1 Handling Events

JavaScript

2 JavaScript Events (Page 1)

- An HTML event is “something interesting” that happens to an HTML element
- Can include:
  - Web document completes its loading
  - The user moves the mouse over a hyperlink or some other object
  - The user clicks the mouse on a button or other object
  - The user strikes a key on the keyboard
- All GUI (graphical user interface) applications behave this way

3 JavaScript Events (Page 2)

- A common way of dealing with these events is to use an *event attribute* of the HTML object and assign the function name as its *value*
- Format:

  <*htmlElement* *eventType*="*functionName*()">
- Example:

```
<button onclick="getTime()">
```

4 The document Object

- Represents the Web page displayed in a browser
- Contains all Web page elements

5 The getElementById() Method

- A method of the document object that returns the first element in a Web document with a matching ID attribute value
- Format:

  document.getElementById("*indentifier*")
  - The *indentifier* is the value of the ID attribute of an HTML element
- Example:

```
document.getElementById("time").innerHTML = time.toLocaleTimeString();
```

6 The innerHTML property

- The innerHTML property sets or returns the HTML content (inner HTML) of an element
  - The content between an opening and closing tag such as <p> or one of the heading tags
- Format:

  htmlElement.innerHTML
- Example:

```
document.getElementById("time").innerHTML = time.toLocaleTimeString();
```

7 □ The <button> Tag (Page 1)

- The <button> tag defines a clickable button
- Inside a <button> element can put content like text or images which appear on the button
  - ▫ This is the primary difference between this element and buttons created with the <input> element

8 □ The <button> Tag (Page 2)

- Format:

  <button type="button">*Text or image on the button*</button>
- Example:

  <button type="button">Click me</button>
  - ▫ *Tip*: Always specify the type attribute for a <button> (different browsers have different default types for the <button> element)

9 □ The onclick Event

- The event that occurs when an element is clicked
- Often used with a button but can be applied to most HTML elements
- Format:

  <*htmlElement* onclick="*functionName*()">
- Example:

  <button onclick="getTime()">

10 □ Try It Out

- events1.htm

11 □ Assigning Code to Event Attributes

- Alternately one or more JavaScript statements can be assigned as the *event attribute* value
- Format:

  <*htmlElement eventType*="*javaScript statement*">
- Example:

  <button onclick="getElementById('time').innerHTML = Date()">
  - ▫ Single 'quotes' and double "quotes" can be used interchangeably in JavaScript

12 □ Try It Out

- events1a.htm

13 □ Anonymous Functions (Page 1)

- Functions may be assigned to event properties of an object in a JavaScript statement
  - ▫ Must follow the declaration of the object (HMTL element), so it needs to be in the <body> section inside a <script> block
- The function is considered *anonymous* since there is no name follow the keyword “function()”

14 **Anonymous Functions (Page 2)**

- Format:

  object.eventType = function() { *functionCode* };

  - The *functionCode* is inside a set of {braces} and can be either the body of an actual function or a function call to the name of a separately defined function

15 **Anonymous Functions (Page 3)**

- Example:

  ```
  document.getElementById("timeButton").onclick = function() { getTime() };
  ```

  - The getTime() function may be defined in a <script> block either in this or another <script> block

16 **Try It Out**

- events1b.htm

17 **The addEventListener() Method (Page 1)**

- The method to attaches (“registers”) an event handler function to a specified element
- Not supported in Internet Explorer 8 and earlier versions, and Opera 6.0 and earlier versions
- An event object is passed to the function as the first parameter

18 **The addEventListener() Method (Page 2)**

- Format:

  object.addEventListener(*eventType*, *functionName*);

- Example:

  ```
  timeButton.addEventListener("click", getTime);
  ```

  - The “on” prefix of the event is eliminated, e.g. just “click”
  - The parentheses following the function name are eliminated, e.g. just “getTime”

19 **Try It Out**

- events1c.htm

20 **The addEventListener() Method (Page 3)**

- The second argument as a complete function:

  ```
  timeButton.addEventListener("click",
      function getTime()
      {
          var time = new Date();
          document.getElementById("time").innerHTML = time.toLocaleTimeString();
      }
  )
  ```

21 **Try It Out**

- events1d.htm

22 □ The attachEvent() Method (Page 1)

- The method to attaches ("registers") an event handler function to a specified element
- Used for Internet Explorer 8 and earlier versions, and Opera 6.0 and earlier versions

23 □ The attachEvent() Method (Page 2)

- Format:

  object.attachEvent(*eventType*, *functionName*);
- Example:

  timeButton.attachEvent("click", getTime);
  - The "on" prefix of the event is eliminated, e.g. just "click"
  - The parentheses following the function name are eliminated, e.g. just "getTime"

24 □ Try It Out

- events1e.htm

25 □ The <input> Tag as a Text Field

- The <input> tag with attribute type="text" creates a text field (textbox) into which the user can type text
- Format:

  <input type="text">
- Example:

  <input type="text" id="fahrenheit">

26 □ The value Property

- The value property sets or returns the value of an element or option (the value sent to the server when the form is submitted)
  - For a text field the value property is the text currently contained within the textbox
- Format:

  object.value;
- Example:

  var fahrenheit = document.getElementById("fahrenheit").value;

27 □ Try It Out

- events2.htm

28 □ DOM Events

- The HTML DOM events allow JavaScript to "register" event handlers on elements in an HTML document
- Events normally are associated with functions so that the function does not execute until the event occurs, e.g. when a user clicks a button
- Tip: The event model was standardized by the W3C in DOM Level 2

29 Form Events (Page 1)

- Events triggered by actions inside an HTML form (applies to almost all HTML elements, but is most used in <form> elements):
  - onblur—occurs when an element loses the focus
  - onchange—occurs when the content of a form element, the selection, or the checked state have changed
    - For <input>, <keygen>, <select>, and <textarea>
  - onfocus—occurs when an element gets the focus

30 Form Events (Page 2)

- Events triggered by actions inside an HTML form (*con.*):
  - onfocusin—occurs when an element is just about to get the focus
  - onfocusout—occurs when an element is just about to lose the focus
  - oninput—occurs when an element gets user input
  - oninvalid—occurs when an element is invalid
  - onreset—occurs when a form is reset

31 Form Events (Page 3)

- Events triggered by actions inside an HTML form (*con.*):
  - onsearch—occurs when the user writes something in a search field
    - For <input="search“>
  - onselect—occurs after the user selects some text
    - For <input> and <textarea>
  - onsubmit—occurs when a form is submitted

32 Creating a Selection List (Page 1)

- The <select> block creates a scrollable menu (drop-down list or list box) used to select from a list of choices
- The list is inserted into the menu within a series of option blocks

33 Creating a Selection List (Page 2)

- Format:

```
<select  id="identifier" attributes>
   <option  value="value1">text1</option>
   <option  value="value2">text2</option>
          ...
</select>
```

  - The value attribute *value* of the option selected is the value for the <select> and is assigned to the name (variable) property when the form is submitted
  - The text are values displayed in the selection list

34 Creating a Selection List (Page 3)

- Example:

```
<select id="carSelect">
   <option value="Ford">Ford</option>
   <option value="Honda">Honda</option>
   <option value="Toyota">Toyota</option>
   <option value="Chevy">Chevy</option>
   <option value="Mercedes">Mercedes</option>
</select>
```

35 □ Try It Out

- events3.htm

36 □ The style Object

- Returns the collection of CSS “style” properties for an HTML element
- Each style property may be *set* or *retrieved*
- Format:

  object.style.*cssProperty*
- Example:

  ```
  document.getElementById("inputBox").style.backgroundColor = "red";
  ```

37 □ Try It Out

- events4.htm

38 □ Window Events (Page 1)

- Represent events related to browser itself:
  - onload—occurs when a document and all its resources are fully loaded and displayed (the most important of all window events)
    - Often used inside the <body> to execute a script once all content is completely loaded
    - Can be used to check visitor’s browser type (loads proper version of Web page accordingly)
    - Also can be used to deal with cookies

39 □ Window Events (Page 2)

- Represent events related to browser itself (*con.*):
  - onunload—occurs when user navigates away from current Web document
  - onbeforeunload—similar to unload but provides the user an opportunity to cancel closing browser window or navigation to another page (*more on this to follow*)

40 □ Window Events (Page 3)

- Represent events related to browser itself (*con.*):
  - onreadstatechange—similar to load but before external resources are fully loaded
  - onresize—occurs when user resizes browser window
  - onscroll—occurs when user scrolls up or down in the browser window’s scroll channel

  - Additionally onfocus and onblur from the form events occur when the browser window receives or loses keyboard focus from the operating system

41 □ The onbeforeunload Event (Page 1)

- The event occurs when the Web page is about to be unloaded and can *cancel* unloading
- Unlike the unload event, it displays a dialog window that gives the user an opportunity to cancel any one of the following:
  - Close the browser window
  - Reload the browser window
  - Navigation to another Web page

42 □ The onbeforeunload Event (Page 2)

- The standard message in the dialog window is something like:
  - "Are you sure you want to leave this page?"
  - This message cannot be removed
  - An additional custom message can be added by the programmer as well

43 □ The onbeforeunload Event (Page 3)

- Format:

  object.onbeforeunload=function() { return *functionCode* };
  - The keyword return allows the function to return a value which in this case is Boolean (if false the Web document will not unload)
- Example:

```
document.getElementById("webDocument").onbeforeunload = function() { return
  unloadFunction() };
```

44 □ The onbeforeunload Event (Page 4)

- The event handler function requires a string return value that is part of the prompt
  - Specifically the custom message that is displayed in the dialog window along with the standard message
- Example:

```
function unloadFunction()
{
   return "Click one of the buttons below";
}
```

45 □ The navigator Object (Page 1)

- The window.navigator object includes both methods and properties with information about a visitor's browser
- May be written without the window prefix, e.g.:
  - navigator.appName

46 □ The navigator Object (Page 2)

- Some navigator object members
  - navigator.appName—property that represents the “app name” of the browser
    - IE11, Chrome, Firefox, and Safari return appName “Netscape”
  - navigator.appCodeName—property that represents the “app code name” of the browser
    - Chrome, Firefox, IE, Safari, and Opera return appCodeName “Mazilla”

47 □ The navigator Object (Page 3)

- Some navigator object members (*con.*):
  - navigator.product—property that represents the engine name of the browser
  - navigator.appVersion—property that represents version information about the browser
  - navigator.userAgent—property that represents additional version information about the browser

48 □ The navigator Object (Page 4)

- Some navigator object members (*con.*):
  - navigator.platform—property that represents operating system (platform) of the browser
  - navigator.language—property that represents the browser’ language

49 □ The navigator Object (Page 5)

- Some navigator object members (*con.*):
  - navigator.cookieEnabled—property that represents a Boolean value (true or false) indicating if cookies are enabled for the browser
  - navigator.javaEnabled()—method that returns a Boolean value (true or false) indicating if Java is enabled for the browser

50 □ Try It Out

- events5.htm

51 □ Mouse Events (Page 1)

- Events triggered by a mouse action:
  - onclick—occurs when the user clicks on an element
  - oncontextmenu—occurs when the user right-clicks on an element to open a context menu
  - ondblclick—occurs when the user double-clicks on an element
  - onmousedown—occurs when the user presses a mouse button over an element

52 □ Mouse Events (Page 2)

- Events triggered by a mouse action (*con.*):
  - onmouseenter—occurs when the pointer is moved onto an element
  - onmouseleave—occurs when the pointer is moved out of an element
  - onmousemove—occurs when the pointer is moving while it is over an element

  - onmouseover—occurs when the pointer is moved onto an element, or onto one of its children

53 □ Mouse Events (Page 3)

- Events triggered by a mouse action (*con.*):
  - onmouseout—occurs when a user moves the mouse pointer out of an element, or out of one of its children
  - onmouseup—occurs when a user releases a mouse button over an element

54 □ Try It Out

- events6.htm

55 □ Key Events

- Events triggered by pressing and releasing keys on the keyboard
  - onkeydown—occurs when the user is pressing a key (key is down)
  - onkeypress—occurs when the user presses a key (a complete press and release)
  - onkeyup—occurs when the user releases a key

56 □ The event Object (Page 1)

- Every event call has a built-in event object as which can be passed as its first parameter
- Format:

  object.event = function() { *functionName*(event) }
  - The event argument is a programmer-defined name
- Example:

  document.getElementById("numberInput").onkeypress = function() {
  numbersOnly(event) }

57 □ The event Object (Page 2)

- To use the event object, define it as a parameter in the called function header
  - As before the event parameter is a programmer-defined name
- Format:

  function *functionName*(event)
  { …
- Example:

  function numbersOnly(event)
  { …

58 □ The event Object (Page 3)

- The properties for an event object are contingent on which is the event that has occurred
- Some objects for the onkeypress event are:
  - event.ctrlKey—returns a boolean value indicating if the <Control> key was down when the event occurred

  - event.charCode—returns a number which is the Unicode value of the key that was pressed
  - event.keyCode—returns a number which identifies the “unmodified” value of the key that was pressed
  - event.repeat—returns a boolean value indicating if the key is being held down and automatically repeating

59 □ **The event Object (Page 4)**

- A summary of most event object *properties* and *methods* can be found at:
  - https://developer.mozilla.org/en-US/docs/Web/API/Event

60 □ **KeyboardEvent charCode Property**

- The charCode property returns the Unicode value (number) of the key that triggered the onkeypress event
- Format:

  event.charCode
- Example:

  if (event.charCode == 48) { …
  - E.g. “Is it the digit 1?”

61 □ **KeyboardEvent keyCode Property**

- The keyCode property returns a number which identifies the “unmodified” value of the pressed key that triggered the onkeypress event
- Format:

  event.keyCode
- Example:

  if (event.keyCode == 48) { …
  - E.g. “Is it the digit 1?”

62 □ **Try It Out**

- events7.htm

63 □ **DOM Events (Page 1)**

- The DOM Level 3 Events specification adds new elements and modifies other functionality:
  - Deprecates a number of event types that were defined by Level 2 but never widely implemented (e.g. DOMActivate, DOMFocusIn, and DOMNodeInserted,)
  - Standardizes the focusin and focusout events as bubbling alternatives to the focus and blur events
  - Standardizes mouseenter and mouseleave events as nonbubbling alternatives to mouseover and mouseout

64 □ **DOM Events (Page 2)**

- The DOM Level 3 Events specification adds new elements and modifies other functionality (*con.*):
  - Standardized support for two-dimensional mouse wheels via the wheelevent
  - Better support for text input events with a textinput event and with a new KeyboardEvent object that is passed as the argument to handlers for keydown, keyup, and keypress events (deprecated keypress in favor of textinput)

65 □ **DOM Events (Page 3)**

- The DOM Level 3 Events specification adds new elements and modifies other functionality (*con.*):
  - Simplifies keydown, keyup, and keypress events by adding new key and char properties to event object
    - Both of these properties are strings.
    - For key events that generate printable characters, key and char will be equal to the generated text
    - For control keys, the key property will be a string like "Enter", "Delete" or "Left" that identifies the key

66 □ **HTML5 Events (Page 1)**

- Inclusion of <audio> and <video> elements for playing sound and video with long list of events:
  - canplay,loadeddata, playing, stalled, canplaythrough, loadedmetadata, progress, suspend, durationchange, loadstart, ratechange, timeupdate, emptied, pause, seeked, volumechange, ended, play, seeking, waiting

67 □ **HTML5 Events (Page 2)**

- Drag-and-drop API allows JavaScript apps to participate in OS-based drag-and-drop operations:
  - dragstart, drag, dragend, dragenter, dragover, dragleave, drop
- Defines a lot of new features for HTML forms including standardizing the form input event and defining a form validation mechanism

68 □ **HTML5 Events (Page 3)**

- Support for offline web applications that can be installed locally in an application cache
  - offline, online, cached, checking, downloading, error, noupdate, obsolete, progress, updateready, message

69 □

70 □ **Touchscreen and Mobile Events**

- (*For some other time*)