Events

- Mouse over, click etc
- Created by certain activities associated with HTML elements
- **Registration** -> connecting handler to event

Event Handling

Three main ways to handle events

Inline event handlers

```
<button onclick="bgChange()"> Button </button>
```

Assigning element itself to be handler

```
const element = document.querySelector("#id")
element.on<event> = handler_function

// Examples
div.onclick = change;
div.onmouseover = inline_func(){ ... }
```

Separate event listener on that element

```
element.addEventListener(event, handler)
```

- These two parameters mandatory

Event Sources

Source	Event	Fires When...
Mouse	click	the mouse is clicked and released on an element
	dblclick	an element is clicked twice
	mousemove	every time a mouse pointer moves inside an element
	mouseover	every time a mouse pointer is placed over an element
Keyboard	keydown	when a key is pressed down
	keyup	when a key pressed is released
	keypress	when a key is pressed and released
Form	submit	a form is submitted
	reset	a form reset button is clicked
	focus	an input element is clicked and receives focus
	blur	an input element loses focus

Event Objects

The handler function for any event is called with a single argument: **event object**.

- Represents an event, has its own properties

```
<button id="myButton">Click Me</button>

<script>
  const button = document.getElementById('myButton');
  button.addEventListener('click', function(event) {
    console.log(event); // Accessing the event object
  });
</script>
```

Properties of the event object

Property	IE5-8 Equivalent	Specifies
target	srcElement	the target of the event (most specific element).
type	-	the name of event fired (without the on prefix)
altKey / shiftKey / ctrlKey / metaKey	-	true/false to signify if Alt Key or Shift Key or Ctrl Key or Meta Key was pressed
charCode	keyCode	Unicode character code of the pressed key
key	-	Key Character Name ('a' or 'F1' or 'CAPS LOCK')
button	-	Returns which mouse button was pressed
clientX, clientY / offsetX, offsetY / screenX, screenY	-	the coordinates of the mouse pointer when the event triggered, relative to, the current window / target element / screen

Event Propagation

A mechanism that defines how events propagate or travel through the DOM tree to arrive at its target

- Way of defining element order when an event occurs
  Eg: If two tags register a click event, which one goes first?

```
elem.addEventListener("event", func_ref, flag);

flag = true :=> Handler registered for Capturing phase
flag = false:=> Handler registered for Bubbling phase (default)
```

Bubbling

Innermost to outermost

Capturing

Outermost to innermost

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Event Capturing and Bubbling</title>
  <style>
    div, section, article {
      padding: 20px;
      margin: 10px;
      border: 2px solid black;
    }
    div { background-color: lightblue; }
    section { background-color: lightgreen; }
    article { background-color: lightcoral; }
  </style>
</head>
<body>
  <h1>Event Capturing and Bubbling Example</h1>
  <div id="div">
    Div Element
    <section id="section">
      Section Element
      <article id="article">
        Article Element
      </article>
    </section>
  </div>

  <script>
    // Add event listeners to the div, section, and article
    const div = document.querySelector("#div");
    const section = document.querySelector("#section");
    const article = document.querySelector("#article");

    // Event listener function
    function handleEvent(event) {
      console.log(
        `${event.currentTarget.tagName} received event during capturing phase.`
      );
    }

    function handleBubble(event) {
      console.log(
        `${event.currentTarget.tagName} received event during bubbling phase.`
      );
    }
```

```
    // Add event listeners for capturing phase
    div.addEventListener("click", handleEvent, true); // true for capturing
    section.addEventListener("click", handleEvent, true);
    article.addEventListener("click", handleEvent, true);

    // Add event listeners for bubbling phase
    div.addEventListener("click", handleBubble, false); // false for bubbling
    section.addEventListener("click", handleBubble, false);
    article.addEventListener("click", handleBubble, false);
  </script>
</body>
</html>
```