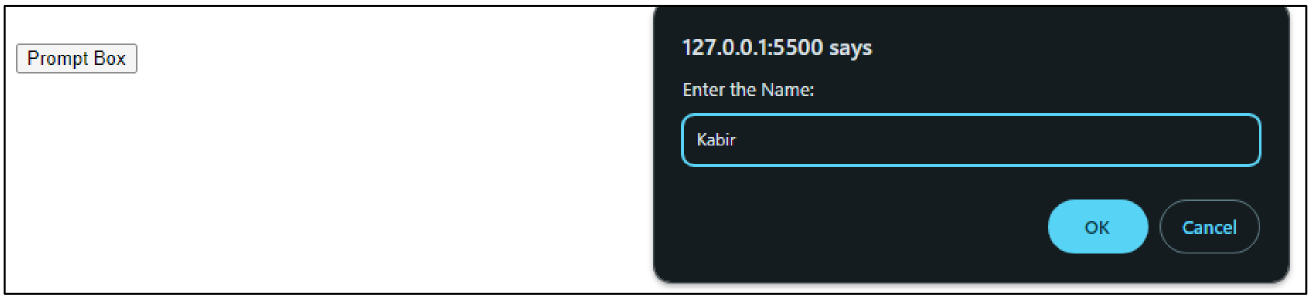
```
window.prompt("sometext","defaultText");
```

- The window.prompt() method can be written without the window prefix.

Example

```
<html>
<head>
    <script>
        function promptdemo() {
            let name = prompt("Enter the Name:");
            if (name == null) {
                document.write("Please Enter the Name");
            }
            else {
                document.write("Name=" + name);
            }
        }
    </script>
</head>
<body>
    <br><input type="button" value="Prompt Box" onClick="promptdemo()" />
</body>
</html>
```

Output



2. JavaScript Events

- In JavaScript, events are actions or occurrences that happen in the system, triggered by users or the system itself.
- Events can be anything from a user clicking the mouse, pressing a key, resizing a window, or the system loading a web page.
- Handling events is a crucial aspect of web development as it allows developers to create interactive and dynamic user experiences. Here's an overview of events in JavaScript:

Event Handlers:

- Event handlers are functions that execute in response to a specific event occurring. In HTML, event handlers can be specified directly as attributes on HTML elements:

```
<button onclick="handleClick()">Click me</button>
```

Or, they can be assigned using JavaScript:

```
document.getElementById("myButton").onclick = function() {
    // Handle click event
};
```

Event Listeners:

- Event listeners are preferred over inline event handlers for better separation of concerns and maintainability.
- They allow attaching multiple event handlers to the same element and provide more flexibility:

```
document.getElementById("myButton").addEventListener("click", function() {
    // Handle click event
});
```

Common DOM Events:

There are numerous events in JavaScript that you can handle. Some common ones include:

- **Mouse Events**: click, mouseover, mouseout, mousemove, mousedown, mouseup, etc.
- **Keyboard Events**: keydown, keyup, keypress
- **Form Events**: submit, change, focus, blur
- **Window Events**: load, resize, scroll, unload
- **Document Events**: DOMContentLoaded, readystatechange
- **Touch Events**: touchstart, touchmove, touchend

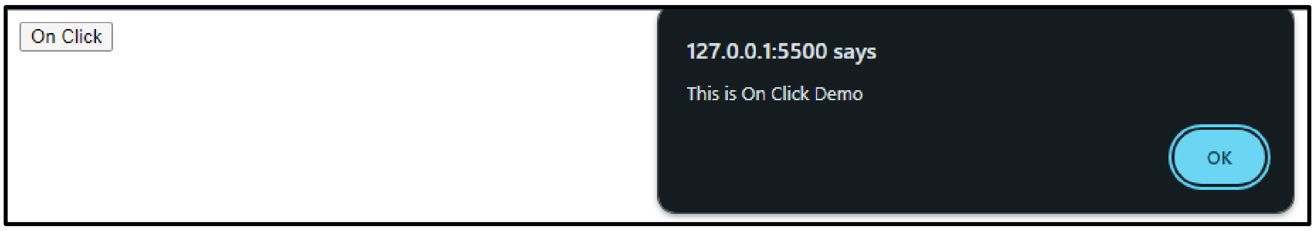
2.1 onclick

This is a mouse event and provokes any logic defined if the user clicks on the element it is bound to.

Example: In this example, we will display a message in the alert box when the button is clicked

```
<html>
<head>
    <script>
        function onclickdemo() {
            alert("This is On Click Demo");
        }
    </script>
</head>
<body>
    <input type="button" value="On Click" onClick="onclickdemo()" />
</body>
</html>
```

Output

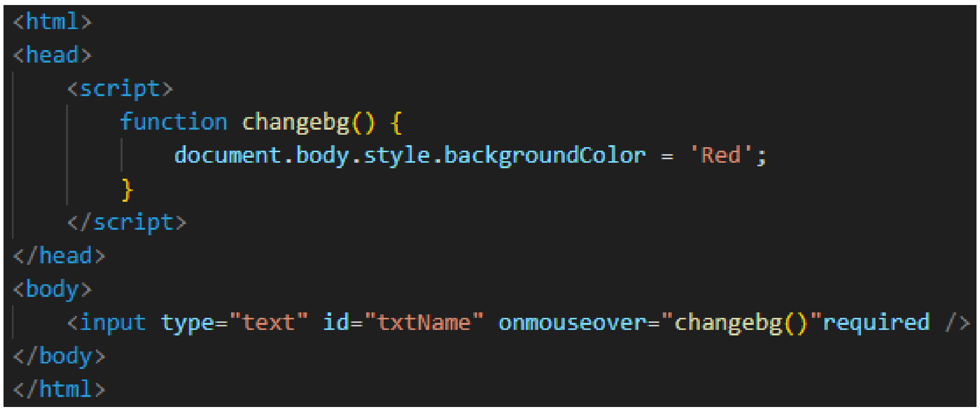


```
127.0.0.1:5500 says
This is On Click Demo
```

2.2 onmouseover

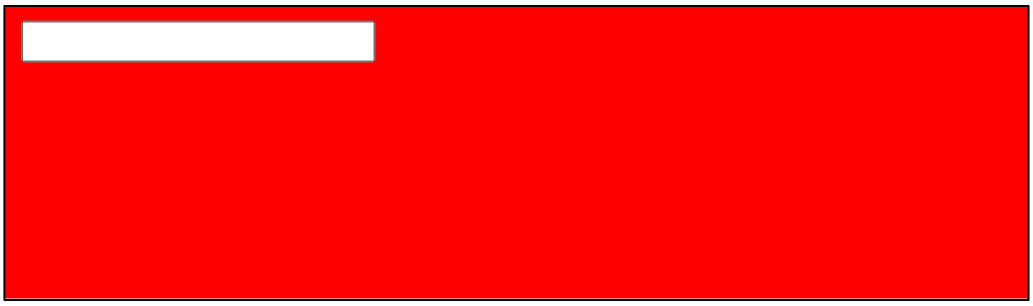
This event corresponds to hovering the mouse pointer over the element and its children, to which it is bound to.

Example: In this example, we will make the box vanish when the mouse will be hovered on it



```
<html>
<head>
    <script>
        function changebg() {
            document.body.style.backgroundColor = 'Red';
        }
    </script>
</head>
<body>
    <input type="text" id="txtName" onmouseover="changebg()"required />
</body>
</html>
```

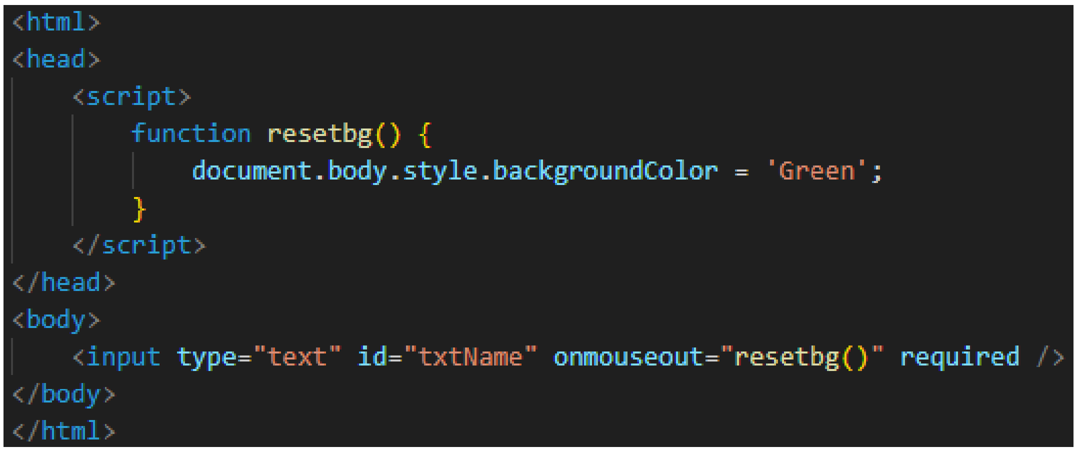
Output



2.3 onmouseout

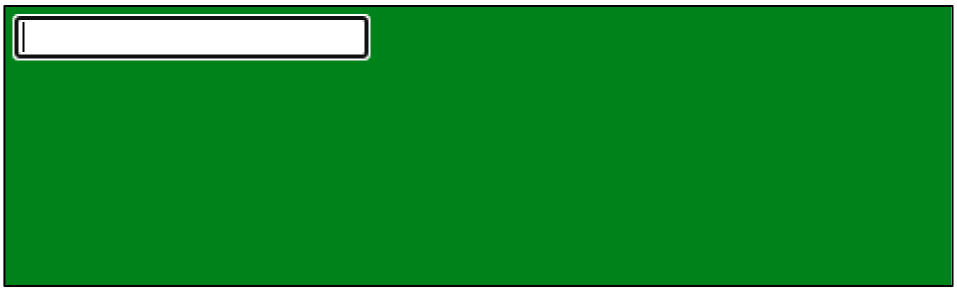
Whenever the mouse cursor leaves the element which handles a mouseout event, a function associated with it is executed.

Example:



```
<html>
<head>
    <script>
        function resetbg() {
            document.body.style.backgroundColor = 'Green';
        }
    </script>
</head>
<body>
    <input type="text" id="txtName" onmouseout="resetbg()" required />
</body>
</html>
```

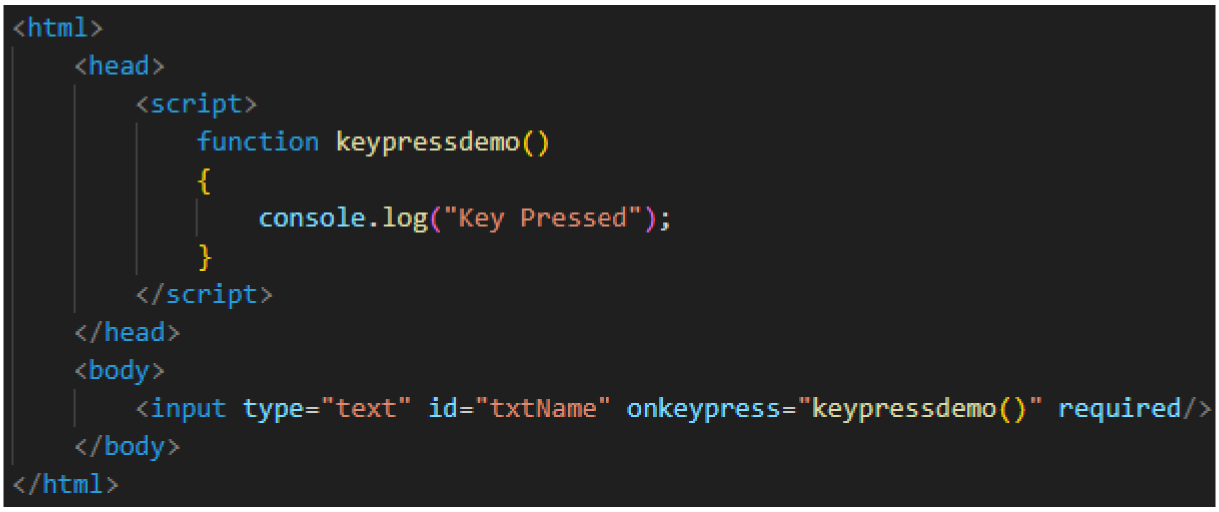
Output



2.4 onkeypress

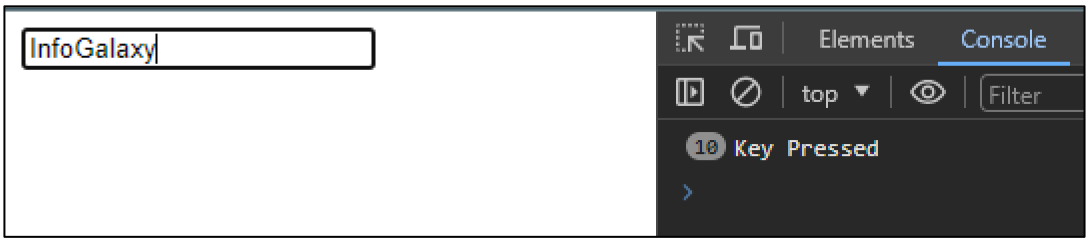
The onkeypress event occurs when the user **presses a key** on the keyboard.

Example:



```
<html>
    <head>
        <script>
            function keypressdemo()
            {
                console.log("Key Pressed");
            }
        </script>
    </head>
    <body>
        <input type="text" id="txtName" onkeypress="keypressdemo()" required/>
    </body>
</html>
```

Output



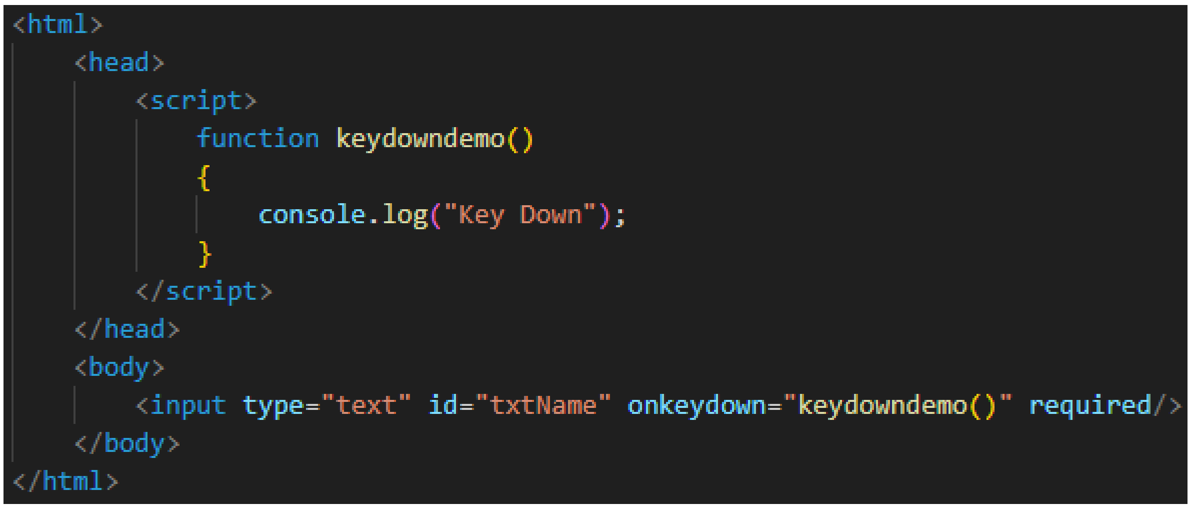
```
InfoGalaxy
```

```
10 Key Pressed
```

2.5 onkeydown

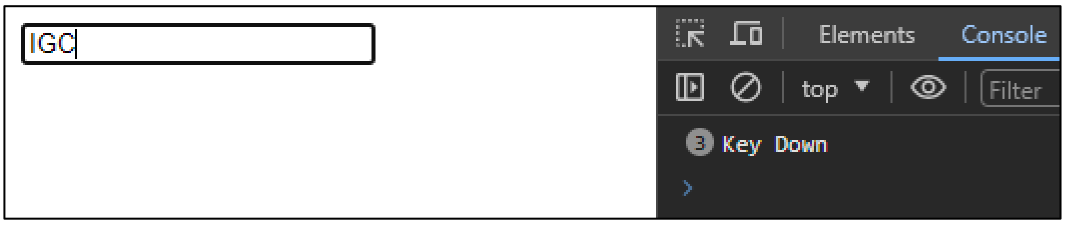
The onkeydown event occurs when the user **presses a key** on the keyboard.

Example:



```
<html>
    <head>
        <script>
            function keydowndemo()
            {
                console.log("Key Down");
            }
        </script>
    </head>
    <body>
        <input type="text" id="txtName" onkeydown="keydowndemo()" required/>
    </body>
</html>
```

Output



```
IGC
```

```
3 Key Down
```

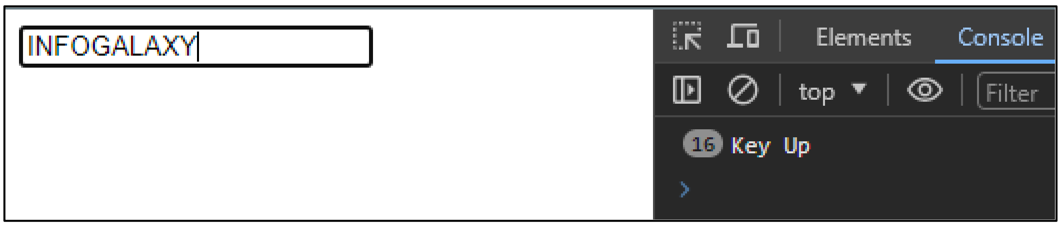
2.6 onkeyup

This event is a keyboard event and executes instructions whenever a key is released after pressing.

Example: In this example, we will change the color by pressing UP arrow key

```html
<html>
    <head>
        <script>
            function keyupdemo()
            {
                console.log("Key Up");
            }
        </script>
    </head>
    <body>
        <input type="text" id="txtName" onkeyup="keyupdemo()" required/>
    </body>
</html>
```

Output



```
INFOGALAXY
16 Key Up
```

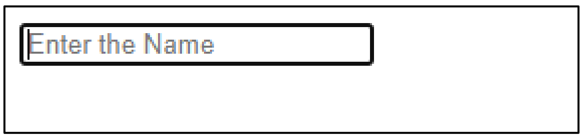
2.7 onfocus

This Javascript function performs when the given instruction receives the focus as per the change or click event. The below code snippet can help you understand the logic.

Example:

```html
<html>
    <head>
        <script>
            function onfocusdemo()
            {
                document.getElementById("txtName").placeholder="Enter the Name";
            }
        </script>
    </head>
    <body>
        <input type="text" id="txtName" onfocus="onfocusdemo()" required/>
    </body>
</html>
```

Output

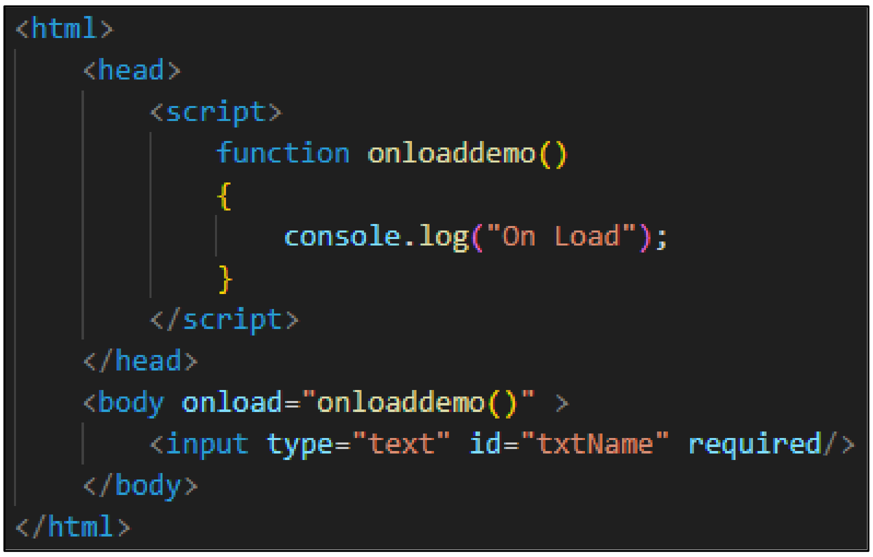


```
Enter the Name
```

2.8 onload

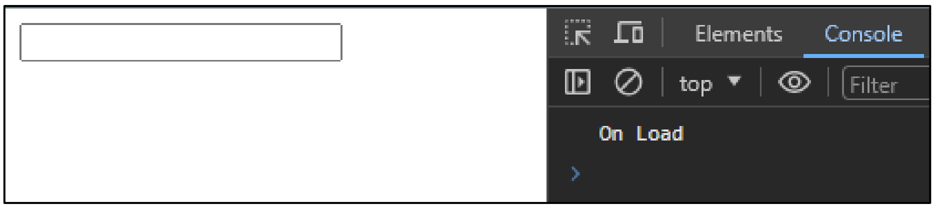
onload event can be utilized when we have a specific requirement to execute a specific function once the page is represented fully. The below code snippet can help you understand the logic.

Example:



```
<html>
    <head>
        <script>
            function onloaddemo()
            {
                console.log("On Load");
            }
        </script>
    </head>
    <body onload="onloaddemo()" >
        <input type="text" id="txtName" required/>
    </body>
</html>
```

Output

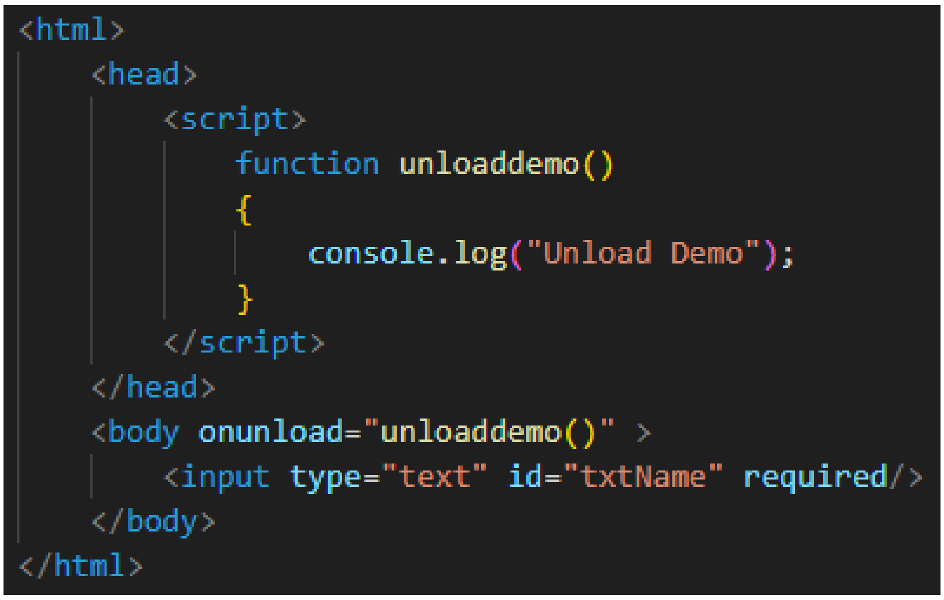


```
On Load
>
```

2.9 onunload

The onunload event occurs once a page has unloaded (or the browser window has been closed).

Example:



```
<html>
    <head>
        <script>
            function unloaddemo()
            {
                console.log("Unload Demo");
            }
        </script>
    </head>
    <body onunload="unloaddemo()" >
        <input type="text" id="txtName" required/>
    </body>
</html>
```

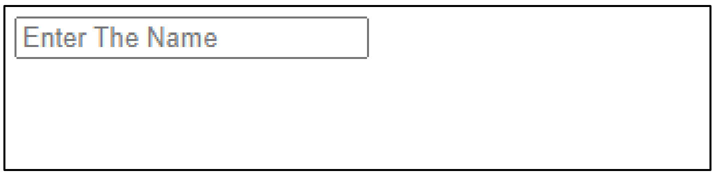
2.10 onblur

Onblur event triggers when a certain object loses focus. We can execute the below code to understand how to implement it.

Example:

```html
<html>
    <head>
        <script>
            function onblurdemo()
            {
                document.getElementById("txtName").placeholder="Enter The Name";
            }
        </script>
    </head>
    <body>
        <input type="text" id="txtName" onblur="onblurdemo()" required/>
    </body>
</html>
```

Output


Enter The Name

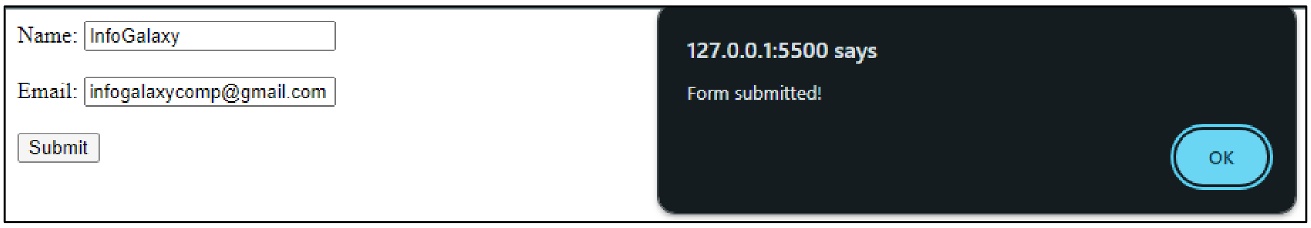
2.11 onsubmit

The onsubmit event occurs when a form is submitted.

Example:

```html
<html>
<head>
    <script>
        function onSubmitForm() {
            alert('Form submitted!');
        }
    </script>
</head>
<body>
    <form onsubmit="return onSubmitForm()">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required><br><br>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required><br><br>
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

Output


Name: InfoGalaxy

Email: infogalaxycomp@gmail.com

127.0.0.1:5500 says

Form submitted!