



华中师范大学伍伦贡联合研究院  
Central China Normal University Wollongong Joint Institute



UNIVERSITY  
OF WOLLONGONG  
AUSTRALIA

# CSIT884

# Web Development

Lecture 03B - JavaScript Events

# Objectives

- Understand JavaScript events
- Write code to handle DOM events
- Confirm & Prompt boxes



# Event

In JavaScript, we use events (such as clicking a button, pressing a mouse, ...) to control user interaction with the web page.

```
<button onClick="handlerFuntion1()">  
  Click me  
</button>  
  
<button onMouseOver="handlerFuntion2()">  
  some text  
</button>
```

JavaScript event handler is often associated with a HTML element (such as a button, an image, a text span,...), by specifying an attribute such as `onClick`, `onMouseOver`, `onChange`, ... We often write a function so that when the event happens the function will get executed.

# onClick

The **onClick** event is activated when the user clicks on the element such as a button, an image, a check box, a radio button, ...

```
<script>
```

```
function cat(){
```

```
    alert("Don't click me, click the dog!");
```

```
}
```

```
function dog(){
```

```
    alert("I don't mind being clicked!");
```

```
}
```

```
</script>
```

```
 . . . .(1)
```

```
 . . . .(2)
```




# onLoad

The **onLoad** event is activated after the element's resource is completely loaded, for example,

- On <body> element, the onLoad event is activated after all the contents of the web page have been loaded (including images, JavaScript files, CSS files, etc.)
- On <img> element, the onLoad event is activated after the image file has been retrieved from the server.

```
<script>
function greeting(){
    alert("Welcome to my website!");
}
</script>
<body onLoad="greeting()">
```



# onFocus

The **onFocus** event is activated when the element is in focus, such as,

- when the user enters into a text/password field to type,
- when the user enters into a textarea to type,
- when the user enters into a drop-down list to make a selection.

```
<input type="text" onFocus="..." />
```

```
<textarea onFocus="...">
```

```
</textarea>
```

```
<select onFocus="...">
```

```
  <option value="...">..</option>
```

```
  <option value="...">...</option>
```

```
</select>
```



# onBlur

The **onBlur** event is activated when the element loses focus, such as,

- when the user moves out of a text/password field,
- when the user moves out of a textarea,
- when the user moves out of a drop-down list.

```
<input type="text" onBlur="..." />
```

```
<textarea onBlur="...">  
</textarea>
```

```
<select onBlur="...">  
  <option value="...">..</option>  
  <option value="...">...</option>  
</select>
```

# onChange

The **onChange** event is activated when the element loses the focus and its value has been changed, such as,

- when the user enters a text/password field, types some text, then moves out of the text field,
- when the user enters a textarea, types some text, then moves out of the textarea,
- when the user makes a new selection in a drop-down list.

Note that if the user goes into a text field but does not make changes to the text, then when the user moves out, this event will not be fired. Similarly, if the user selects the same option in a drop-down list, then this event will not be fired.



# onChange

Example: When the user enters the discount code, leaves the text field, a function is triggered which transforms the discount code to uppercase:

Enter discount code:

```
<input type="text" id="discountCode" onChange="uppercase( )">
```

```
function uppercase( ) {  
    var discountField = document.getElementById("discountCode");  
    discountField.value = discountField.value.toUpperCase();  
}
```

# onSelect

The **onSelect** event is activated when the user selects (highlights) some text in a text/password field or a textarea.

```
<input type="text" onSelect="..." />
```

```
<textarea onSelect="...">  
</textarea>
```

# Mouse Event

- **onMouseDown**: the event is activated when the user presses a mouse button over the element.
- **onMouseUp**: the event is activated when the user releases a mouse button over the element.
- **onMouseOver**: the event is activated when the user moves the mouse over the element.
- **onMouseOut**: the event is activated when the user moves the mouse out of the element.

# Mouse Event

```
<span id="demo" onMouseDown="mouseDown( )" onMouseUp="mouseUp( )">  
  Mouse Events  
</span>
```

```
function mouseDown() {  
  var demoSpan = document.getElementById("demo");  
  demoSpan.innerHTML = "Release Me";  
}  
  
function mouseUp() {  
  var demoSpan = document.getElementById("demo");  
  demoSpan.innerHTML = "Thank You";  
}
```

# Mouse Event

```
<span id="demo" onMouseOver="mouseover( )" onMouseOut="mouseout( )">  
  Mouse Events  
</span>
```

```
function mouseover() {  
  var demoSpan = document.getElementById("demo");  
  demoSpan.innerHTML = "Mouse Over Me";  
}  
  
function mouseout() {  
  var demoSpan = document.getElementById("demo");  
  demoSpan.innerHTML = " Mouse out";  
}
```

# onSubmit

The **onSubmit** event is activated when the user submits a form.

```
<script>
function validateForm(){
  if (... something wrong ...) {
    return false;
  }
  return true;
}
</script>
```

```
<form onSubmit="return validateForm()" action="..." method="..." >
...
</form>
```

We often use this to validate the form

- we can use function that checks the user input,
- if the user input is not valid, the function should return false, and that will stop the form from submission to the server
- If everything is valid, the function should return true, in this case, the form will be submitted to the server

# Confirm box

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

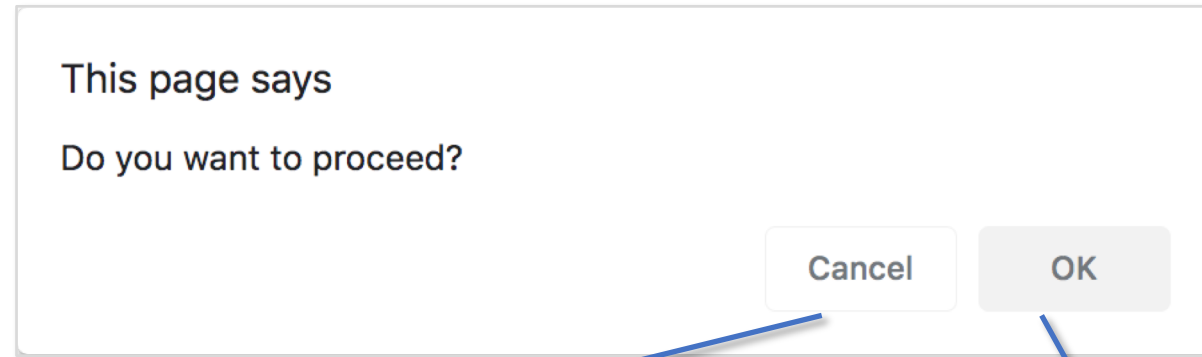
When a confirm box pops up, the user will have to click either "OK" or "Cancel".

If the user clicks "OK", the box returns `true`.

If the user clicks "Cancel", the box returns `false`.

```
var ok = confirm("Do you want to proceed?");  
if(ok){  
    alert("User clicked OK");  
}else{  
    alert("User clicked Cancel.");  
}
```

# Confirm box



This page says  
User clicked Cancel.

OK

This page says  
User clicked OK

OK



# Prompt box

When a prompt box pops up, the user will have to click either "OK" or "Cancel".

If the user clicks "OK" the box returns the input value.

If the user clicks "Cancel" the box returns null.

We can also specify the default text in the input box:

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

```
var name = prompt("Please enter your name", "cat in the hat");  
if(name != null){  
    alert("Hello " + name);  
}else{  
    alert("User clicked Cancel.");  
}
```

# Prompt box

This page says

Please enter your name

Cancel OK

This page says  
User clicked Cancel.

This page says  
Hello John

This page says  
Hello cat in the hat

OK

OK



# References

- Robert W. Sebesta, Programming the World Wide Web, 8th edition, Pearson, 2015
- Jennifer Niederst Robbins, Learning Web Design - A Beginner's guide to HTML, CSS, JavaScript and Web Graphics, 5th edition, O' Reilly Media, 2018
- David Flanagan, JavaScript: The Definitive Guide, Seventh Edition, O' Reilly Media, Inc., 2020
- <http://www.w3schools.com/js>
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript>