SOEN 287

Chapter 5: JavaScript and HTML Documents (2)

Dr. Yuhong Yan
CSE, Concordia University
Winter, 2024

Events and Event Handling

- An *event* is a notification that something specific has occurred, either with the browser or an action of the browser user
- An *event handler* is a script that is implicitly executed in response to the appearance of an event
- The process of connecting an event handler to an event is called *registration*
- Don't use `document.write` in an event handler because the output may go on top of the display

Click a button

- Method1: assign handler to attribute

```
<form id = "form1">
  <input type = "button"  id = "myButton"
       onclick = "alert('you clicked my button');" />
</form>
```

or onclick = "myButtonHandler();"

- Method2: assign handler to property

```
<form id = "form1">
  <input type = "button"  id = "myButton"/>
</form>
…
function myButtonHandler(){
            alert("You clicked my button!!");
}
document.getElementById("myButton").onclick =
   myButtonHandler;
```

Click a button – Method3

- Method3: addEventListener

```
<form id = "form1">
  <input type = "button"  id = "myButton"/>
</form>
…
function () myButtonHandler{
              alert("You clicked my button!!");
}
document.getElementById("myButton”).
addEventListener("click", myButtonHandler);
```

→**SHOW** `chap5/button_click_24.html`

→**SHOW** `chap5/button_click_24_2.html`

→**SHOW** `chap5/button_click_24_3.html`

Explain Code

- `${}`

```
const rndCol = `rgb(${random(255)}, ${random(255)},
${random(255)})`;

const rndCol = `rgb(${random(255)} ${random(255)}
${random(255)})`;
```

→**SHOW** `chap5/button_click_24.html`

Explain Code

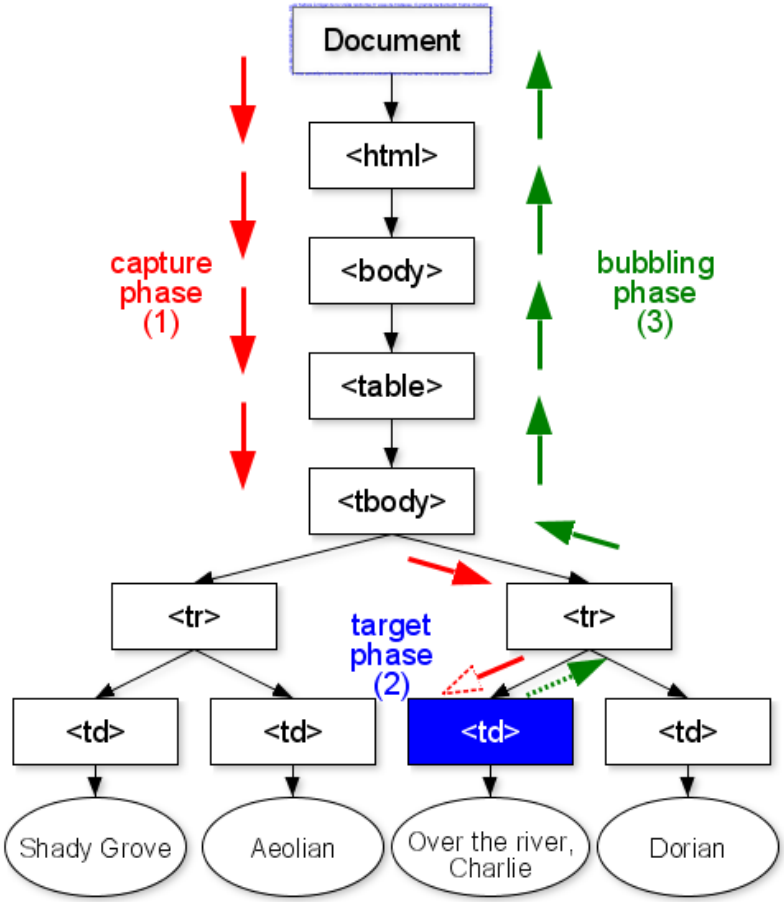
- `${} ` : evaluate and embed expressions dynamically in template literals

```
const name = 'John Doe';
const age = 20;

// Using template literals for string interpolation
console.log(`My name is ${name} and I'm ${age} years
old.`);;
```

→**SHOW** `chap5/button_click_24.html`

Event propagation: capture phase and bubbling phase



Capture or bubbling?

- *useCapture*=true: the capturing propagation, the outer most element's event is handled first and then the inner
- *useCapture*=false: the bubbling propagation, the inner most element's even is handled first and then the outer

addEventListener(event, function, useCapture);

https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_usecapture

Events and Attributes

Event	Tag Attribute
blur	onblur
change	onchange
click	onclick
dblclick	ondblclick
focus	onfocus
keydown	onkeydown
keypress	onkeypress
keyup	onkeyup
load	onload
mousedown	onmousedown
mousemove	onmousemove
mouseout	onmouseout
mouseover	onmouseover
mouseup	onmouseup
reset	onreset
select	onselect
submit	onsubmit
unload	onunload

Event attributes and their tags

Attribute	Tag	Description
onblur	<a>	The link loses the input focus
	<button>	The button loses the input focus
	<input>	The input element loses the input focus
	<textarea>	The text area loses the input focus
	<select>	The selection element loses the input focus
onchange	<input>	The input element is changed and loses the input focus
	<textarea>	The text area is changed and loses the input focus
	<select>	The selection element is changed and loses the input focus
onclick	<a>	The user clicks on the link
	<input>	The input element is clicked
ondblclick	Most elements	The user double clicks the left mouse button
onfocus	<a>	The link acquires the input focus
	<input>	The input element receives the input focus
	<textarea>	A text area receives the input focus
	<select>	A selection element receives the input focus

Event attributes and their tags (continued)

Attribute	Tag	Description
onkeydown	<body>, form elements	A key is pressed down
onkeypress	<body>, form elements	A key is pressed down and released
onkeyup	<body>, form elements	A key is released
onload	<body>	The document is finished loading
onmousedown	Most elements	The user clicks the left mouse button
onmousemove	Most elements	The user moves the mouse cursor within the element
onmouseout	Most elements	The mouse cursor is moved away from being over the element
onmouseover	Most elements	The mouse cursor is moved over the element
onmouseup	Most elements	The left mouse button is unclicked
onreset	<form>	The reset button is clicked
onselect	<input>	The mouse cursor is moved over the element
	<textarea>	The text area is selected within the text area
onsubmit	<form>	The *Submit* button is pressed
onunload	<body>	The user exits the document

Events and Event Handling (continued)

- The same attribute can appear in several different tags
  e.g., The `onclick` attribute can be in `<a>` and `<input>`
- *A HTML element gets focus in three ways:*
  1. When the user puts the mouse cursor over it and presses the left button
  2. When the user tabs to the element
  3. By executing the `focus` method
- Reference: all the events for HTML tags:
  http://www.w3schools.com/tags/tag_a.asp

Event

```
function bgChange(e) {
        const rndCol = `rgb(${random(255)} ${random(255)}
   ${random(255)})`;
        e.target.style.backgroundColor = rndCol;
        console.log(e);
      }

btn.addEventListener("click", bgChange);
```

→**SHOW** `chap5/button_click_24_3.html`

The `load` event at the `body` element

- the `load` event - triggered when the loading of a document is completed

→SHOW `load.html`, `load.js`, & display

```
<head>
   <title> load.html </title>
   <script type = "text/javascript"  src = "load.js" >
   </script>
 </head>
<body onload = "load_greeting();">
  This is my page<p/>
 </body>
```

load.js

```
// load.js
//   An example to illustrate the load event

// The onload event handler

function load_greeting () {
  alert("You are visiting the home page of \n" +
        "Pete's Pickled Peppers \n" + "WELCOME!!!");
  //document.write("<br/>where am I?");
}
```

Handling Events from Radio buttons

- Radio buttons - use the `onclick` property, treat different values for different choices

→**SHOW** **`radio_click.html`**, **`radio_click.js`**

→**SHOW the second way** **`radio_click2.html`**, **`radio_click2.js`**

radio_click.html

```
<head>
        <script type = "text/javascript"  src = "radio_click.js" />
</head>
 <body>
        <form id = "myForm"  action = "">
        <label> <input type = "radio"  name = "planeButton"
                value = "152"
                onclick = "planeChoice(152)" />
                  Model 152 </label>
    …
```

radio_click.js

```
function planeChoice (plane) {
  switch (plane) {
    case 152:
      alert("A small two-place airplane for flight training");
      break;
    …
    default:
      alert("Error in JavaScript function planeChoice");
      break;
  }
}
```

radio_click2.html

```
<form id = "myForm"  action = "">
     <label> <input type = "radio"  name = "planeButton"
                    value = "152" />
       Model 152 </label> <br />
       <label> <input type = "radio"  name = "planeButton"
                    value = "172" />
       Model 172 (Skyhawk) </label>
       <br />
...
       <script type = "text/javascript"  src = "radio_click2r.js" >
   </script>
```

radio_click2r.js

```
// radio_click2r.js
//   The event registering code for radio_click2

var dom = document.getElementById("myForm");
dom.elements[0].onclick = planeChoice;
dom.elements[1].onclick = planeChoice;
dom.elements[2].onclick = planeChoice;
dom.elements[3].onclick = planeChoice;
```

radio_click3r.js

```
// radio_click2r.js
//   The event registering code for radio_click2

var dom = document.getElementById("myForm");
dom.elements[0].addEventListener("click", planeChoice);
dom.elements[1].addEventListener("click", planeChoice);
dom.elements[2].addEventListener("click", planeChoice);
dom.elements[3].addEventListener("click", planeChoice);
```

radio_click2.js

```
function planeChoice () {
  var dom = document.getElementById("myForm");
  for (var index = 0; index < dom.planeButton.length;
       index++) {
    if (dom.planeButton[index].checked) {
      var plane = dom.planeButton[index].value;
      break; }}
  switch (plane) {
    case "152":
      alert("A small two-place airplane for flight training");
      break;
    case "172":
      alert("The smaller of two four-place airplanes");
      break;
```

iClicker question

- SHOW `radio_click.html`, `radio_click.js`
- What is the method used in the above code to register the event handler?

Answer: A

  A. Assign the event handler to attribute
  B. Assign the event handler to element property
  C. None of the above

- Does the event handler take inputs?
  A. Yes. It is the value of attribute "value"
  B. Yes. It is the value of attribute "name"
  C. No. The parameter list is empty
  D. None of the above

Answer: A

iClicker question

- SHOW `radio_click2.html`, `radio_click2.js`
- What is the method used in the above code to register the event handler?

Answer: B

  A. Assign the event handler to attribute
  B. Assign the event handler to element property
  C. None of the above

- How does the value of attribute "value" is passed to the event handler?

  A. Passed as a parameter of the event handler
  B. Obtained by checking the value of the selected radio button
  C. Known by itself
  D. None of the above

Answer: B

Assign handlers to element properties

- The disadvantage of specifying handlers by assigning them to element properties is that there is no way to use parameters

- The advantages of specifying handlers by assigning them to event properties are:
  1. It is good to keep HTML and JavaScript separate
  2. The handler could be changed during use

Handling Events from Textbox and Password Elements

- The focus event
  - See forced blur()

SHOW `nochange.html` & `nochange.js`

iClicker question

- SHOW `nochange.html` & `nochange.js`
- What if any of the quantity values is not a number?

A. The "Total Cost" text field is empty
B. The error console shows an error message
C. Both A and B
D. The "Total Cost" text field shows undefined
E. The "Total Cost" text field shows NaN

Answer: E

Validate Inputs

- Things that must be done:
  1. Detect the error and produce an `alert` message
  2. Put the element in focus (the `focus` function)
  3. Select the element, if there is a value (the `select` function)
  4. The handler return `false`, if not valid

SHOW `pswd_chk.html` & `pswd_chk.js`

Validation using patterns

- *Another Example* – Checking the format of a name and phone number

→ **SHOW `validator.html`, `validator.js`, and `validatorr.js`**

- The event handler will be triggered by the `change` event of the text boxes for the name and phone number
- If an error is found in either, an `alert` message is produced and both `focus` and `select` are called on the text box element

Useful links

- HTML standard events

(http://www.w3schools.com/TAGS/ref_eventattributes.asp)

- JavaScript object reference (http://www.w3schools.com/jsref/default.asp)
- W3C Document Object Model Events (https://www.w3.org/TR/2007/WD-DOM-Level-3-Events-20071221/events.html)

The End