Events

Interactions create event

Events trigger code

Code responds to users

Scripts often respond to events by updating the content of the page (via DOM) which makes the page appear interactive.
(Some) Event Types

**UI events** – occur when a user interacts with the browser’s user interface (UI) – work with window object

<table>
<thead>
<tr>
<th>UI Events</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>load</td>
<td>Web page has (just) finished loading</td>
</tr>
<tr>
<td>unload</td>
<td>Web page is unloading</td>
</tr>
<tr>
<td>error</td>
<td>Browser encounters a JavaScript error or unavailable web resources</td>
</tr>
<tr>
<td>resize</td>
<td>Browser window has been resized</td>
</tr>
<tr>
<td>scroll</td>
<td>User has scrolled up or down the page</td>
</tr>
</tbody>
</table>
(Some) Event Types

**Focus events** – occur when an element gains or loses focus

<table>
<thead>
<tr>
<th>Focus Events</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>focus / focusin</td>
<td>Element gains focus</td>
</tr>
<tr>
<td>blur / focusout</td>
<td>Focus loses focus</td>
</tr>
</tbody>
</table>

**Keyboard events** – occur when a user interacts with the keyboard

<table>
<thead>
<tr>
<th>Keyboard Events</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>keydown</td>
<td>User first presses a key</td>
</tr>
<tr>
<td>keyup</td>
<td>User releases a key</td>
</tr>
<tr>
<td>keypress</td>
<td>Character is being inserted</td>
</tr>
</tbody>
</table>
## (Some) Event Types

**Mouse events** – occur when a user interacts with a mouse, trackpad, or touchscreen

<table>
<thead>
<tr>
<th>Mouse Events</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>click</td>
<td>User presses and releases a button over the same element</td>
</tr>
<tr>
<td>dblclick</td>
<td>User presses and releases a button twice over the same element</td>
</tr>
<tr>
<td>mousedown</td>
<td>User presses a mouse button while over an element</td>
</tr>
<tr>
<td>mouseup</td>
<td>User releases a mouse button while over an element</td>
</tr>
<tr>
<td>mousemove</td>
<td>User moves a mouse (not on a touchscreen)</td>
</tr>
<tr>
<td>mouseover</td>
<td>User moves a mouse over an element (not on a touchscreen)</td>
</tr>
<tr>
<td>mouseout</td>
<td>User moves a mouse off an element (not on a touchscreen)</td>
</tr>
</tbody>
</table>
### (Some) Event Types

**Form events** – occur when a user interacts with a form element

<table>
<thead>
<tr>
<th>Form Events</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>Value in any <code>&lt;input&gt;</code> or <code>&lt;textarea&gt;</code> element has changed or any element with the contenteditable attribute</td>
</tr>
<tr>
<td>change</td>
<td>Value in select box, checkbox, or radio button changes</td>
</tr>
<tr>
<td>submit</td>
<td>User submits a form (using a button or a key)</td>
</tr>
<tr>
<td>reset</td>
<td>User clicks on a form’s reset button</td>
</tr>
<tr>
<td>cut</td>
<td>User cuts content from a form field</td>
</tr>
<tr>
<td>paste</td>
<td>User pastes content into a form field</td>
</tr>
<tr>
<td>select</td>
<td>Users selects some text in a form field</td>
</tr>
</tbody>
</table>
# (Some) Event Types

Mutation events – occur when the DOM structure has been changed by a script

<table>
<thead>
<tr>
<th>Mutation Events</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOMSubtreeModified</td>
<td>Change has been made to a document</td>
</tr>
<tr>
<td>DOMNodeInserted</td>
<td>Node has been inserted as a direct child of another node</td>
</tr>
<tr>
<td>DOMNodeRemoved</td>
<td>Node has been removed from another node</td>
</tr>
</tbody>
</table>
How Events Trigger Code

Select the **element** node the script will respond to

Indicate which **event** on the selected node will trigger the response (bind an event to a DOM node)

Specify the **code** to run when the event occurs

“Event handling”
Example: Event Handling (Lazy)

```html
<form action="event-ex1.html" onsubmit="return (validateInfo())">
  <label>Username: </label>
  <input type="text" id="username" autofocus required onblur="checkUsername()" />
  <div id="user-msg" class="feedback"></div>
  
  <label>Password: </label>
  <input type="password" id="pwd" required />
  <div id="pwd-msg" class="feedback"></div>
  
  <input type="submit" value="Sign in" />
</form>

```

```javascript
function checkUsername() {
  var msg = document.getElementById("user-msg");
  var user = document.getElementById("username");
  if (user.value.length < 8 && user.value.length > 0)
    msg.textContent = "Username is too short";
  else
    msg.textContent = "";
}
```

bind-event-lazy.html
Example: Event Handling (Tradition)

```html
<form action="event-ex1.html" onsubmit="return (validateInfo())">
  <label>Username: </label>
  <input type="text" id="username" autofocus required />
  <div id="user-msg" class="feedback"></div>
  <br/>
  <label>Password: </label>
  <input type="password" id="pwd" required />
  <div id="pwd-msg" class="feedback"></div>
  <br/>
  <input type="submit" value="Sign in" />
</form>

function checkUsername() {
  var msg = document.getElementById("user-msg");
  if (this.value.length < 8 && this.value.length > 0)
    msg.textContent = "Username is too short";
  else
    msg.textContent = "";
}

bind-event-tradition.html

var user = document.getElementById("username");
user.onblur = checkUsername;
```
Example: Event Handling (Modern)

```
<form action="event-ex1.html" onsubmit="return (validateInfo())">
    <label>Username: </label>
    <input type="text" id="username" autofocus required />
    <div id="user-msg" class="feedback"></div>
    <br/>
    <label>Password: </label>
    <input type="password" id="pwd" required />
    <div id="pwd-msg" class="feedback"></div>
    <br/>
    <input type="submit" value="Sign in" />
</form>

function checkUsername() {
    var msg = document.getElementById("user-msg");
    if (this.value.length < 8 && this.value.length > 0) {
        msg.textContent = "Username is too short";
    } else {
        msg.textContent = "";
    }
}

var user = document.getElementById("username");
user.addEventListener('blur', checkUsername, false);
```

bind-event-modern.html
Example: Using Parameters with Event Listener

```html
<form action="event-ex1.html" onsubmit="return (validateInfo())">
  <label>Username: </label>
  <input type="text" id="username" autofocus required />
  <div id="user-msg" class="feedback"></div>
  <br/>
  <label>Password: </label>
  <input type="password" id="pwd" required />
  <div id="pwd-msg" class="feedback"></div>
  <br/>
  <input type="submit" value="Sign in" />
</form>

```javascript
var user = document.getElementById("username");

function checkUsername(minLength) {
  var msg = document.getElementById("user-msg");
  if (user.value.length < minLength && user.value.length > 0) {
    msg.textContent = "Username must be " + minLength + " characters or more";
  } else {
    msg.textContent = "";
  }
}

user.addEventListener('blur', function() {
  checkUsername(8);
}, false);

bind-event-with-param.html
Example: Supporting Older Version of IE (w/o Param)

```javascript
var user = document.getElementById("username");
var msg = document.getElementById("user-msg");

function checkUsername() {
    if (user.value.length < 8 && user.value.length > 0)
        msg.textContent = "Username is too short"
    else
        msg.textContent = "";
}

if (user.addEventListener) { // if event listener supported
    user.addEventListener('blur', checkUsername, false);
}
else // for older versions of IE
     // old IE has its own method attachEvent that does the same as addEventListener
    user.attachEvent('onblur', checkUsername);

bind-event-support-multiple-browsers.html
```
Example: Supporting Older Version of IE (w/ Param)

```javascript
var user = document.getElementById("username");
function checkUsername(minLength) {
    var msg = document.getElementById("user-msg");
    if (user.value.length < minLength && user.value.length > 0)
        msg.textContent = "Username is too short";
    else
        msg.textContent = "";
}

if (user.addEventListener) {
    user.addEventListener('blur', function() {
        checkUsername(8);
    }, false);
} else { // older version of IE uses attachEvent method, which does the same thing as addEventListener
    user.attachEvent('onblur', function() {
        checkUsername(8);
    });
}
```

// if event listener supported
// when username field loses focus
// call checkUsername function
// use default type of event flow

bind-event-with-param.html
Summary

• Events occur when user interacts with a web browser or a web page (screen).

• Binding specifies the event that can happen and on which element the event happens.

• When an event occurs on an element, it can trigger a JavaScript function. The function then manipulates the page to response to the user.