JavaScript For Web

Week 3, Lecture 5 - JavaScript in Web Development: Event

Instructor: Jason Xu

Today's Overview

- What is callback function
- What is events
- How to use event listeners
- Tutorial: Asssignment 2, Exercise 4

What is Callback Function

Callbacks are functions that are passed into another function as an argument(parameter). For example:

```
const fruits = ['Apple', 'Banana', 'Orange']

fruits.forEach((fruit) => {
  console.log(fruit)
})
```

This is the `forEach` array method. This method simply takes a callback function as its argument.

Below is what `forEach` definition might look like, notice it calls the `callback` function each time it loops over an item.

```
function myForEach(array, callback) {
  for (let i = 0; i < array.length; i++) {
    callback(array[i]) // This is when the callback function gets called, or executed
  }
}

// You would call it like this:
const myArry = [2, 3, 4, 2];
myForEach(myArry, (item) => {
  console.log(item + 2)
})
```

Check more examples

What are Events

Events are actions that occur on your webpage such as mouse-clicks or keypresses, and using JavaScript we can make our webpage listen and react to these events.

Common Events:

- `onchange`: An HTML element has been changed.
- `onclick`: The user clicks an HTML element.
- `onmouseover`: The user moves the mouse over an HTML element.
- `onmouseout`: The user moves the mouse away from an HTML element.
- `onkeydown`: The user pushes a keyboard key.
- `onload`: The browser has finished loading the page.

Today, we will focus on the `onclick` event. Check more events here.

How to use Events in JavaScript

```
<!-- the HTML file -->
<button id="btn">Click Me</button>
```

```
// the JS file
const btn = document.querySelector('#btn')
btn.onclick = () => alert("Hello World")
```

In this example, we create a button and set a function to `onclick` property. When user click the button, the function will be executed. We will only have one `onclick` property

What are Event Listeners

You can also attach event listeners to the DOM nodes/objects in your JavaScript. Event listeners are definitely the preferred method.

- The `addEventListener()` method attaches an event handler to the specified element.
- You can add many event handlers to one element.
- You can add many event handlers of the same type to one element, i.e two "click" events.

We can rewrite our previous example like this:

```
const btn = document.querySelector('#btn')

btn.addEventListener('click', () => {
  alert("Hello World")
})

btn.addEventListener('click', () => {
  console.log("Hello World")
})
```

Instead of set `onclick` property directly, `addEventListener()` is more flexible and powerful because we could have multiple event handlers of the same type of one element.

Event Listeners continued

Let's take look into this example in details, we are passing a callback function into `addEventListener()`:

```
btn.addEventListener('click', (event) => {
  alert("Hello World")
})
```

The `event` parameter in this arrow function is an object that reference the event itself. Within that object you have access to many useful properties and methods (functions that live inside an object) such as which mouse button or key was pressed, or information about the event's target - the DOM node that was clicked.

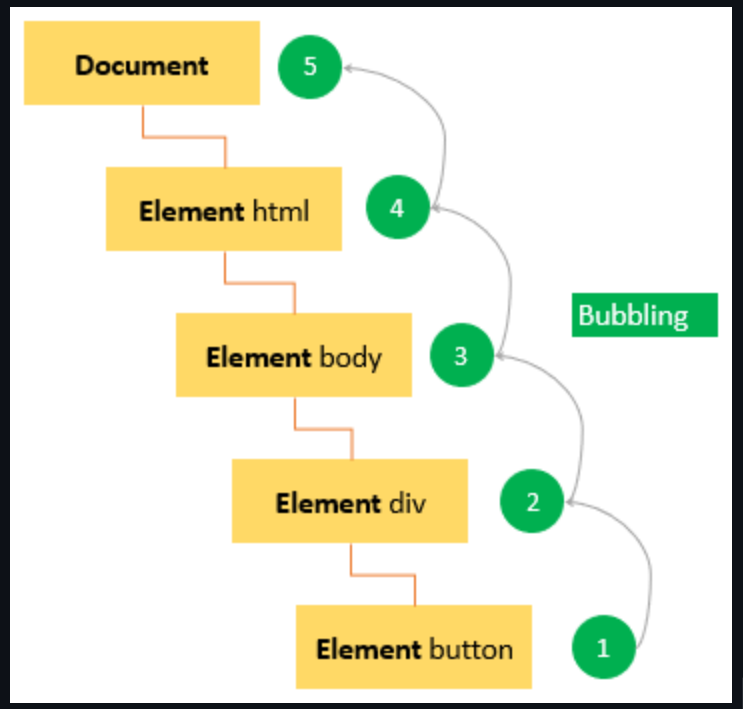
```
btn.addEventListener('click', (event) => {
  console.log(event.target)
})
```

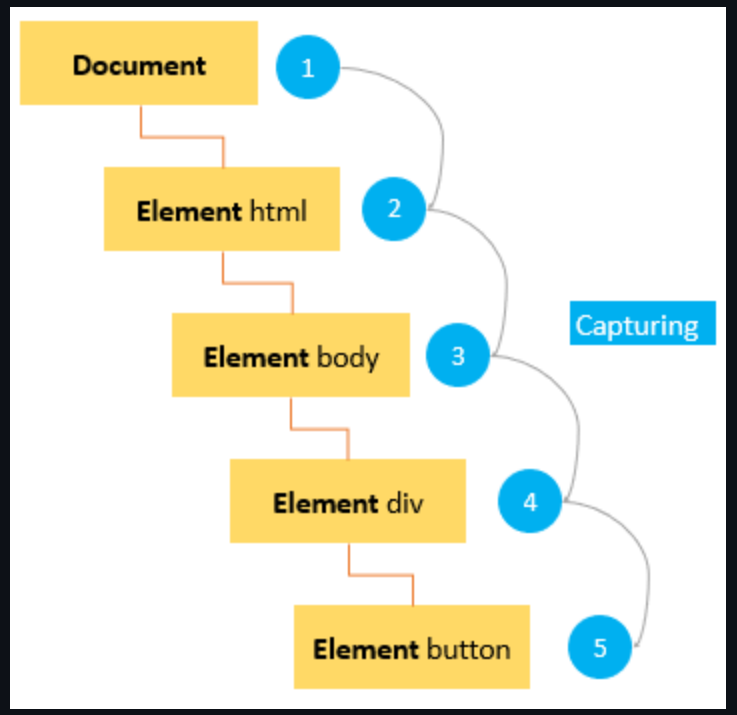
We can also try to modify the button using `event.target`:

```
btn.addEventListener('click', (event) => {
  event.target.style.background = 'blue'
})
```

Event Bubbling and Capturing

- In the event bubbling model, an event starts at the most specific element and then flows upward toward the least specific element.
- In the event capturing model, an event starts at the least specific element and flows downward toward the most specific element.
- The `stopPropagation()` method immediately stops the flow of an event through the DOM tree.
- Note that the `preventDefault()` method does not stop the event from bubbling up the DOM. And an event can be canceled when its `cancelable` property is `true`.





Video: Difference between "bubbling" and "capturing"

Web Dev Notes App: Button for adding new note

Step 1: create a default block in `html` file for adding new note field, including a text label, input and button.

```
<div>
  <label for="note">Enter a new note:</label>
  <input type="text" name="note" id="note">
  <button>Add new note</button>
</div>
```

Step 2: add a click event to this button.

- When user click the button, we create a note element which takes text from the input value and append to the note list.
- The new note should contain at least one character.

```
let input = document.querySelector('input')
let button = document.querySelector('button')

button.addEventListener('click', () => {
  const inputText = input.value
  input.value = ''

  if(inputText) {
    let newNote = document.createElement('li')
    newNote.textContent = inputText
    noteList.append(newNote)
  }

  input.focus()
})
```

Summary

- Callback function
- Events
- Event listeners
- Event Bubbling and Capturing

Thank you