JavaScript Events

Understanding Event Handling in JavaScript

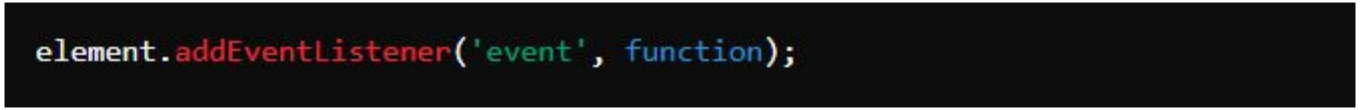
What are JavaScript Events?

Definition: Events are actions or occurrences that happen in the browser, such as a user clicking a button, typing in a field, or loading a page.

Common Events: Click, Mouseover, Keydown, Load, etc.

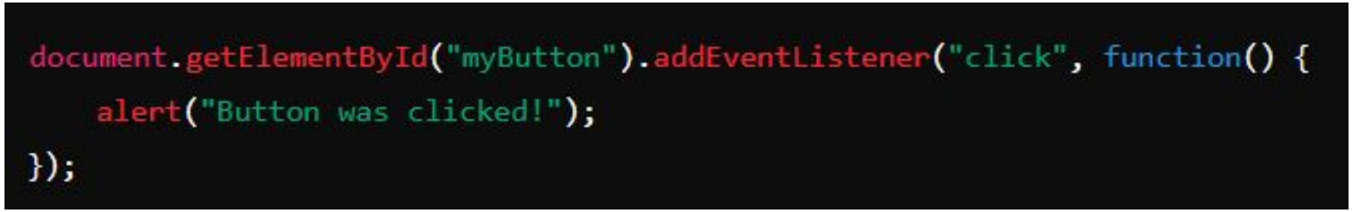
How to Add Event Listeners

Using `addEventListener :`



```
element.addEventListener('event', function);
```

`Example :`



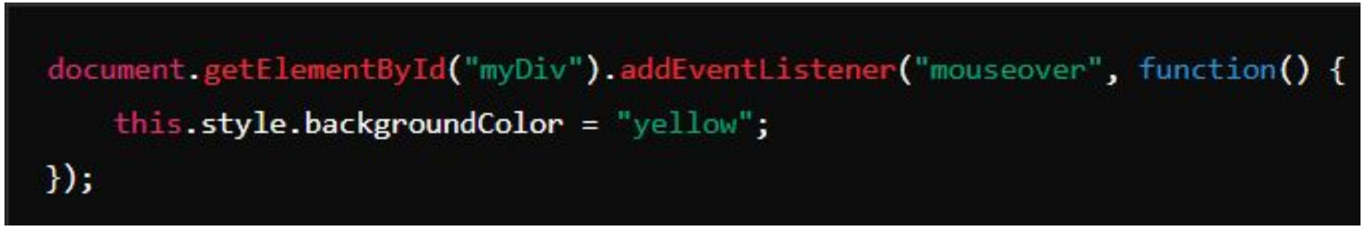
```
document.getElementById("myButton").addEventListener("click", function() {
    alert("Button was clicked!");
});
```

Types of Events

Mouse Events :

- `click`
- `dblclick`
- `mouseover`
- `mouseout`

Examples:

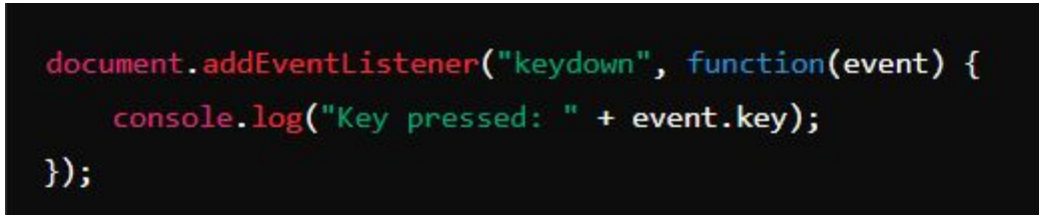


```
document.getElementById("myDiv").addEventListener("mouseover", function() {
    this.style.backgroundColor = "yellow";
});
```

Keyboard Events

- `keydown`
- `keyup`
- `keypress`

Examples :

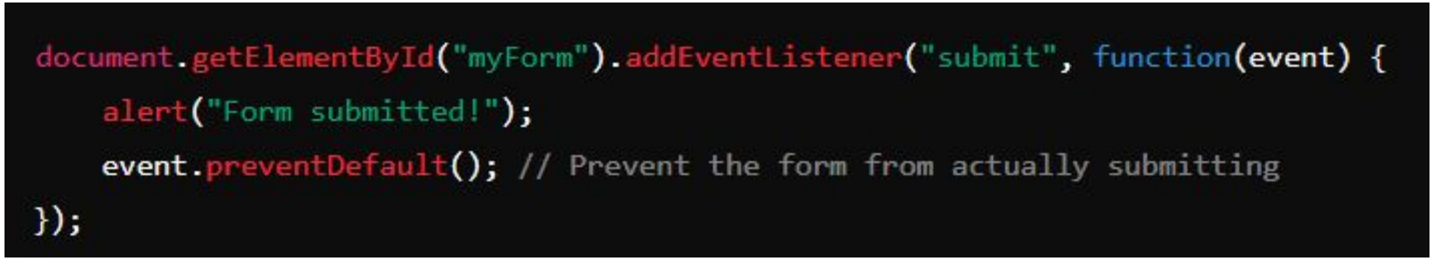


```
document.addEventListener("keydown", function(event) {
    console.log("Key pressed: " + event.key);
});
```

Form Events

- `submit`
- `focus`
- `blur`
- `change`

Examples:



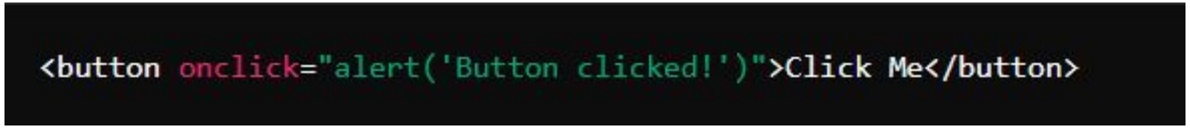
```
document.getElementById("myForm").addEventListener("submit", function(event) {
    alert("Form submitted!");
    event.preventDefault(); // Prevent the form from actually submitting
});
```

Event Handling

Inline Event Handling :

Definition: Directly assign an event to an HTML element.

Example :



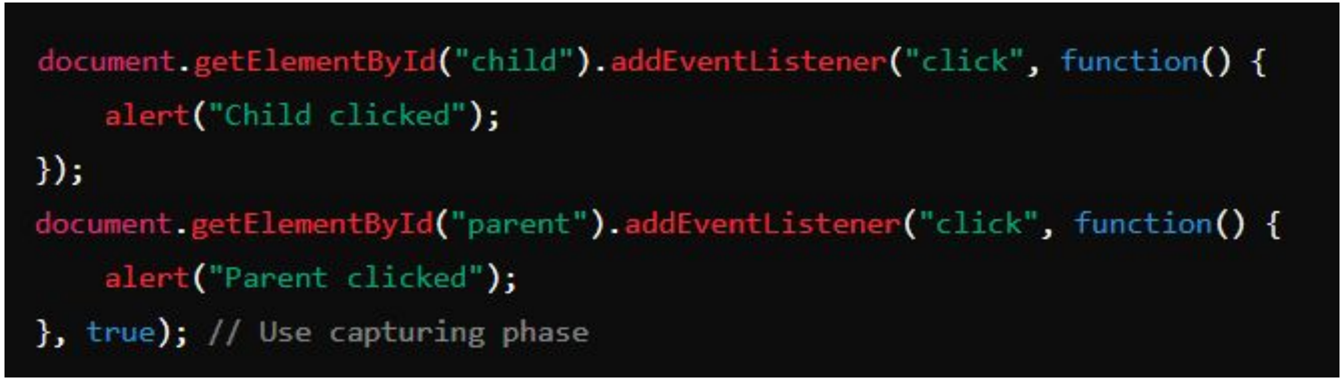
```
<button onclick="alert('Button clicked!')">Click Me</button>
```

Event Propagation

Bubbling vs. Capturing :

- **Bubbling**: Events move from the target element up to the root.
- **Capturing**: Events move from the root down to the target element.

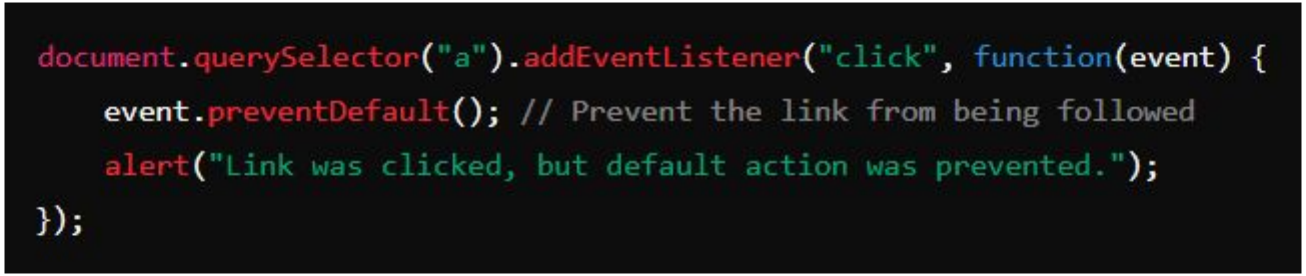
Example :



```
document.getElementById("child").addEventListener("click", function() {
    alert("Child clicked");
});
document.getElementById("parent").addEventListener("click", function() {
    alert("Parent clicked");
}, true); // Use capturing phase
```

Preventing Default Actions

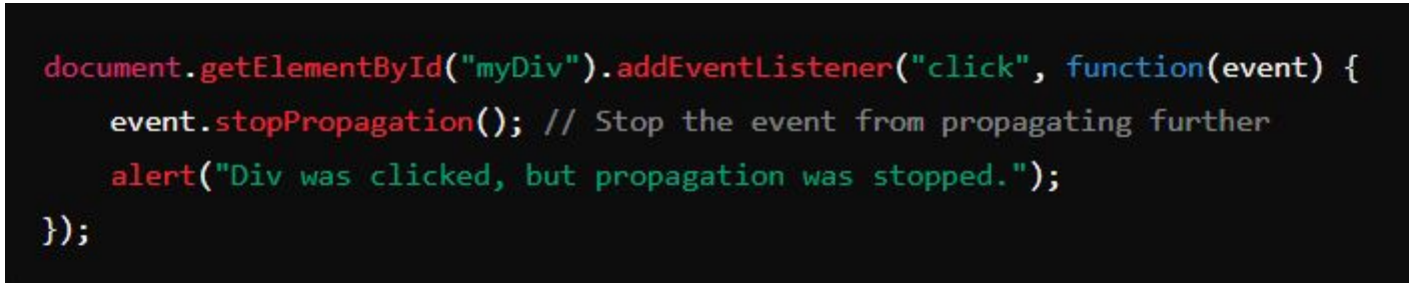
Using `preventDefault()` :



```
document.querySelector("a").addEventListener("click", function(event) {
    event.preventDefault(); // Prevent the link from being followed
    alert("Link was clicked, but default action was prevented.");
});
```

Stopping Event Propagation

Using `stopPropagation()` :



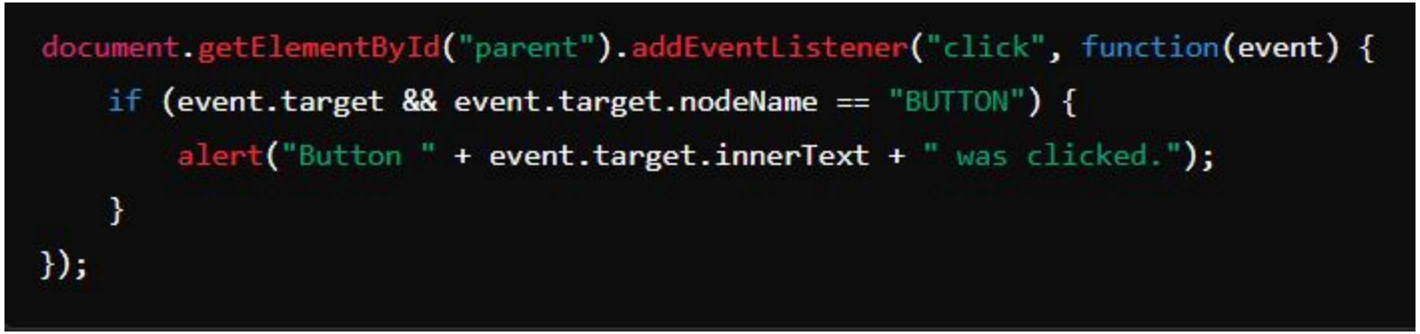
```
document.getElementById("myDiv").addEventListener("click", function(event) {
    event.stopPropagation(); // Stop the event from propagating further
    alert("Div was clicked, but propagation was stopped.");
});
```

Advanced Event Handling

Event Delegation

- **Definition**: Attaching a single event listener to a parent element to manage events for all child elements.

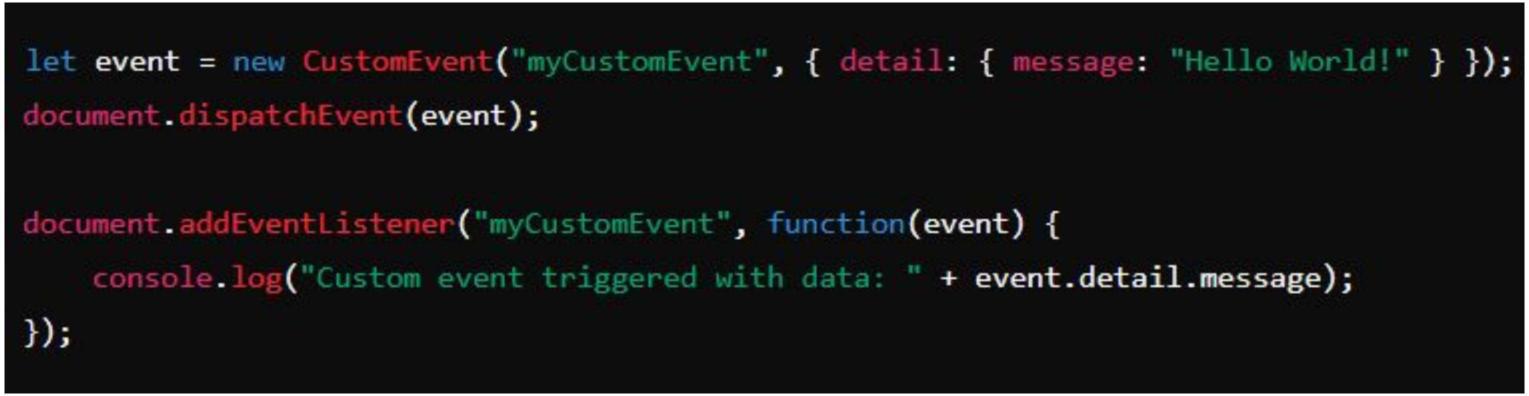
Example :



```
document.getElementById("parent").addEventListener("click", function(event) {
    if (event.target && event.target.nodeName == "BUTTON") {
        alert("Button " + event.target.innerText + " was clicked.");
    }
});
```

Custom Events

Creating and Dispatching Custom Events :

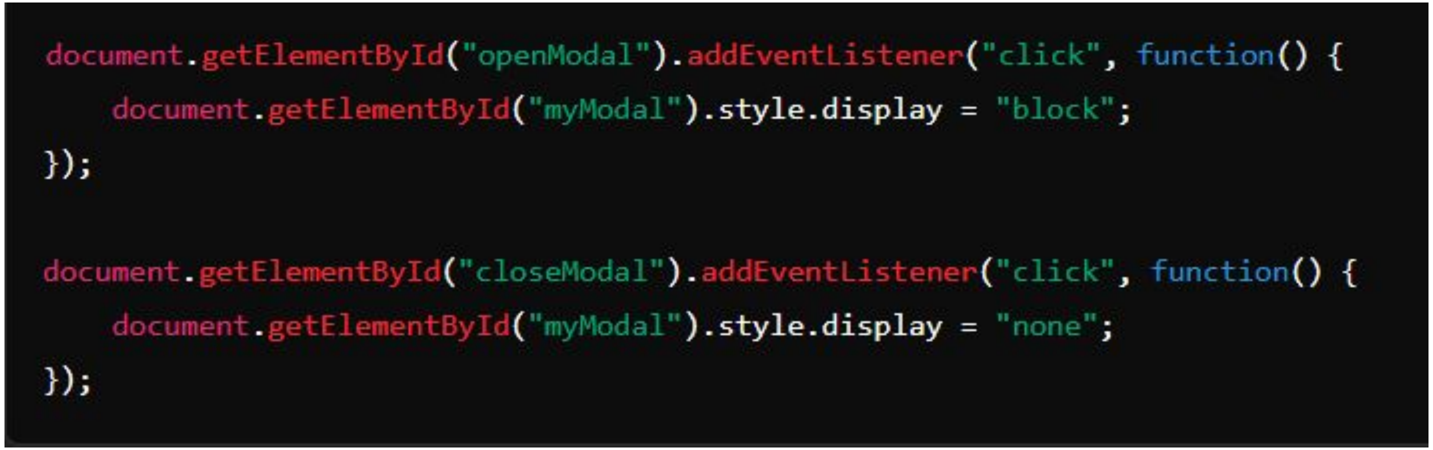


```
let event = new CustomEvent("myCustomEvent", { detail: { message: "Hello World!" } });
document.dispatchEvent(event);

document.addEventListener("myCustomEvent", function(event) {
    console.log("Custom event triggered with data: " + event.detail.message);
});
```

Creating a Simple Modal

Code Snippet :

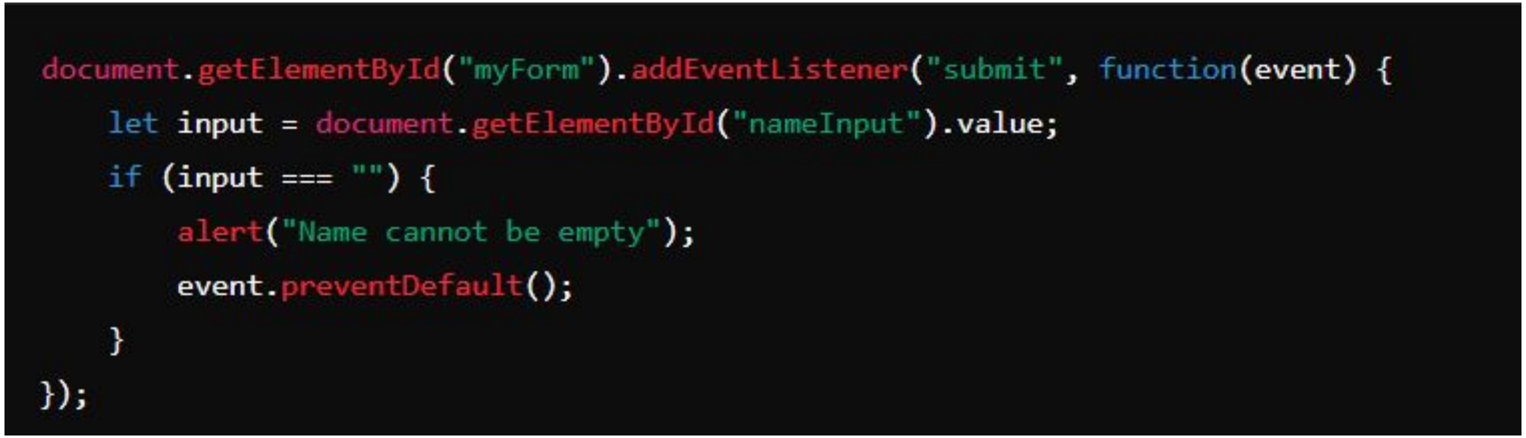


```
document.getElementById("openModal").addEventListener("click", function() {
    document.getElementById("myModal").style.display = "block";
});

document.getElementById("closeModal").addEventListener("click", function() {
    document.getElementById("myModal").style.display = "none";
});
```

Form Validation

Code Snippet :



```
document.getElementById("myForm").addEventListener("submit", function(event) {
    let input = document.getElementById("nameInput").value;
    if (input === "") {
        alert("Name cannot be empty");
        event.preventDefault();
    }
});
```

Summary

Key Takeaways:

- JavaScript events allow you to make web pages interactive.
- Various events exist for different actions (click, keypress, form submit, etc.).
- Proper event handling can significantly improve user experience.