

DOM
events
featuring...

INDIANA JONES

and the

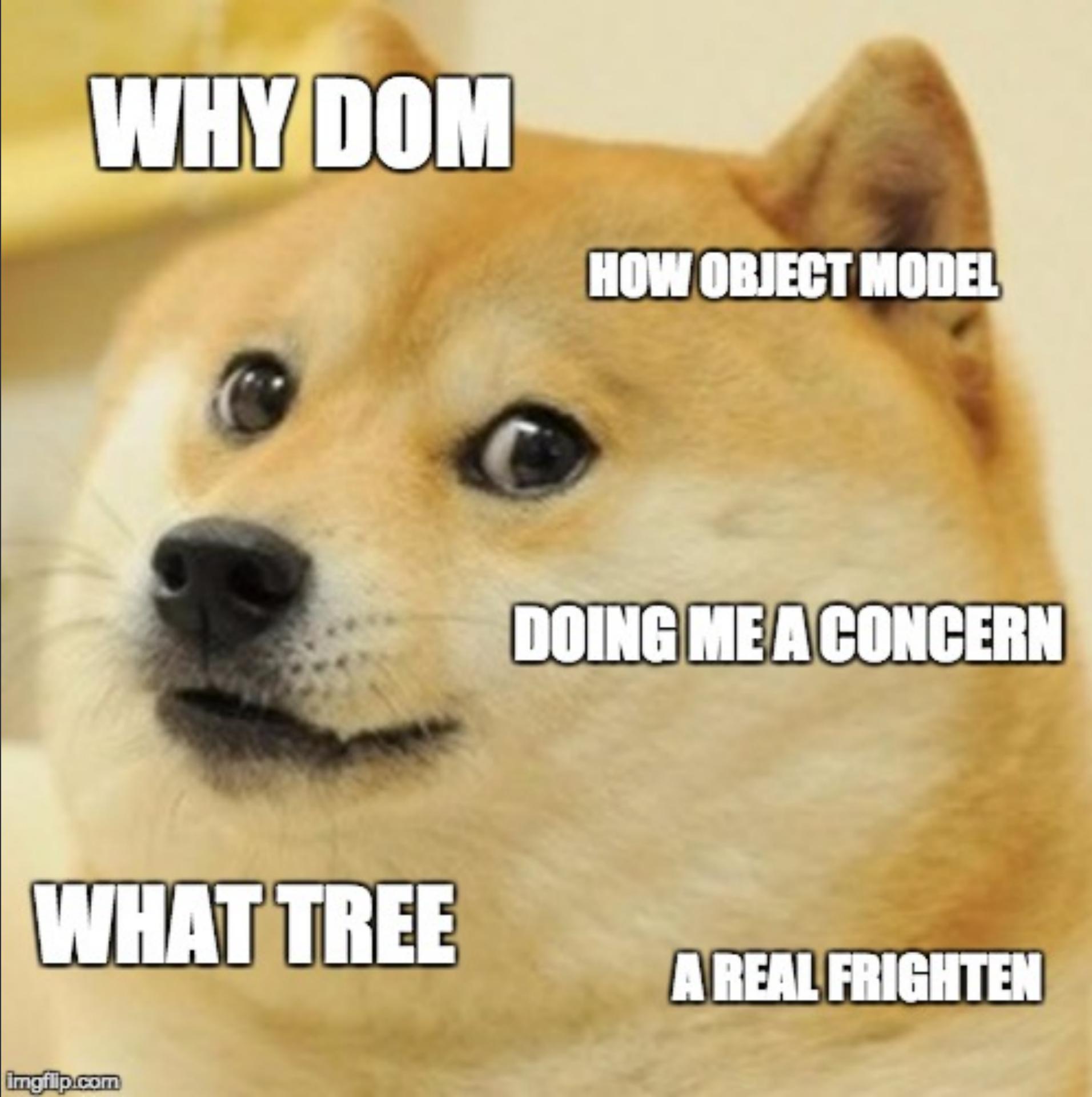
TEMPLE OF DOOM



Objectives

**By the end of this lesson you should
be able to:**

- Identify event *targets*, event *listeners*, and event *callbacks*
- Attach *event listeners* to DOM elements
- Use *callbacks* with methods like *addEventListener()*
- Explain event bubbling
- Explain the difference between *this* and *event.target* in callbacks



WHY DOM

HOW OBJECT MODEL

DOING ME A CONCERN

WHAT TREE

A REAL FRIGHTEN

Rather than thinking about the DOM as an abstract concept where things happen inexplicably in some invisible place, think of it as setting booby traps in your fortress of HTML/JS solitude.



Setting the Trap

HTML Doc

what to find

what makes it do the thing

```
document.getElementById("statue").addEventListener("mouseup", releaseBoulder);
```

how to find the element

get it ready to do a thing

the thing it do

The diagram illustrates the components of an event listener assignment. It consists of several text labels connected by brackets to specific parts of the code: 'HTML Doc' points to the entire code block; 'what to find' points to 'getElementById("statue")'; 'what makes it do the thing' points to 'addEventListener("mouseup", releaseBoulder)'; 'how to find the element' points to 'document.'; 'get it ready to do a thing' points to 'releaseBoulder'; and 'the thing it do' points to the parameter 'mouseup'.



`document.getElementById("gun")`



```
document.getElementById("tile").addEventListener("click", shootArrow)
```



Every DOM element also has its own **addEventListener()** method, that can listen for common events

Event Types

- Mouse Events
- Keyboard Events
- Drag and Drop Events
- Focus Events

Paired research: Event types

How to add/remove

```
// doStuff is a "callback"  
var doStuff = function() {  
    alert("PARTY TIME");  
}  
  
myElement.addEventListener("click", doStuff);
```

```
// Or just anonymous function  
myElement.addEventListener("click", function(){  
    alert("COWABUNGA")  
});  
  
myElement.removeEventListener("click");  
// Clicky no more worky
```

A man with a beard and sunglasses, wearing a light-colored hoodie, is talking on a white smartphone. He is standing next to a red brick wall. In the background, there's a blue car and some foliage. The image has a slightly grainy, meme-like quality.

Hey Girl,

**I wish I were an
Event Listener...**

...so you would give me a callback

Event Object

Callback functions can take an *event object* as a parameter. This object represents most everything we need to know about the event that occurred. For example we can find out the *target* of the event (the DOM element that triggered the event)

```
var logTarget = function(event) {  
  console.log(event.target); // Log it  
}
```

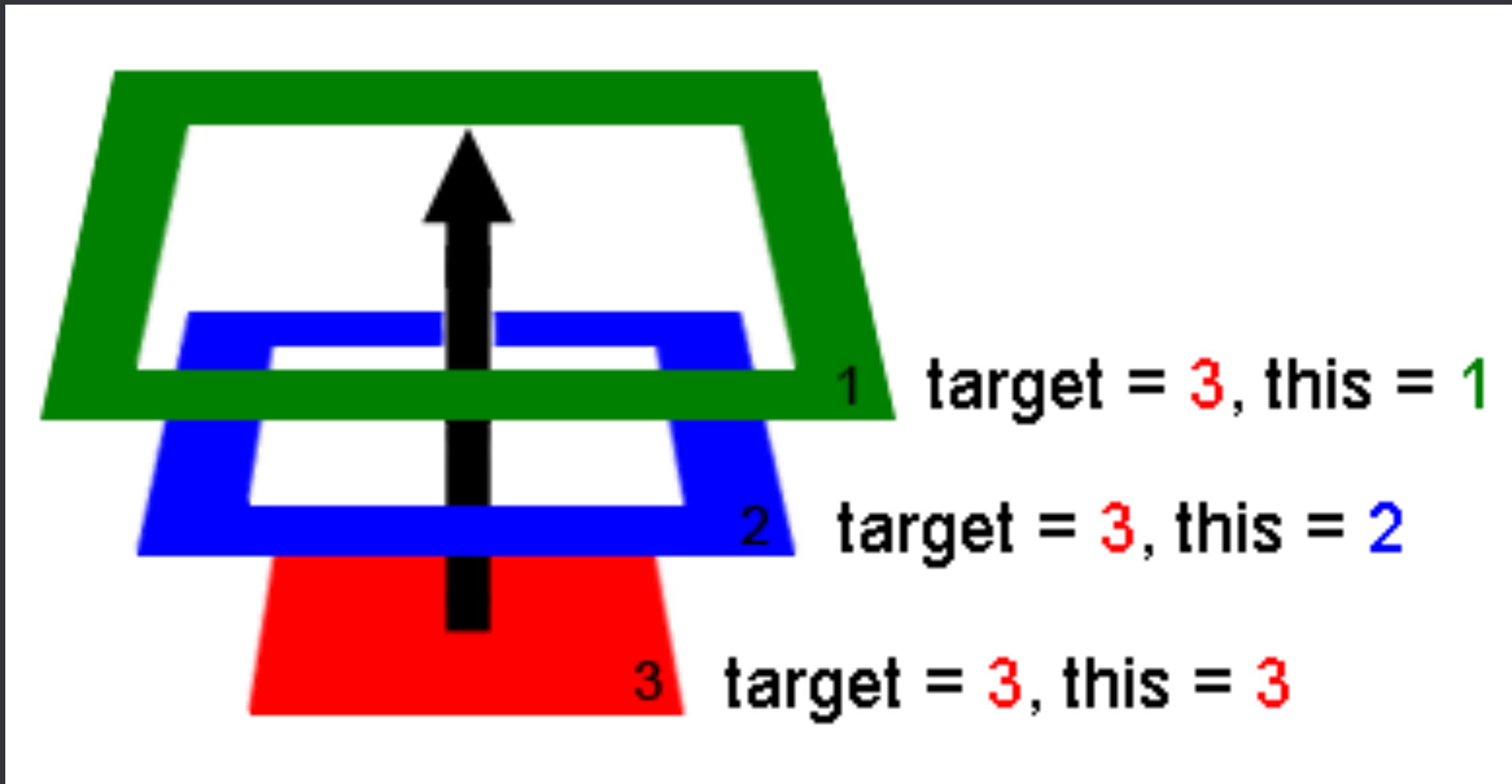
```
var ele = document.getElementById("someElement"); // Grab my element  
ele.addEventListener("click", logTarget); // Add listener/callback
```

Event Bubbling

We can place an event listener on a parent element that captures the events from all the children. The event "bubbles up" from the child (*event.target*) to the parent with the attached listener (*this*).

Codepen example

Event Bubbling



Best Practice: Wait for the DOM to finish loading

The DOMContentLoaded hook is itself implemented with an event listener. Ensure you place *ALL* your javascript within this block, including your various event listeners.

```
document.addEventListener("DOMContentLoaded", function() {  
    // ALL JS IN HERE  
  
    var img = document.querySelector('img');  
    img.addEventListener('mouseover', imgLog);  
});
```

Review Objectives

- Identify event *targets*, event *listeners*, and event *callbacks*
- Attach *event listeners* to DOM elements
- Use *callbacks* with methods like *addEventListener()*
- Explain event bubbling
- Explain the difference between *this* and *event.target* in callbacks