Intro to JS - Part 2 - Day 3: Sept 21

Windows and document events

Load

Fires on the window object. Code is executed after page load.

```
window.addEventListener("load", function () { console.log("Page loaded successfully!") })
```

DOMContentLoaded Event

Fires when the initial HTML document has been completely parsed, but before any stylesheets, images, etc. Used to trigger the execution of scripts that need to run once the DOM is ready, but not necessarily its additional resources.

```
document.addEventListener("DOMContentLoaded", function () {
  console.log("DOM content loaded successfully!")})
```

Readystatechange

Fires on the document object. Code is executed when the **readystate** attribute of the document changes (loading, interactive, and complete).

```
document.addEventListener("readystatechange", function () {
console.log("Ready state changed: " + document.readyState) })
```

Mouse events

Triggered when a user interacts with the mouse.

Click

```
const myButton = document.getElementById("my-button")
myButton.addEventListener("click", function () {
  alert("Button clicked!")})
```

Mouseover

```
const myDiv = document.getElementById("my-div")
myDiv.addEventListener("mouseover", function () {
```

```
  myDiv.style.backgroundColor = "blue"})
```

Mouseout

```
const myDiv = document.getElementById("my-div")
myDiv.addEventListener("mouseout", function () {
myDiv.style.backgroundColor = "green" })
```

Keyboard events

Keydown

```
const myInput = document.getElementById("my-input")
const output = document.getElementById("output")
myInput.addEventListener("keydown", function (event) { output.textContent
= `You pressed down the ${event.key} key!` })
```

Keyup

```
const myInput = document.getElementById("my-input")
const output = document.getElementById("output")
myInput.addEventListener("keyup", function (event) { output.textContent =
`You released the ${event.key} key!` })
```

Keypress

```
const myInput = document.getElementById("my-input")
const output = document.getElementById("output")
myInput.addEventListener("keypress", function (event) { output.textContent
= `You pressed and released the ${event.key} key!` })
```

Input events

Fires when a user interacts with an input field.

Change

```
const mySelect = document.getElementById("my-select")
const output = document.getElementById("output")
mySelect.addEventListener("change", function (event) { output.textContent
= `You selected ${event.target.value}!` })
```

Focus

```
const myInput = document.getElementById("my-input")
const output = document.getElementById("output")
myInput.addEventListener("focus", function (event) { output.textContent =
"The input is in focus!" })
```

Blur (the opposite of focus)

```
const myInput = document.getElementById("my-input")
const output = document.getElementById("output")
myInput.addEventListener("blur", function (event) { output.textContent =
"The input no longer has focus!" })
```

Animation events

Animationstart

```
const box = document.getElementById("box")
const output = document.getElementById("output")
box.addEventListener("animationstart", function (event) {
output.textContent = "The animation has started!" })
```

Animationend

```
const box = document.getElementById("box")
const output = document.getElementById("output")
box.addEventListener("animationend", function (event) { output.textContent
= "The animation has ended!" })
```

Animationiteration

```
const box = document.getElementById("box")
const output = document.getElementById("output") let iterationCount = 0
box.addEventListener("animationiteration", function (event) {
iterationCount++ output.textContent = `The animation has looped
${iterationCount} times!` })
```

Bubbling

Bubbling is a technique where an event is first handled by the innermost element and then propagated up to the outer elements. This is the default behavior in JavaScript.

Capturing (opposite of bubbling)

Rarely used. To use capturing, pass a third argument to the **addEventListener()** method with the value of **true**. The event is first handled by the outermost element and then propagated down to the inner elements.

```
element.addEventListener(eventType, eventHandler, true);
```

Delegation

Attach a single event listener to a parent element that will handle events triggered by its child elements. Useful when many child elements need to trigger the same event listener.

```
const myList = document.getElementById("myList")
myList.addEventListener("click", function (event) { if
(event.target.tagName === "LI") {
event.target.classList.toggle("selected") } })
```

Advanced DOM

Creating elements

createElement method

To create an HTML element dynamically in JavaScript

```
const newHeadingElement = document.createElement('h2')
```

createTextNode

To create a text node that can be added to an HTML element

createAttribute

To create an attribute for an HTML element.

appendchild

To add a child element to an HTML element.

insertBefore

To insert a new child element before an existing child element of an HTML element

cloneNode

To create a copy of an HTML element

Removing elements

removeChild

To remove a child node from the DOM

replaceChild

To replace an existing child node with a new child node

DOM Traversal

Parent element & Parent node

parentElement returns only an element node.
parentNode can return any node type.

Children & Child nodes

The **children** property returns only element nodes.
The **childNodes** property can return any node type.

hasChildNotes

To check if an element has any child nodes. Returns “true” or “false”

firstChild and firstElementChild

The **firstChild** property can return any node type.
The **firstElementChild** property returns only an element node.

lastChild and lastElementChild

The **lastChild** property can return any node type.
The **lastElementChild** property returns only an element node.

nextSibling and nextElementSibling

The **nextSibling** property can return any node type.
The **nextElementSibling** property returns only an element node.

previousSibling and previousElementSibling

The **previousSibling** property can return any node type.
The **previousElementSibling** property returns only an element node.