CM1040 Web Development Week 15 Lecture Note

Notebook:	Web Development		
Created:	2020-10-13 4:07 PM	**Updated:**	2020-12-14 5:11 PM
Author:	SUKHJIT MANN		

Cornell Notes	**Topic:** React to the User	Course: BSc Computer Science Class: Web Development CM1040[Lecture] Date: December 14, 2020

Essential Question:

How does JavaScript reacts to User Input?

Questions/Cues:

- What are events?
- What is polling
- What is event-driven model?
- What is event flow?
- What are event listeners/handlers?

Notes

- Events = Events are actions that are triggered in the browser when something happens
- Polling = The OS check for an action or key press and puts in a queue to be accessed at a later time by any program that would need to check for user inputs. The problem with this approach is that when it was used the user would press or do an action and have to wait for the program to check the queue, making the whole process feel a little less responsive.
- Event-driven model = The system (the browser in this case) notifies the code (JavaScript) when an event happens
- Event flow = the order in which events are handled on the page, in other words it's the way the browser decides whether the link, <a> tag should receive the event before or after the body tag in the case where you clicking a link on a page
    - every event is on event when using event handlers
    - By using event handlers, you could run into errors where the HTML is loaded before the code for the handler is executed, producing an error

onlineonclick.html

```
<html>
    <script>
        window.onload = function(){
            var btn = document.getElementById("btn");
            btn.onclick = function(){
                alert(this.id);
            }
        }
    </script>
    <body>
        <button id="btn">Click me</button>
    </body>
</html>
```

- This can be avoided by enclosing the event handler in a try catch block, so that no errors are produced and so they quietly fail in the background as shown below

```
<input type="button" value="ClickMe" onclick="try{showMessage();}catch(ex){}/>
```

- it is advised to use event listeners instead of event handlers to avoid timing errors
- Event listeners work in almost the same way as handlers in that we still need to attach them to an element in the DOM and we need to assign a function in JS to react to the event triggered by the browser. Also we still need to identify the event type.
  - However with event listeners we can assign multiple functions to the same DOM element
- To create an event listener, we usually use an addEventListener method which accepts three arguments: the event name or type, the event listener function, and the Boolean value which indicates whether the event listener should be called during the capture phase or the bubble phase
  - The capture phase calls the event handler for the outermost element first and then works its way inwards while the bubble phase queries the innermost element first and works its way outwards
    - If we want to invoke the bubble phase then the Boolean should be false. If we want the capture phase, then we set the Boolean to true

test.html

```html
<html>
    <body>
        <button id="btn">Click me</button>

        <script>
            var btn = document.getElementById("btn");

            function once(){
                btn.textContent = "You have clicked me"
            }

            btn.addEventListener("click", once);

        </script>
    </body>
</html>
```

- the addEventListener method is supported in browsers IE 8 and earlier, instead they have a similar method called attach event, with the difference between the two being that attach event takes two arguments instead of three because bubble capture is the only option in these browsers.
- functions called inside the addEventListener method are called event listeners not functions and they don't have parentheses to ensure that they are not called as a line of code is loaded
- unintentional/indirect events can be attached to the browser window or a particular element like the scroll event

scroll.html

```html
<html>
    <body>
        <div id="progress"></div>
    </body>

    <footer>
        <style>
            body {
                height:2000px;
                background:salmon;
            }
            #progress {
                border-bottom: 2px solid green;
                width:0;
                position:fixed;
                top:0; left:0;
            }

        </style>
        <script>
            var bar = document.getElementById("progress");
            window.addEventListener("scroll", function(){
                bar.style.width = pageYOffset;
            })
        </script>
    </footer>
</html>
```

Summary

In this week, we learned about JavaScript events, moreover event handlers and listeners.