Lecture 6.1

Intro to Events

Topics

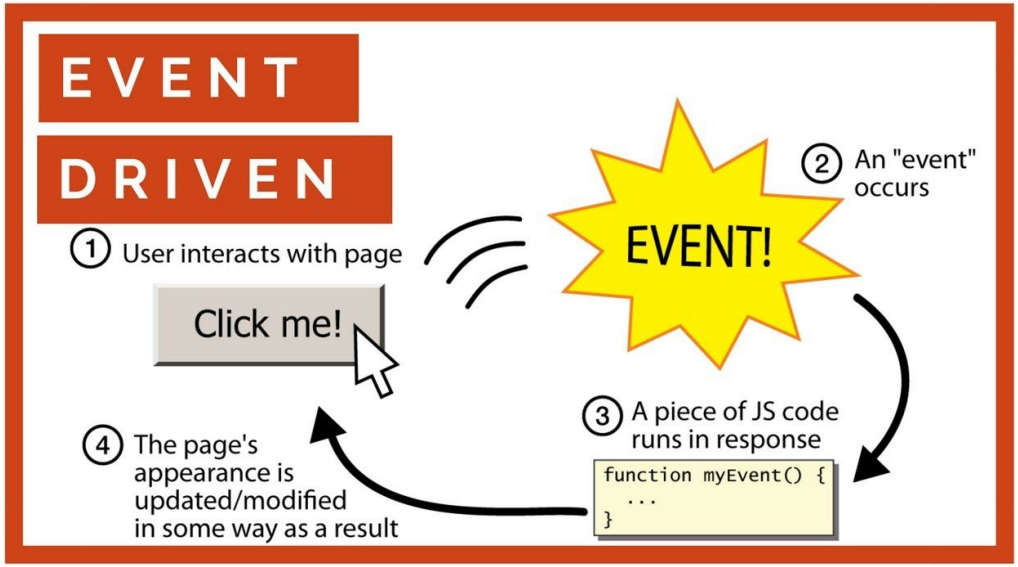
- **Intro to Events**

Intro to Events

Event Driven

An event is an action that occurs in the web browser, which the web browser feedbacks to you so that you can respond to it.

For example, when users click a button on a webpage, you may want to respond to this click event by displaying a dialog box.



A series of fortunate events

As mentioned above, **events** are actions or occurrences that happen in the system you are programming — the system produces (or "fires") a signal of some kind when an event occurs, and provides a mechanism by which an action can be automatically taken (that is, some code running) when the event occurs.

For example, in an airport, when the runway is clear for take off, a signal is communicated to the pilot. As a result, the plane can safely takeoff.

Web Events

In the case of the Web, events are fired inside the browser window, and tend to be attached to a specific item that resides in it — this might be a single element, set of elements, the HTML document loaded in the current tab, or the entire browser window.

Web Events

There are many different types of events that can occur. For example:

- The user selects a certain element or hovers the cursor over a certain element.
- The user chooses a key on the keyboard.
- The user resizes or closes the browser window.
- A web page finishes loading.
- A form is submitted.
- A video is played, paused, or finishes.
- An error occurs.

You can look at the MDN, there a lot of events that can be responded to
https://developer.mozilla.org/en-US/docs/Web/Events

Event Handlers

Event Handlers

Each available event has an **event handler**, which is a block of code (usually a JavaScript function that you as a programmer create) that runs when the event fires.

When such a block of code is defined to run in response to an event, we say we are **registering an event handler**.

Note: Event handlers are sometimes called **event listeners** — they are pretty much interchangeable for our purposes, although strictly speaking, they work together. The listener listens out for the event happening, and the handler is the code that is run in response to it happening.

An event handler is also known as an event listener. It listens to the event and executes when the event occurs.

```
<button id="btn">Click Me!</button>
```

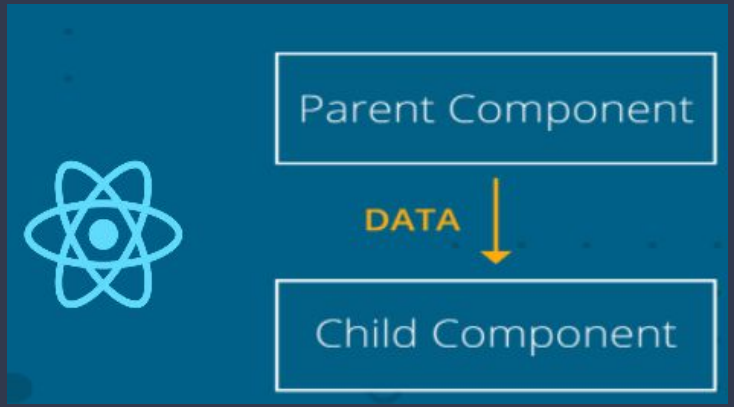
To define the code that will be executed when the button is clicked, you need to register an event handler using the addEventListener() method:

```
let btn = document.querySelector('#btn');

function display() {
    alert('It was clicked!');
}

btn.addEventListener('click',display);
```

Event Flow

Video - Events

Flow



The concept of flow is ubiquitous in JavaScript today. For example, one-way data flow in React, flow in Node and DOM event flow are all vivid manifestations of flow.

As for the specific concept of flow, in terminology, flow is an abstraction of input and output devices. From a programmatic point of view, streams are directed data.

```html
<!DOCTYPE html>
<html>
<head>
    <title>JS Event Demo</title>
</head>
<body>
    <div id="container">
        <button id='btn'>Click Me!</button>
    </div>
</body>
```

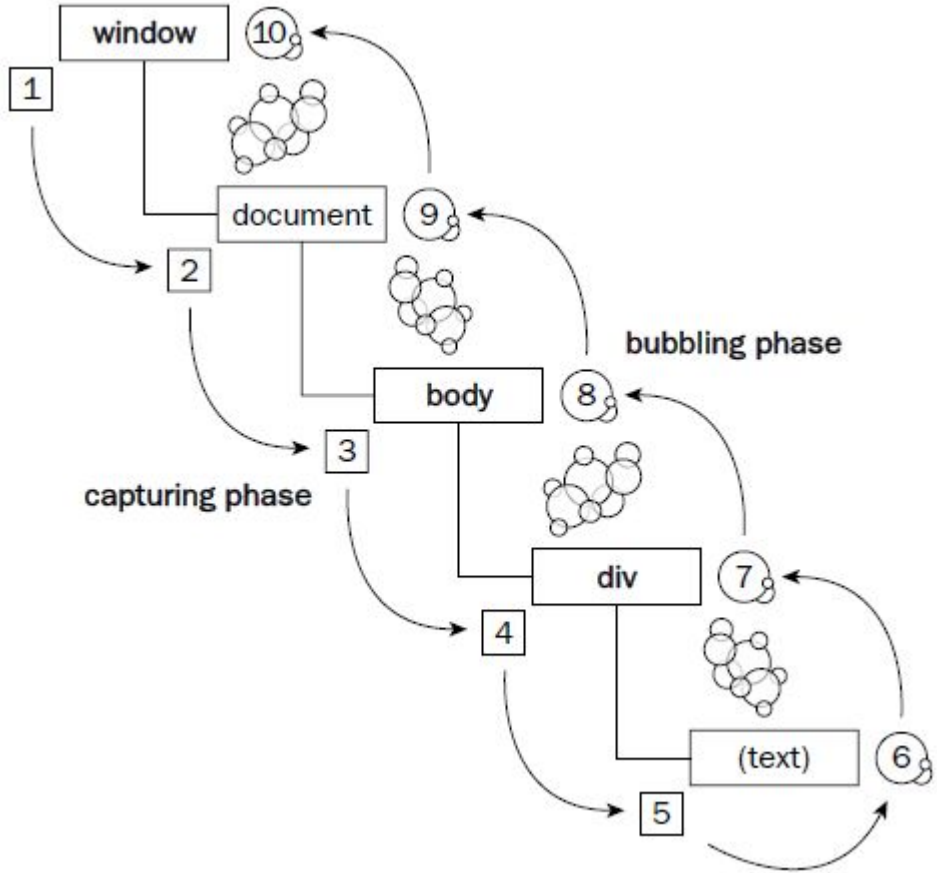
When you click the button, you're clicking not only the button but also the button's container, the div, and the whole webpage.

Event Flow

Event Flow cont..

Event flow explains the order in which events are received on the page from the element where the event occurs and propagated through the DOM tree.

There are two main event models: event bubbling and event capturing.



Event Bubbling

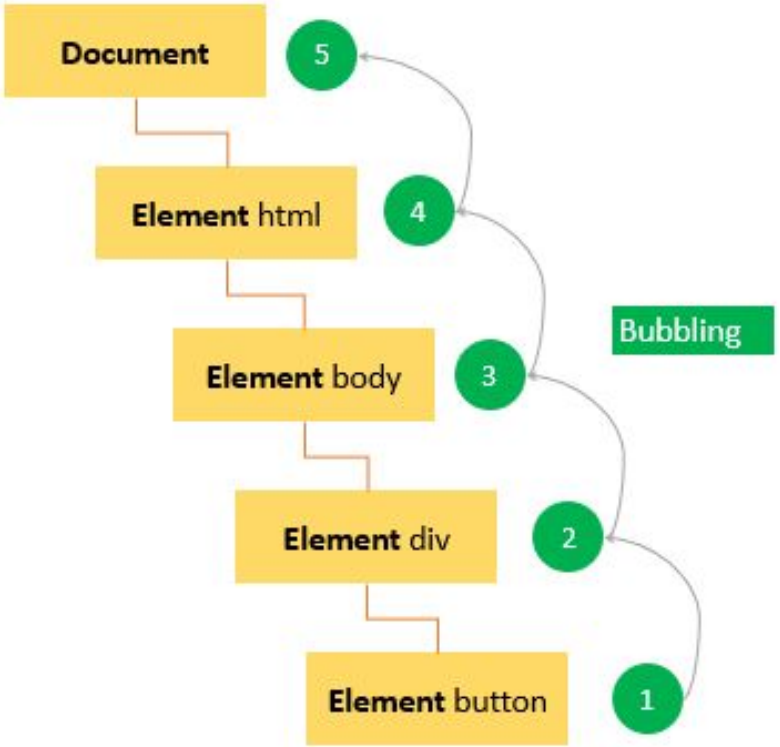
In the event bubbling model, an event starts at the most specific element and then flows upward toward the least specific element (the document or even window).

When you click the button, the click event occurs in the following order:

1. button
2. div with the id container
3. body
4. html
5. document

The click event first occurs on the button which is the element that was clicked.

Then the click event goes up the DOM tree, firing on each node along its way until it reaches the document object.



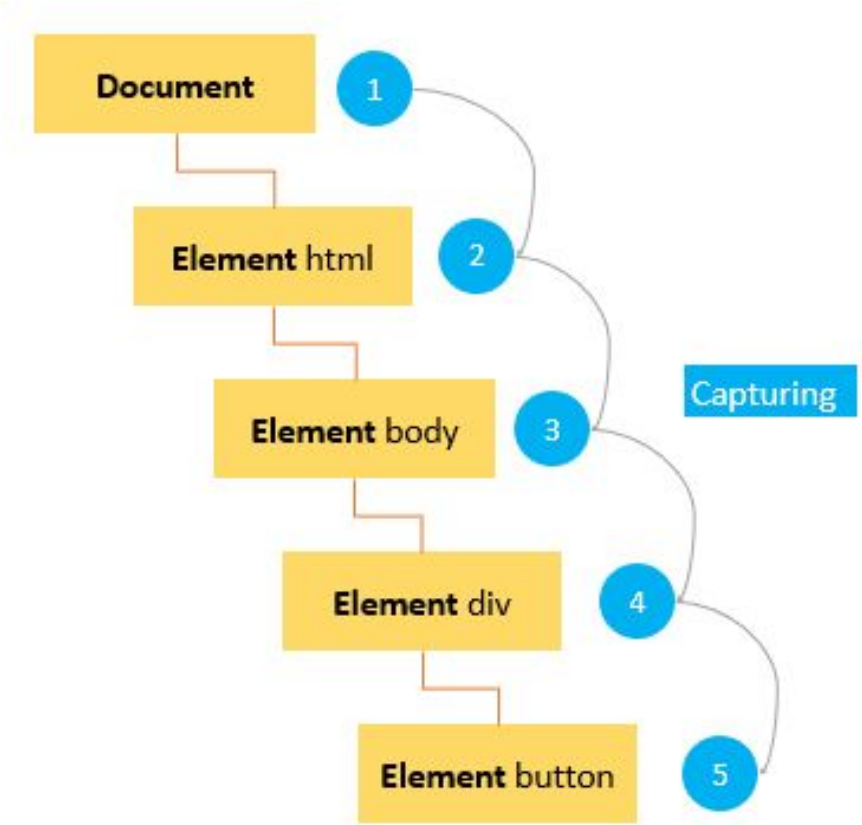
Note that modern web browsers bubble the event up to the window object.

Event Capturing

In the event capturing model, an event starts at the least specific element and flows downward toward the most specific element.

When you click the button, the click event occurs in the following order:

1. document
2. html
3. body
4. div with the id container
5. button

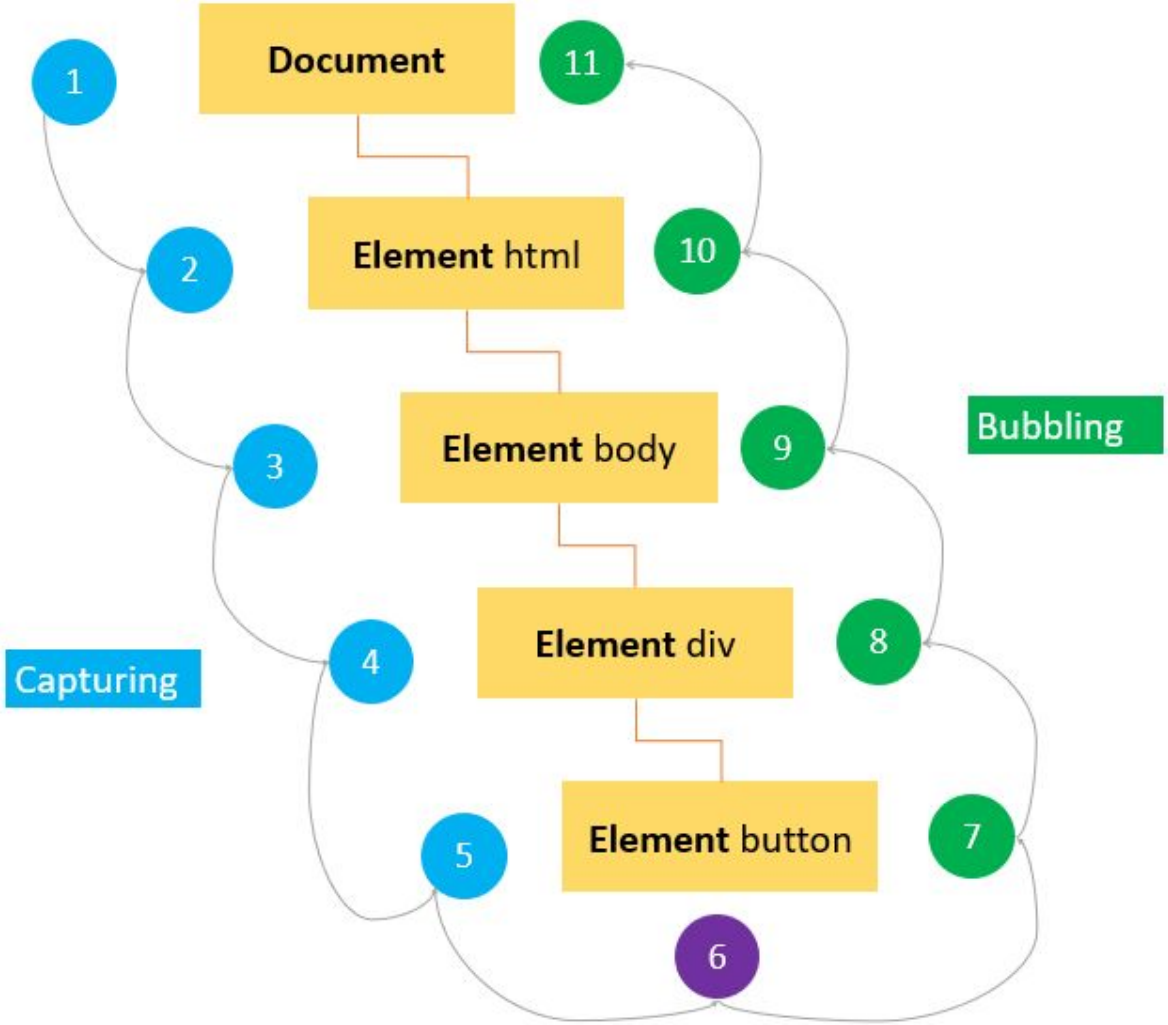


The event capturing effect..

DOM Level 2 Event Flow

DOM level 2 events specify that event flow has three phases:

1. First, event capturing occurs, which provides the opportunity to intercept the event.
2. Then, the actual target receives the event.
3. Finally, event bubbling occurs, which allows a final response to the event.


Document
Element html
Element body
Element div
Element button
1
2
3
4
5
6
7
8
9
10
11
Capturing
Bubbling

Event Object

Video - Event Bubbling

The following table shows the most commonly-used properties and methods of the event object:

Property / Method	Description
bubbles	true if the event bubbles
cancelable	true if the default behavior of the event can be canceled
currentTarget	the current element on which the event is firing
defaultPrevented	return true if the preventDefault() has been called.
detail	more informatio nabout the event

When the event occurs, the web browser passed an Event object to the event handler:

The following table shows the most commonly-used properties and methods of the event object:

eventPhase	1 for capturing phase, 2 for target, 3 for bubbling
preventDefault()	cancel the default behavior for the event. This method is only effective if the `cancelable` property is true
stopPropagation()	cancel any further event capturing or bubbling. This method only can be used if the `bubbles` property is true.
target	the target element of the event
type	the type of event that was fired

When the event occurs, the web browser passed an Event object to the event handler:

Event Prevent Default Behavior

To prevent the default behavior of an event, you use the preventDefault() method.

The method tells the user agent that if the event does not get explicitly handled, its default action should not be taken as it normally would be

```
event.preventDefault();
```

Event Stop Propation

The **stopPropagation()** method of the Event interface prevents further propagation of the current event in the capturing and bubbling phases.

It does not, however, prevent any default behaviors from occurring; for instance, clicks on links are still processed. If you want to stop those behaviors, see the preventDefault() method.

```
event.stopPropagation();
```