

# EVENT HANDLING

# TODAY'S OBJECTIVES

- Understand what event-driven programming is
- Select DOM objects and attach anonymous functions with `addEventListener()`
- Describe event bubbling and how it works
- Describe default browser behavior and what elements need to have default behavior handled (`form, a`)
- List possible event types and what elements get those events
- Describe how to add listeners to newly created DOM elements

# WHAT IS EVENT-DRIVEN PROGRAMMING?

So far, we have controlled the flow of our program logic completely. But when a user is working with a web page, doing things like clicking, entering data in fields, and so on, our code needs to run based on the actions of the user.

The browser is aware of anything that happens on a page

- Mouse click, mouse over, input field change, form submit, etc.
- These are all **events**
- **Event-driven programming** is programming that responds to these event

# MORE ABOUT EVENTS

- Every **event relates to** specific **DOM element**.
- When an event occurs (is triggered) on a DOM element, it **publishes** the event.
- Our JavaScript code can ask to be **notified** when specific events are published. This is called **subscribing to an event** and is accomplished by adding an **event listener**.
- An event listener is a JavaScript function that we write.
- Our code responds to the flow dictated by the user - this is called an **event-driven interface**.


# PAGE LOADING SEQUENCE

- Browser reads the page and starts processing elements top-down
  - As it is read, browser builds the DOM
  - Page also may start to render
- When a `<script>` tag is encountered, the browser stops other processing and runs the script
- When entire page is read and the DOM is built, the `document.DOMContentLoaded` event is triggered
  - **We want to wait for this event before we start manipulating the DOM**
- Browser continues to get external files (CSS, IMG) to complete the page
- When all external content has been loaded, `window.load` is triggered
- More info:
  - <https://javascript.info/onload-ondomcontentloaded>
  - <https://www.innoq.com/en/blog/loading-javascript>

# LISTENING FOR THE DOMCONTENTLOADED EVENT

Here's how we would add an eventListener that will listen for the DOMContentLoaded event. When the event is fired, this function will get called and we can do the rest of our DOM setup. This code should be in global scope.

Element to add  
eventListener to



```
document.addEventListener('DOMContentLoaded', () => {  
  // When the DOM Content has loaded we register other  
  // eventListeners here.  
});
```

# LISTENING FOR THE DOMCONTENTLOADED EVENT

Here's how we would add an eventListener that will listen for the DOMContentLoaded event. When the event is fired, this function will get called and we can do the rest of our DOM setup. This code should be in global scope.

Element to add  
eventListener to

function to add  
event listener

```
document.addEventListener('DOMContentLoaded', () => {  
  // When the DOM Content has loaded we register other  
  // eventListeners here.  
});
```

# LISTENING FOR THE DOMCONTENTLOADED EVENT

Here's how we would add an eventListener that will listen for the DOMContentLoaded event. When the event is fired, this function will get called and we can do the rest of our DOM setup. This code should be in global scope.

Element to add  
eventListener to

function to add  
event listener

event to listen for

```
document.addEventListener('DOMContentLoaded', () => {  
  // When the DOM Content has loaded we register other  
  // eventListeners here.  
});
```



# THE EVENT OBJECT

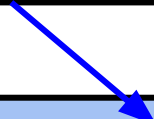
Property	Found In	Purpose
currentTarget	All events	Holds the element that the event was triggered on, ie. the button clicked or the select box that changed
clientX	Mouse events	The X coordinate on the screen of the click
clientY	Mouse events	The Y coordinate on the screen of the click
altKey, metaKey, ctrlKey, shiftKey	Mouse and Keyboard events	A boolean on whether the specified key was pressed down during the event
key	Keyboard events	The key that was pressed, taking the Shift key into account. Arrow keys show up as 'ArrowRight', 'ArrowDown', 'ArrowLeft', and 'ArrowUp'

# LISTENING FOR MOUSE EVENTS

We can listen for these mouse events:

- click
- dblclick
- mouseover
- mouseout
- mouseleave
- mousemove

event object we can use  
in our method



```
element.addEventListener('click', (event) => {  
    // do something with the event  
});
```

# LISTENING FOR KEYBOARD EVENTS

We can listen for these key events:

- keydown
- keyup

To read the key value, we check the event's key property.

[https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key/Key\\_Values](https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key/Key_Values)

# OTHER TYPES OF EVENTS

We can listen for these mouse events:

- Event
  - change (input, select or textarea)
  - submit (form)
  - reset (form)
- FocusEvent
  - focus
  - blur

# EVENT BUBBLING (PROPAGATION)

- An event is triggered on some source element
- Browser looks for event handler on the element, invokes if found
- Then it keeps looking for event handler on the element's parent, invokes if found
- And so on, up to the window object
- If you want to change this and stop the "bubbling", call **`event.stopPropagation()`**

# PREVENTING DEFAULT

- Anchors `<a>` and Submit buttons `<input type="submit">` have default behavior
- Anchor navigates to a URL when clicked
- Form posts to server when submitted
- Clicking a checkbox toggles the checked state of the control
- You may want to override their behavior
  - E.g., use an anchor to hide a section
- To prevent the default behavior from happening, call `event.preventDefault()`
- **NOTE: `preventDefault` does not stop propagation**