Introduction to JavaScript Events

- When you browse the web, your browser registers different types of events. It's the browser's way of saying , "Hey, this just happened." Your script can then respond to these events.
- Scripts often respond to these events by updating the content of the web page via DOM which makes the page feel more interactive.
- Events occur when users click or tap on a link, hover or swipe over an element, type on the keyboard, seizes the window, or when the page they requested has loaded.
- When an events occurs, or fires, it can be used to trigger a particular function. Different code can be triggered when users interact with different parts of the page.
- The events can trigger the kinds of changes the DOM is capable of. This is how web page reacts to users.
- **Events Fire or Raised**: When an event has occurred, it is often described as having **fired** or been **raised**.
- **Events trigger script:** Events are said to **trigger** a function or script. When the click event fires on the element, it could trigger a script that does the particular task.

Different Event Types

Some of the common events are listed below:

UI Events: Occur when a user interacts with the browser's user interface rather than the web page.

Events	Description
load	Web page has finished loading
unload	Web page is unloading (usually because a new page was requested)
error	Browser encounters a JavaScript error or and asset doesn't exist
resize	Browser window has been resized
scroll	User has scrolled up or down the page

Keyboard EVENTS: Occur when a user interacts with the keyboard

Events	Description
keydown	User first presses a key

keyup	User releases a key
keypress	Character is being inserted

MOUSE EVENTS: Occur when a user interacts with a mouse, trackpad, or touchscreen

Events	**Description**
click	User presses and releases a button over the same element
dblclick	User presses and releases a button twice over the same element
mosedown	User presses a mouse button while over an element
mouseup	User releases a mouse button while over an element
mousemove	User moves the mouse
mouseover	User moves the mouse over an element (not on a touchscreen)
mouseout	User moves the mouse off an element (not on a touchscreen)

FOCUS EVENTS: Occur when an element (eg, a link or form field) gains or loses focus.

Events	**Description**
focus / focusin	Element gains focus
flur/focusout	Element loses focus

FORM EVENTS: occur when a user interact with form element.

Events	**Description**
input	Value in any <input> or <textarea> element has changed or any element with the content editable attribute.
change	Value in select box, checkbox or radio button changes
submit	User submits a form using a button to key
reset	User clicks on a form's reset button
cut	User cuts the content from a form field
copy	User copies content form a form field
paste	User paste content into a form field
select	User selects some text in a form field

HOW EVENTS TRIGGER JAVASCRIPT CODE

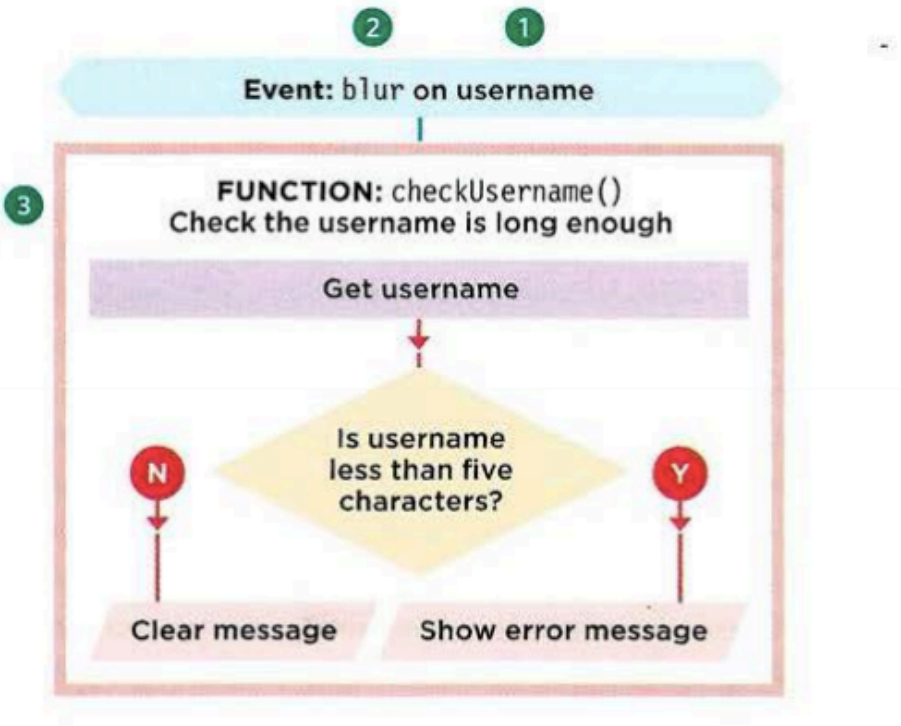
When the user interacts with the HTML on a web page, there are three steps involved in getting it to trigger some JavaScript code. Together these steps are known as **event handling.**

1. Select the **element** node(s) you want the script to respond to.
2. Indicate which **event** on the selected node(s) will trigger the response.
3. State the **code** you want to run when the event occurs.

EXAMPLE: In given example you can see how event handling can be used to provide feedback to users filling in a registration form. It will show an error message if their username is too short.

1. **Select element**: The element that users are interacting with is the text input where they enter the user name.
2. **Specify event**: when users move out of the text input, it loses focus, and the blur event fires on this element.

3. **Call code:** When the blur event fires on the username input, it will trigger a function called checkUsername( ).This function checks if the username is less than 5 characters. If there are not enough characters, it shows an error message that prompts the user to enter a longer username. If there are enough characters, the element that holds the error message should be cleared. This is because an error message may have been shown to the user already and they subsequently corrected their mistake. (If the error message was still visible when they had filled in the form correctly, it would be confusing.)



In javaScript, there are

several ways to associate an event handler to the event.

1. HTML Event handlers
   - Early versions of HTML included a set of attributes that could respond to events on the element they were added to.

```
<input type="button" name="" value="click me" id="btn"
onmouseover="this.style.background='red';this.style.color='yellow'"
onmouseout="this.style.background='black';this.style.color='white'">
```

   - This method can be written using method call. The attribute name should matched event name. Their value called the function that was to run when that even occurred.
   - The above same code will look like below:

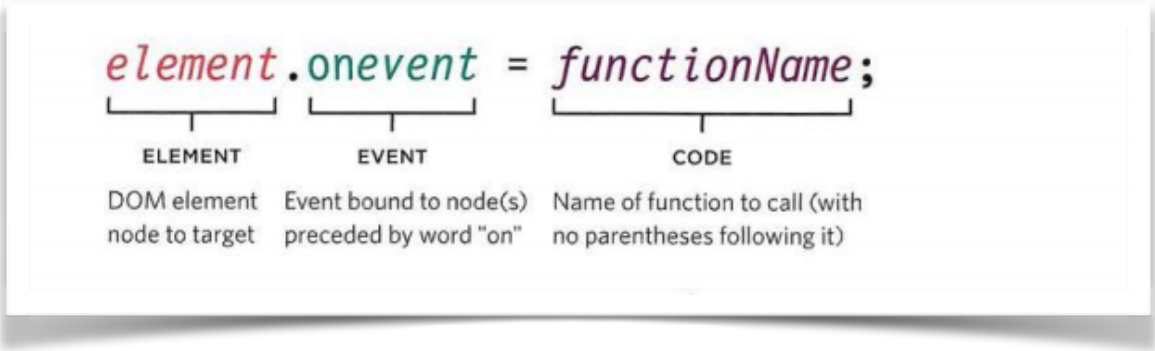
2. Traditional DOM event handlers
   - DOM event handlers were introduced in the original specification for the DOM. They are considered better than HTML event handlers because they let you separate the Javascript code from HTML.
   - All modern browser understand this way of creating an event handler, but you can only attach one function to each event handler.

```
<input type="button" value="Click me" id="btn"
onmouseover="changeColorOnMouseOver()"
onmouseout="changeColorOnMouseOut()" />

<script type="text/javascript">
    function changeColorOnMouseOver()
    {
        var control = document.getElementById("btn");
        control.style.background = 'red';
        control.style.color = 'yellow';
    }

    function changeColorOnMouseOut()
    {
        var control = document.getElementById("btn");
        control.style.background = 'black';
        control.style.color = 'white';
    }
</script>
```

- Syntax:

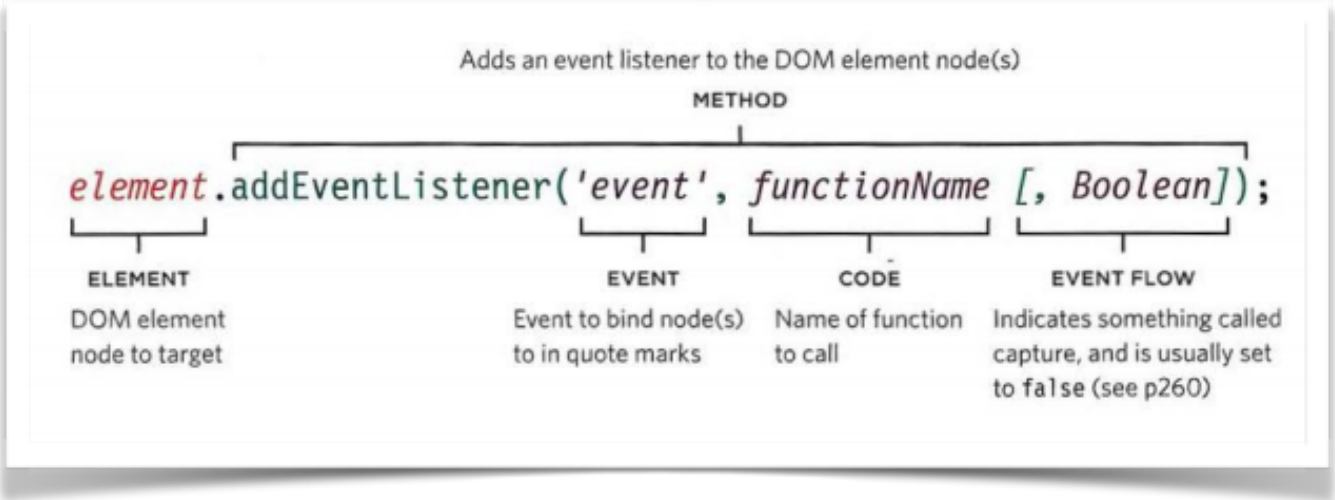


Example:

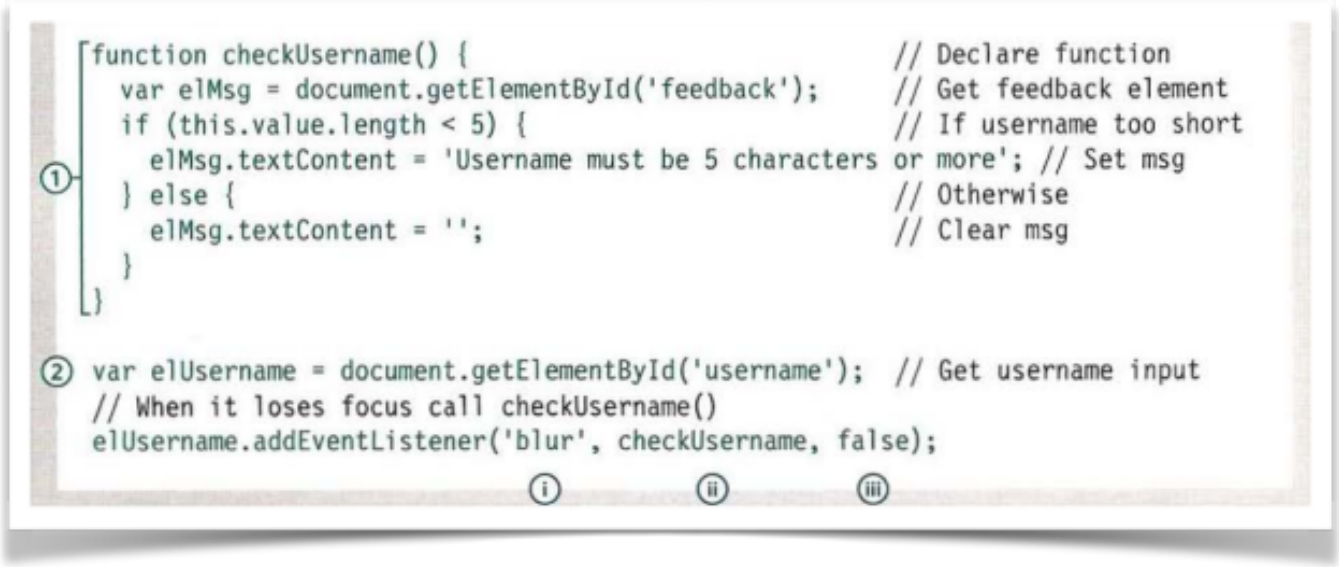
```
function checkUsername() {                                  // Declare function
  var elMsg = document.getElementById('feedback');          // Get feedback element
  if (this.value.length < 5) {                              // If username too short
    elMsg.textContent = 'Username must be 5 characters or more';  // Set msg
  } else {                                                  // Otherwise
    elMsg.textContent = '';                                 // Clear message
  }
}

var elUsername = document.getElementById('username');  // Get username input
elUsername.onblur = checkUsername;  // When it loses focus call checkuserName()
```

3. DOM level 2 event listener
   - Events listeners are a more recent approach to handling events. They can deal with more than one function at a time but they are not supported in older browsers.
   - Syntax



- Example


function checkUsername() {
// Declare function
var elMsg = document.getElementById('feedback');
// Get feedback element
if (this.value.length < 5) {
// If username too short
elMsg.textContent = 'Username must be 5 characters or more'; // Set msg
} else {
// Otherwise
elMsg.textContent = '';
// Clear msg
}
}
1
2
var elUsername = document.getElementById('username');
// Get username input
// When it loses focus call checkUsername()
elUsername.addEventListener('blur', checkUsername, false);
i
ii
iii