

CMSC388A

Web Application Development with JavaScript



Events

Department of Computer Science
University of MD, College Park

Slides material developed by Ilchul Yoon, Nelson Padua-Perez

Arrow Functions

- Alternative to anonymous functions
 - “Lambda Expressions”
- **Rely on the => operator**
- **Format**
 - **Parameters => code**
 - Parenthesis for parameters is only required if the function has no parameters or two or more parameters. A function with one parameter does not require parenthesis surrounding the parameters
 - If the code is a single expression, no curly braces or return statements are required
- **Example:** ArrowFunc.html

Events

- **Event:** Notification that something has occurred
- Example situations that make the web browser generate an event
 - Browser finishes loading a document
 - When the user clicks on a button
 - When the user moves the mouse
 - Others
- **Event handler** (also known as **event listener**)
 - JavaScript function or code fragment that is executed when a particular event occurs
- **Event handler registration**
 - **Associating an event handler with a particular event**

Event-driven Programming

- **Normal (control flow-based) Programming**
 - Approach
 - » Start at main()
 - » Continue until the end of the program or exit()
- **Event-driven Programming**
 - Start at main()
 - Register event handlers
 - Await events & perform associated computation
- **GUIs (Graphical User Interfaces)**
 - Example of event-driven software

Event Handler Attributes for Most HTML

- Mouse Related
 - **onclick** - mouse button is pressed and released
 - **ondblclick** - mouse button is doubled-click over the element
 - **onmousedown** - the mouse is pressed down while the cursor is over the element
 - **onmouseup** - the mouse is released while the cursor is over the element
 - **onmouseenter** - mouse moves onto the element
 - **onmouseover** - mouse pointer enters into an element and its child elements
 - **onmouseout** - mouse moves off an element
 - **onmousemove** - mouse pointer is moved over an element

Accessing Data From Text Fields

- We can access data in text fields by first accessing the DOM element using:
 - `document.getElementById("elementId")`
 - `document.querySelector("#elementId")`
- We can then access the value using **value**
- **Retrieving the value of a text field**

`let login = document.getElementById("loginId").value;`

or

`let login = document.querySelector("#loginId").value`

Associating Function with Event

- We can define a function (callback) for an event associated with an element by first accessing the DOM element using:
 - `document.getElementById("elementId")` or `document.querySelector("#elementId")`and then assigning a callback function to the event
- **Defining which function to call when an element (e.g., button) is clicked on**
`document.getElementById("processButtonId").onclick = callback;`
- **Another way to associate a function is to use `addEventListener`**
 - Allows several events to be added`document.getElementById("displayValueButtonId").addEventListener("click", callback);`
- **Another way is to set the `onclick` property in the element**
 - `<input type="button" value="Display School Name" onclick="displaySchoolName()" />`

Associating Function with Event

- **Example:** AssociateButtonWithFunction.html
- **Example:** GetValueInTextField.html, UpdateValueInTextfield.html
- **Example:** GetValueOnChange.html

Form Data Access (Attributes)

- **We can access/modify attributes using `getAttribute()/setAttribute()`**

```
let imageElement = document.getElementById("myImage");  
let imageName = imageElement.getAttribute("src");  
imageElement.setAttribute("src", "imageFile.jpg");
```
- **You can access and modify the attribute directly**

```
alert(document.querySelector("#myImage").src)  
document.querySelector("#myImage").src = "testudo1.jpg"
```
- **Example:** GetSetAttribute1.html, GetSetAttribute2.html
 - Notice the difference in the path returned by GetSetAttribute1.html and GetSetAttribute2.html

setTimeout/clearInterval/setInterval

- You can execute code at some point in the future use `setTimeout`
- You can execute code at a particular interval using **`setInterval`**
- **`setInterval`** returns an id that **`clearInterval`** uses to stop execution
- **Example:** `SetTimeout.html`
- **Example:** `Animation.html`

Modifying a Page Area Using innerHTML

- `document.writeln()` - replaces the whole page after a page has been rendered. What if you want to update an area of the current page?
- **Example:** InnerHTML.html

Loading JavaScript from a File

- **Example:** LoadingJSFromFile.html (e.g., `<script src="code.js"></script>`)
- **defer** attribute - script will be downloaded in parallel with parsing the page and executed after the page has been parsed
 - There is no need to place your `<script src="..">` at the end of the HTML file
- **async** attribute - script will be downloaded in parallel with parsing the page and executed as soon as it is available (before parsing finishes)
- If both **defer** or **async** are missing
 - Script is downloaded and executed immediately, **blocking the page parsing until the script finishes**
- **Reference**
 - https://www.w3schools.com/tags/att_script_defer.asp