Chapter 7: Events

“DOM is an interface that allows you to use JavaScript to interact with a web page”

“they are what provides the link between the web page and user interactions.”

Event Listeners

“are like setting a notification to alert you when something happens”

“the event listener will let it know when the event happens, and the program can then respond appropriately”
“blocking approach to programming because checking for the click is blocking the rest of the program from running.”

“non-blocking approach that uses event listeners to listen out for any clicks on the page. Every time the page is clicked, a callback function will be called”

“following code can be used to attach an event listener to the document that fires when the user clicks anywhere on the page:

document.body.addEventListener("click", doSomething);”

“It will call the function doSomething() when any part of the page is clicked on”

Inline Event Handlers

example that adds onclick event handler to a paragraph element

```
<p onclick="console.log('You Clicked Me!')">Click Me</p>
```

but it isn’t recommended for a number of reasons:

The JavaScript code is mixed up with the HTML markup, breaking the concept of unobtrusive JavaScript, which advocates that JavaScript code should be kept out of the HTML.

Only one event handler for each event-type can be attached to an element.

The code for the event handlers is hidden away in the markup, making it difficult to find where these events have been declared.

The JavaScript code has to be entered in a string, so you need to be careful when using apostrophes and quote marks.”

inline event handlers are best avoided

Older Event Handlers

the following example would cause a message to be logged to the console when the page is clicked:

```
document.onclick = function() { console.log('You clicked on the page'); }
```

it keeps to JavaScript out of the HTML markup

Using Event Listeners

`addEventListeners()` method is called on a node object, the node to which the event listener is being applied

`addEventListener()` methods can also be called without a node, in which case it is applied to the global object, usually the whole browser window

first parameter is the type of event, & the second is a callback function that is invoked when the event occurs

The Event Object

whenever an event handler is triggered by an event, the callback function is called

Types of Events

`type` property returns the type of event that occurs

The Event Target

`target` property returns a reference to the node that fired the event

Coordinates of Events

"There are a variety of ways to find the position of where a mouse event occurs.

The screenX and screenY properties show the number of pixels from the left and top of the screen respectively where the event took place"

Types of Events

Mouse Events

mouseup
mousedown
dblclick
mouseover

"mouseover event occurs when the mouse pointer is placed over the element to which the event listener is attached"

mouseout

"mouseout event occurs when the mouse pointer moves away from an element"

mouseover

"mousemove event occurs whenever the mouse moves. It will only occur while the cursor is over the element to which it's applied."

Keyboard Events

keydown
keypress
keyup

"The keydown event occurs when a key is pressed and will continue to occur if the key is held down.

The keypress event occurs after a keydown event but before a keyup event. The keypress event only occurs for keys that produce character input (plus the 'Delete' key). This means that it's the most reliable way to find out the character that was pressed on the keyboard.

The keyup event occurs when a key is released"

"distinguish between a physical key on the keyboard and a character that appears on the screen. The keydown event is the action of pressing a key, whereas the keypress event is the action of a character being typed on the screen"

"Each of these keyboard events have an key property that returns the printed representation of the key that was pressed"

Modifier Keys

"Shift, Ctrl, Alt and meta (Cmd on Mac) will fire the keydown and keyup events, but not the keypress event as they don't produce any characters on the screen"

"shiftKey, ctrlKey, altKey, and metaKey are all properties of the event object and return true if the relevant key was held down"

Touch events

“touchstart event occurs when a user initially touches the surface”

“Be careful when using the touchstart event as it fires as soon as a user touches the screen. They may be touching the screen because they want to zoom in or swipe, and a touchstart event listener could prevent them from doing this.”

“touchend event occurs when a user stops touching the surface”

“touchmove event occurs after a user has touched the screen then moves around without leaving”
“touchenter event occurs when a user has already started touching the surface, but then passes over the element to which the event listener is attached”

“touchleave event occurs when the user is still touching the surface, but leaves the element to which the event listener is attached.”

“touchcancel event occurs when a touch event is interrupted”

Touch Event Properties

“touch event objects have a property called touches. This is a list of touch objects that represents all the touches taking place on that device. It has a length property that tells you how many touch points (usually the user's fingers, but could be a stylus) are in contact with the surface”

“touch.screenX and touch.screenY to find the coordinates of the touch point”
“touch.radiusX and touch.radiusY, which give an indication of the area covered by the touch”
“touch.force, which returns the amount of pressure being applied by the touch as a value between 0 and 1”

Removing Event Listeners

“An event listener can be removed using the removeEventListener() method”
“adds a click event listener to a paragraph element, but then removes it in the callback function named remove”

Stopping Default Behavior

“a form is submitted when the user clicks on the Submit button.” ← default behavior example
“preventDefault() is a method of the event object that can be used inside the callback function to stop the default behavior happening”

Event Propagation

“When you click on an element, you are actually clicking on all the elements it’s nested inside of”
“If you click on one of the <li> elements, you’re also clicking on the <ul>, <body> and <html> elements”
“An event is said to propagate as it moves from one element to another”
“Event propagation is the order that the events fire on each element”
“two forms of event propagation: bubbling and capturing”

“Bubbling is when the event fires on the element clicked on first, then bubbles up the document tree, firing an event on each parent element until it reaches the root node”

“Capturing starts by firing an event on the root element, then propagates downwards, firing an event on each child element until it reaches the target element that was clicked on.”

Bubbling

default behavior is bubbling

“The event then bubbles up to the parent <ul> element and displays a message in the console saying “Clicked on ul”. The event will continue to bubble all the way to the root HTML element, but nothing will happen because none of the other elements had event listeners attached to them.”

Capturing

“addEventListener() method has a third parameter, which is a boolean value that specifies whether capturing should be used or not”
defaults to false

“you might want events on outer elements to fire before any events fire on the element that was actually clicked on”

“you want the event to both capture and bubble, you must set a separate event handler for both cases”

“bubble phase can be stopped from occurring by adding the event.stopPropagation() method into the callback function. In”

Event Delegation

“Event delegation can be used to attach an event listener to a parent element in order to capture events that are triggered by its child elements”

“better way is to attach the event listener to the parent <ul> element, then use the target property to identify the element that was clicked on”

Chapter Summary

“Events occur when a user interacts with a web page.

An event listener is attached to an element, then invokes a callback function when the event occurs.

Events occur when a user interacts with a web page.

An event listener is attached to an element, then invokes a callback function when the event occurs.

The event object is passed to the callback function as an argument, and contains lots of properties and methods relating to the event.

There are many types of event, including mouse events, keyboard events, and touch events.

You can remove an event using the removeEventListener method.

The default behavior of elements can be prevented using the preventDefault() function.

Event propagation is the order the events fire on each element.

Event delegation is when an event listener is added to a parent element to capture events that happen to its children elements.”

Questions:

Why did DOM Levels stop @ 3?

Why teach us about inline event handlers if they are best avoided?