DOM Event Handling

Lecture 3

Follow-ups Yesterday

- Source code of HTMLElement in Chrome
  - Class Header: https://github.com/chromium/chromium/blob/master/third_party/blink/renderer/core/html/html_element.h
  - Implementation: https://github.com/chromium/chromium/blob/master/third_party/blink/renderer/core/html/html_element.cc

Events

- Web pages become *interactive* when they can respond to inputs from the user.
    - Clicks a button
    - Submits a form
    - Swipes on a touch device
- Beyond user input events, there are events the browser can notify your code of.
    - When the page finishes loading
    - When the computer loses/gains access to the internet
- More thorough list: https://developer.mozilla.org/en-US/docs/Web/Events

General Eventing Strategy

- Your responsibility as the application developer is to tell the browser:

  "When some *type of event* happens on *a specific object** in my application, please *call this function*."

* We'll see that objects in the DOM can listen for events that happen to its children, as well.

The *EventTarget* Interface

- The *interface* of objects that allow you to register event handling functions with implement `EventTarget`
- Many BOM and DOM classes implement the *EventTarget* Interface including…
  - Window
  - Document
  - Element (and, thus, every HTMLElement!)

EventTarget Interface

- Objects implementing EventTarget have the following three methods associated with them:

- #addEventListener(“event type”, handlerFn); - Register. When the event occurs, the event is “raised” and any handlerFns are called.

- #removeEventListener(“event type”, handlerFn); - Unregister a handler. This requires having a reference to the handler function.

- #dispatchEvent(event); - Raise an event synthetically (*typically* for testing, occasionally clever things you can do with this)

- Reference: https://developer.mozilla.org/en-US/docs/Web/API/EventTarget

Mouse Events

- Example types of mouse events:
  - mousedown/mouseup - button clicked/released
  - mouseover/mouseout - mouse pointer enters/leaves
  - mousemove - mouse pointer moves while inside an event
- Reference:
  - https://developer.mozilla.org/en-US/docs/Web/API/MouseEvent
  - https://javascript.info/mouse-events-basics

Let’s Tinker with Events!

Event Bubbling

- When an event occurs on an HTMLElement, the event is first raised on the most specific element it applied to.
- Then, it is raised on the target’s parent element. Then on its parent element, and so on, until the event reaches the root element.
- This enables event delegation.

Event Bubbling

- `this` or `event.currentTarget` is the element whose handler is currently running
- `target` was the element which kicked off the event
- Event bubbling propagates the event up the hierarchy

Event Delegation

- A powerful, common pattern in web applications that involve many similarly typed “components” coexisting in some container (such as thumbnails in an image gallery) is event delegation.

- Event delegation employs event bubbling by listening for events on a parent container *for its children’s events* rather than on each child individually.

- This technique has two important advantages:

1. Fewer event listeners needed (one per parent container rather than one per child).

2. Less book keeping of event listeners when the children are dynamically added and removed (with delegation, you do not need to manually add/remove listeners to each child as they’re added/removed).

- Reference: https://javascript.info/event-delegation