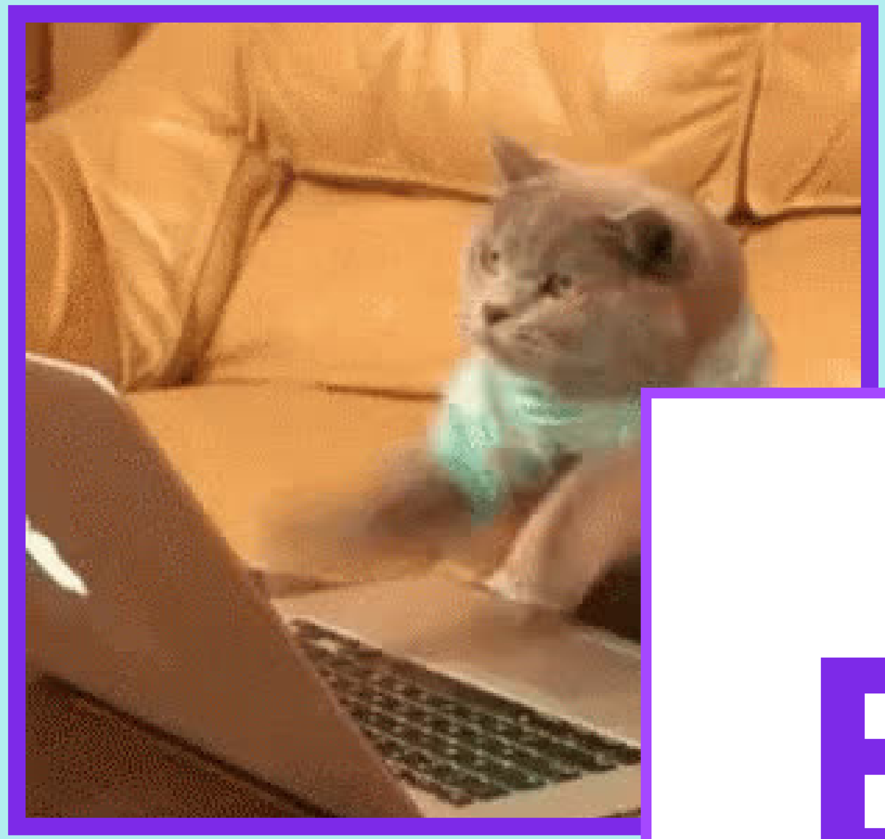
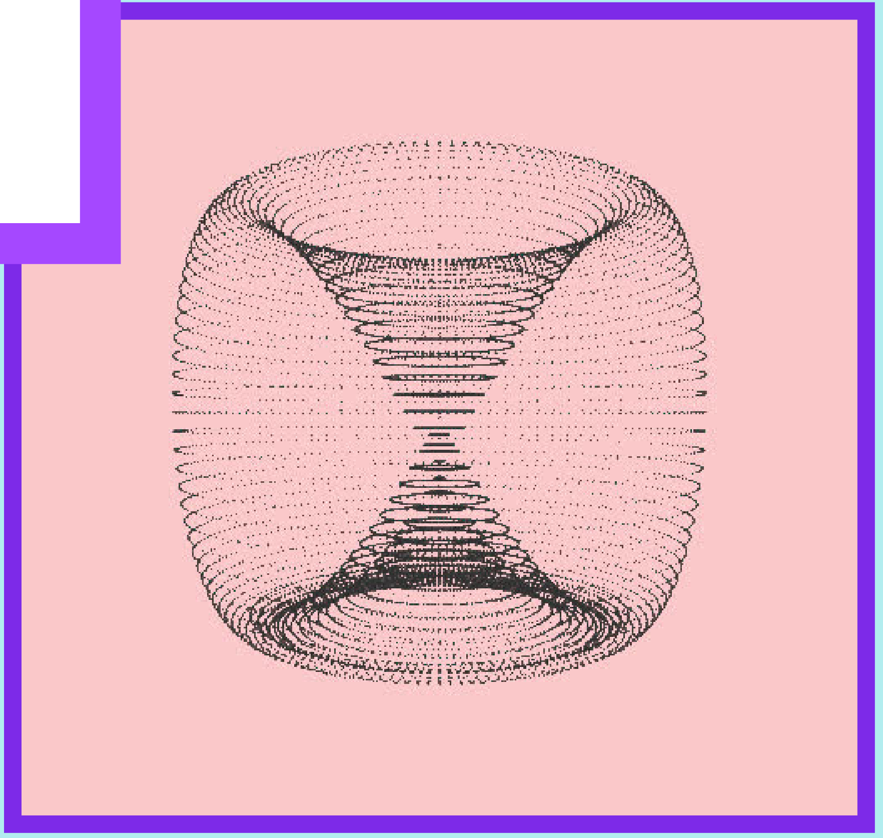


Events

Presented by
Anna Azzam



What is an Event?

An *event* is a signal that a "thing" has happened to a DOM element

This "thing" can be the element *loading*, a *click*, or a *key press*

We can use events to run JavaScript code in response to user actions

What are some examples of events?

Mouse events

click
dblclick
mouseup
mouseenter
mousedown
mouseleave

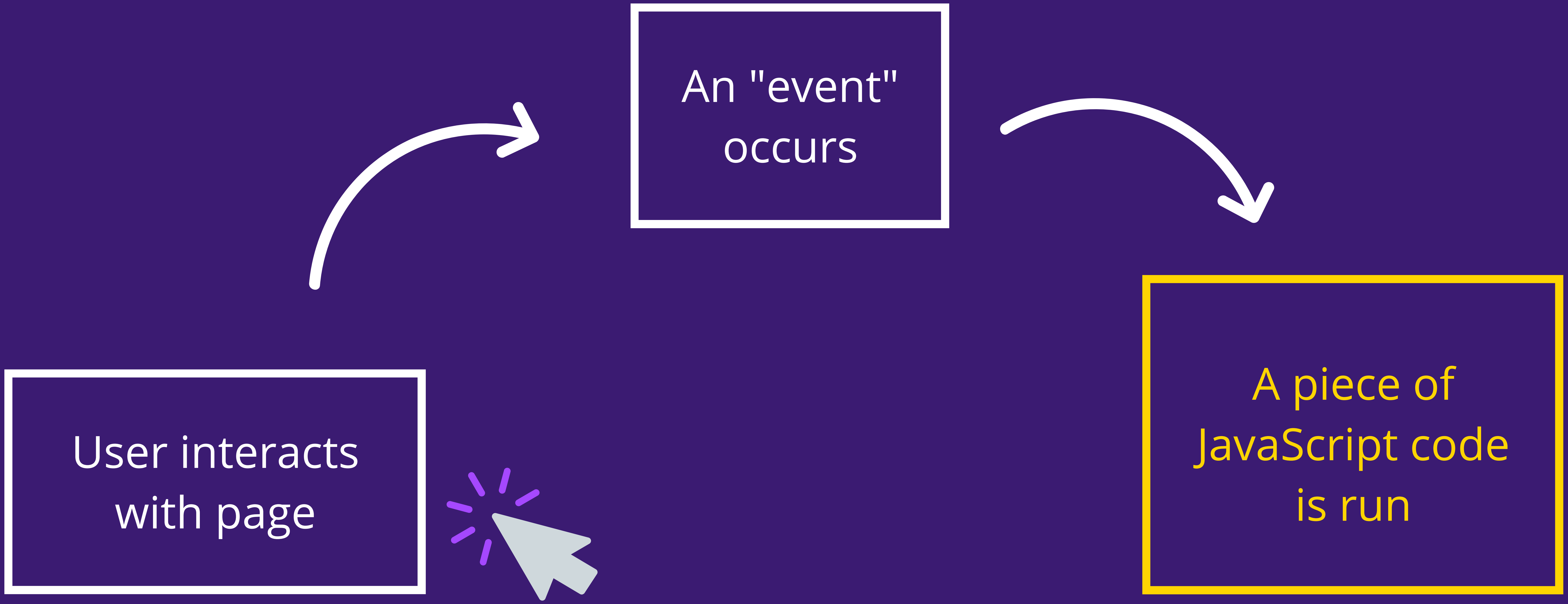
Keyboard events

keydown
keypress
keyup

...and more!

error
load
fullscreenchange
submit
canplay
canplaythrough
animationstart

Event Handlers



Adding Event Handlers

1) In HTML
2) DOM Property
3) addEventListener

```
<input
  value="Click me"
  onclick="alert('Clicked!')"
  type="button"
>
```

Adding Event Handlers

1) In HTML
2) DOM Property
3) addEventListener

```
let element = document.getElementById('btn');

element.onclick = () => {
  alert('Button was clicked!');
};
```

```
function doSomething() {
  alert('hello');
}

element.onclick = doSomething();
```

What's wrong with this code?

A

```
function doSomething() {
  alert('hello');
}

element.onclick = doSomething();
```

By adding parentheses, we are executing doSomething rather than assigning a function to onclick.

A

```
function doSomething() {
  alert('hello');
}

element.onclick = doSomething;
```

By adding parentheses, we are executing doSomething rather than assigning a function to onclick.

Adding Event Handlers

1) In HTML
2) DOM Property
3) addEventListener

```
// Definition:
target.addEventListener(
  type, // e.g. 'click', 'mousemove'
  listener, // the callback
  [options]
);

// Example:
let element = document.getElementById('btn');
let handler = () => {
  alert('button was clicked');
})

element.addEventListener('click', handler);
element.removeEventListener('click', handler);
```

Event Interface

The parameter to an event handler is an event object

```
document.addEventListener('mousemove', (event) => {
  console.log(event.clientX);
  console.log(event.clientY);
});
```

The event object represents the event that has taken place, and has properties describing details of the event

Event Interface Properties

Some of the properties on the event interface include:

```
event.currentTarget // current element handler is running on
event.timeStamp // time the event was created (in ms)
event.type // name of the event, e.g. 'click'
```

Different types of events have specific properties:

```
event.clientX // A MouseEvent has the X and Y coordinate
event.key // A KeyboardEvent has the keycode of the key that was pressed
```

Keyboard Events Example

The Event Loop

- The event loop is a single-threaded loop which runs in the browser, and manages all events.
- When an event is triggered, the event is added to the queue.

```
while (queue.waitForMessage()) {
  queue.processNextMessage();
}
```

The Event Loop

JavaScript uses a run-to-completion model, meaning it will not handle a new event until the current event has completed.

Event Capturing and Bubbling

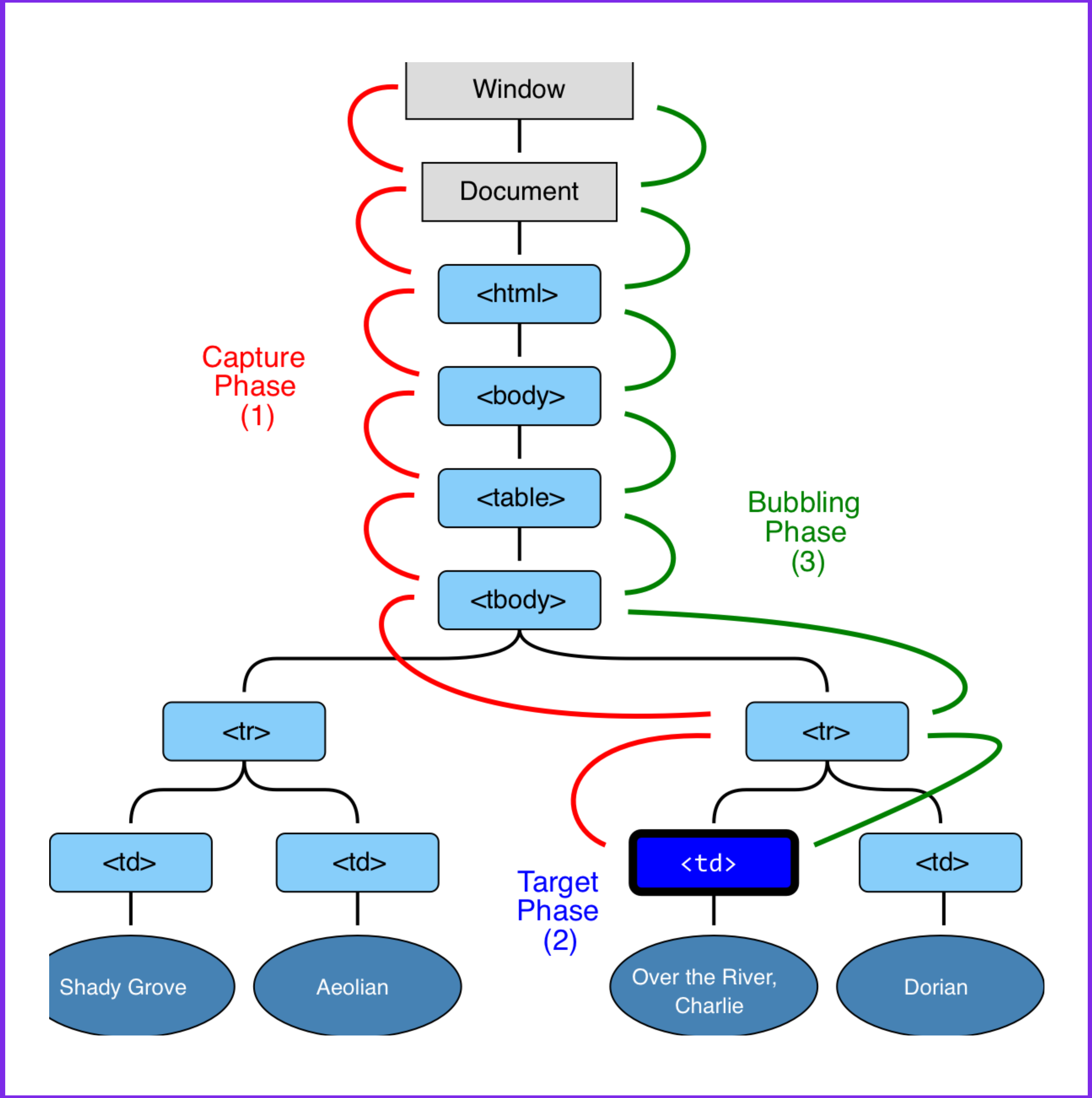


Image Source: https://javascript.info/bubbling-and-capturing

Event Capturing and Bubbling Example

Prevent Default

Some types of DOM elements have default behaviour, e.g.

1. Clicking an input checkbox toggles the checkbox
2. Images have a default drag and drop behaviour to allow you to drag them into another location
3. Key presses into a text input field has the default behaviour of entering that text into the input field

To stop the default behaviour of an event, use:

```
event.preventDefault()
```

Prevent Default Example

Basketball drag and drop game