JQUERY

Events and event handling

When you view a website

The browser loads all the JS, CSS and HTML and then... waits...

Events and event handling

The browser loads all the JS, CSS and HTML and then... waits...

It waits for the user to interact with the site, for things to happen internally

Events and event handling

The browser loads all the JS, CSS and HTML and then... waits...

It waits for the user to interact with the site, for things to happen internally

And then it reacts when things happen!

Events and event handling

And then it reacts when
things happen!

These "things" are events

Events and event handling

These "things" are events

Events are generated due to **user actions** - clicking on a button, hovering over an element, typing in textboxes

Events and event handling

These "things" are events

Other events are system driven - page completes loading, a view scrolls to the very bottom, video completes playing

Events and event handling

These "things" are events

Whenever something interesting happens the browser fires an event

Events and event handling

These "things" are events

Whenever something interesting happens the browser fires an event

It is an announcement of sorts so anyone who is interested is informed

anyone who is interested

Actions constantly happen on a webpage but the only way to be informed about them is to listen for them

Events and event handling

anyone who is interested

listen for them

Events and event handling

anyone who is interested

Your code is notified when you have it listen for events

Events and event handling

anyone who is interested

To tell the browser what to do
when an event occurs you
specify an event handler

When an event is fired, the event handler kicks into action and your code is run in response to that event

How do you listen to an event?

```
<button onclick="alert('Hello')">Say hello</button>
```

Specify the code to be executed on the onclick attribute of the button

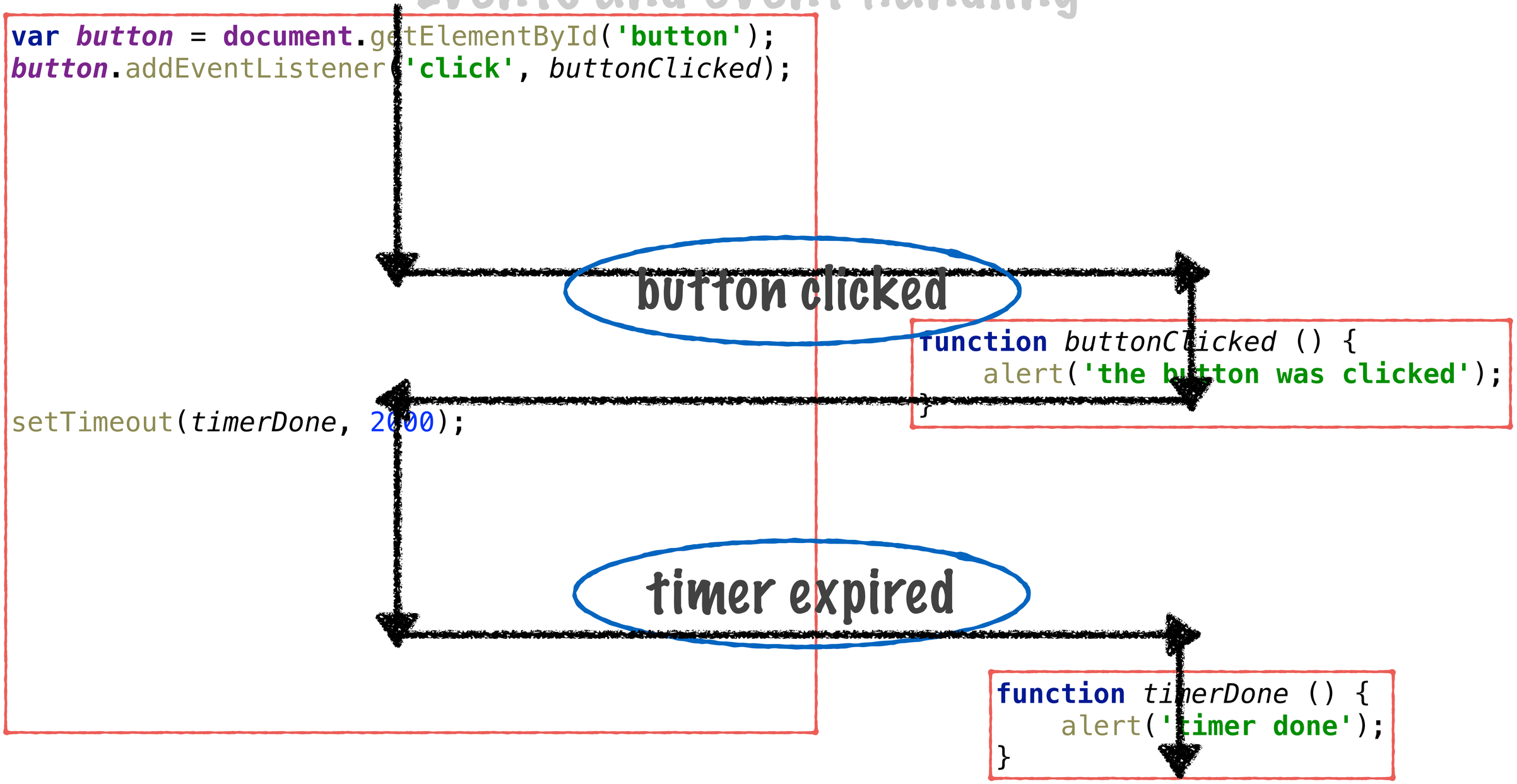
How do you listen to an event?

```
// Event binding using addEventListener
var helloBtn = document.getElementById( "helloBtn" );

helloBtn.addEventListener( "click", function( event ) {
    alert( "Hello." );
}, false );
```

Add a listener using the addEventListener method of the DOM element

How does execution work when an event is received?

Events and event handling



EXAMPLE 9

EventHandlers.html

EventHandlers.html

```
<head>
    <title>Event Handlers</title>
    <script src="../../lib/jquery.js"></script>
    <script>
        $(document).ready(function () {
            $('#clickButton').click(function (e) {
                alert('This is a "click" handler!');
            });
            $('#bindButton').bind('click', function (e) {
                alert('This is a "bind" handler!');
            });
            $('#onButton').on('click', function (e) {
                alert('This is an "on" handler!');
            });

            $("body").on({
                click: function(e) {
                    alert( "Hello." );
                }
            }, "button");

            $("body").on( "click", "button", function(e) {
                alert( "Hello again." );
            });
        });
    </script>
</head>
<body>
<button id="clickButton">Uses jQuery Click</button>
<button id="bindButton">Uses jQuery Bind</button>
<button id="onButton">Uses jQuery On</button>
</body>
```

EventHandlers.html

```
<head>
    <title>Event Handlers</title>
    <script src="../../lib/jquery.js"></script>
    <script>
        $(document).ready(function () {
            $('#clickButton').click(function (e) {
                alert('This is a "click" handler!');
            });
            $('#bindButton').bind('click', function (e) {
                alert('This is a "bind" handler!');
            });
            $('#onButton').on('click', function (e) {
                alert('This is an "on" handler!');
            });

            $("body").on({
                click: function(e) {
                    alert( "Hello." );
                }
            }, "button");

            $("body").on( "click", "button", function(e) {
                alert( "Hello again." );
            });
        });
    </script>
</head>
<body>
<button id="clickButton">Uses jQuery Click</button>
<button id="bindButton">Uses jQuery Bind</button>
<button id="onButton">Uses jQuery On</button>
</body>
```

EventHandlers.html

```
$('#clickButton').click(function (e) {
    alert('This is a "click" handler!');
});
```

Select the button with the id clickButton

EventHandlers.html

```
$('#clickButton').click(function (e) {
    alert('This is a "click" handler!');
});
```

The click() function on the jQuery object adds a listener or event handler to the click event

EventHandlers.html

```
$('#clickButton').click(function (e) {
    alert('This is a "click" handler!');
});
```

The event handler code will be executed when the user clicks the button

EventHandlers.html

```
<head>
    <title>Event Handlers</title>
    <script src="../../lib/jquery.js"></script>
    <script>
        $(document).ready(function () {
            $('#clickButton').click(function (e) {
                alert('This is a "click" handler!');
            });
            $('#bindButton').bind('click', function (e) {
                alert('This is a "bind" handler!');
            });
            $('#onButton').on('click', function (e) {
                alert('This is an "on" handler!');
            });

            $("body").on({
                click: function(e) {
                    alert( "Hello." );
                }
            }, "button");

            $("body").on( "click", "button", function(e) {
                alert( "Hello again." );
            });
        });
    </script>
</head>
<body>
<button id="clickButton">Uses jQuery Click</button>
<button id="bindButton">Uses jQuery Bind</button>
<button id="onButton">Uses jQuery On</button>
</body>
```

EventHandlers.html

```
$('#bindButton').bind('click', function (e) {
  alert('This is a "bind" handler!');
});
```

The bind method on the jQuery object can also be used to add event handlers

EventHandlers.html

```
$('#bindButton').bind('click', function (e) {
  alert('This is a "bind" handler!');
});
```

You need to specify the event that you want to listen to, in this case the click event

EventHandlers.html

```
<head>
    <title>Event Handlers</title>
    <script src="../../lib/jquery.js"></script>
    <script>
        $(document).ready(function () {
            $('#clickButton').click(function (e) {
                alert('This is a "click" handler!');
            });
            $('#bindButton').bind('click', function (e) {
                alert('This is a "bind" handler!');
            });
            $('#onButton').on('click', function (e) {
                alert('This is an "on" handler!');
            });

            $("body").on({
                click: function(e) {
                    alert( "Hello." );
                }
            }, "button");

            $("body").on( "click", "button", function(e) {
                alert( "Hello again." );
            });
        });
    </script>
</head>
<body>
<button id="clickButton">Uses jQuery Click</button>
<button id="bindButton">Uses jQuery Bind</button>
<button id="onButton">Uses jQuery On</button>
</body>
```

EventHandlers.html

```
$('#onButton').on('click', function (e) {
  alert('This is an "on" handler!');
});
```

The on() method can also be used to listen for events

EventHandlers.html

```
$('#onButton').on('click', function (e) {
  alert('This is an "on" handler!');
});
```

This also requires you to specify the **type** of event that you want to listen to

EventHandlers.html

```
$('#onButton').on('click', function (e) {
  alert('This is an "on" handler!');
});
```

```
$('#clickButton').click(function (e) {
   alert('This is a "click" handler!');
});
```

```
$('#bindButton').bind('click', function (e) {
  alert('This is a "bind" handler!');
});
```

bind() and click() are just syntactic sugar which call the on() method under the hood

EventHandlers.html

```
$('#onButton').on('click', function (e) {
  alert('This is an "on" handler!');
});
```

```
$('#clickButton').click(function (e) {
   alert('This is a "click" handler!');
});
```

```
$('#bindButton').bind('click', function (e) {
  alert('This is a "bind" handler!');
});
```

Note that you can only listen to events on elements which exist on the DOM

EventHandlers.html

```
$('#onButton').on('click', function (e) {
  alert('This is an "on" handler!');
});
```

```
$('#clickButton').click(function (e) {
   alert('This is a "click" handler!');
});
```

```
$('#bindButton').bind('click', function (e) {
  alert('This is a "bind" handler!');
});
```

Not on elements which are yet to be created! ... that requires event delegation, we'll study that a little later

EventHandlers.html

```
$('#onButton').on('click', function (e) {
  alert('This is an "on" handler!');
});
```

```
$('#clickButton').click(function (e) {
   alert('This is a "click" handler!');
});
```

```
$('#bindButton').bind('click', function (e) {
  alert('This is a "bind" handler!');
});
```

Let's see how the demo works with only these 3 event handlers on the 3 buttons

EventHandlers.html

Example9Demo-1

EventHandlers.html

```
<head>
    <title>Event Handlers</title>
    <script src="../../lib/jquery.js"></script>
    <script>
        $(document).ready(function () {
            $('#clickButton').click(function (e) {
                alert('This is a "click" handler!');
            });
            $('#bindButton').bind('click', function (e) {
                alert('This is a "bind" handler!');
            });
            $('#onButton').on('click', function (e) {
                alert('This is an "on" handler!');
            });

            $("body").on({
                click: function(e) {
                    alert( "Hello." );
                }
            }, "button");

            $("body").on( "click", "button", function(e) {
                alert( "Hello again." );
            });
        });
    </script>
</head>
<body>
<button id="clickButton">Uses jQuery Click</button>
<button id="bindButton">Uses jQuery Bind</button>
<button id="onButton">Uses jQuery On</button>
</body>
```

EventHandlers.html

```
$("body").on({
    click: function(e) {
        alert( "Hello." );
    }
}, "button");
```

There are other ways to listen on an event

EventHandlers.html

```
$("body").on({
      click: function(e) {
            alert( "Hello." );
      }
}, "button");
```

A listener on the body will listen on click events on all specified elements

EventHandlers.html

```
$("body").on({
    click: function(e) {
        alert( "Hello." );
    }
}, "button");
```

We've specified the button element which means clicks on all buttons within the <body> will be handled

EventHandlers.html

```
$("body").on({
    click: function(e) {
        alert( "Hello." );
    }
}, "button");
```

Notice that we pass an object {} to the on() method

EventHandlers.html

```
$("body").on({
    click: function(e) {
        alert( "Hello." );
    }
}, "button");
```

click is a property on this object

EventHandlers.html

```
$("body").on({
    click: function(e) {
        alert( "Hello." );
    }
}, "button");
```

Which has the event handler code

EventHandlers.html

```
$("body").on({
    click: function(e) {
        alert( "Hello." );
    }
}, "button");
```

What if we had this event handler along with the 3 original handlers we saw earlier?

EventHandlers.html

```
$("body").on({
    click: function(e) {
        alert( "Hello." );
    }
}, "button");
```

Each of the buttons would now have 2 listeners on them

EventHandlers.html

```
$('#clickButton').click(function (e) {
    alert('This is a "click" handler!');
});

$('#bindButton').bind('click', function (e) {
  alert('This is a "bind" handler!');
});

$('#onButton').on('click', function (e) {
  alert('This is an "on" handler!');
});
```

```
$("body").on({
    click: function(e) {
        alert( "Hello." );
    }
}, "button");
```

Each of the buttons would now have 2 listeners handling click events

EventHandlers.html

```
$('#clickButton').click(function (e) {
    alert('This is a "click" handler!');
});

$('#bindButton').bind('click', function (e) {
  alert('This is a "bind" handler!');
});

$('#onButton').on('click', function (e) {
  alert('This is an "on" handler!');
});
```

```
$("body").on({
    click: function(e) {
        alert( "Hello." );
    }
}, "button");
```

One of these listeners

EventHandlers.html

```
$('#clickButton').click(function (e) {
    alert('This is a "click" handler!');
});

$('#bindButton').bind('click', function (e) {
  alert('This is a "bind" handler!');
});

$('#onButton').on('click', function (e) {
  alert('This is an "on" handler!');
});
```

```
$("body").on({
    click: function(e) {
        alert( "Hello." );
    }
}, "button");
```

And this listener which handles all buttons within the <body>

EventHandlers.html

Example9Demo-2

EventHandlers.html

```
<head>
    <title>Event Handlers</title>
    <script src="../../lib/jquery.js"></script>
    <script>
        $(document).ready(function () {
            $('#clickButton').click(function (e) {
                alert('This is a "click" handler!');
            });
            $('#bindButton').bind('click', function (e) {
                alert('This is a "bind" handler!');
            });
            $('#onButton').on('click', function (e) {
                alert('This is an "on" handler!');
            });

            $("body").on({
                click: function(e) {
                    alert( "Hello." );
                }
            }, "button");

            $("body").on( "click", "button", function(e) {
                alert( "Hello again." );
            });
        });
    </script>
</head>
<body>
<button id="clickButton">Uses jQuery Click</button>
<button id="bindButton">Uses jQuery Bind</button>
<button id="onButton">Uses jQuery On</button>
</body>
```

EventHandlers.html

```
$("body").on( "click", "button", function(e) {
    alert( "Hello again." );
});
```

An event handler on the <body> element can also be set up this way

EventHandlers.html

```
$("body").on( "click", "button", function(e) {
    alert( "Hello again." );
});
```

The event type and the kind of element to listen to is specified first

EventHandlers.html

```
$("body").on( "click", "button", function(e) {
    alert( "Hello again." );
});
```

This event handler listens to click events on all buttons within the <body>

JQUERY

The event object

The event object

```
$('#clickButton').click(function (e) {
    alert('This is a "click" handler!');
});
```

Notice that event handlers take in a single argument

The event object

```
$('#clickButton').click(function (e) {
    alert('This is a "click" handler!');
});
```

This is the event object which contains information about the event

The event object

This is the **event object** which
contains information about the event

What information does the event object have?

The event object

What information does the event object have?

type

target

timeStamp

which

data

pageX, pageY

nameSpace

The event object

type

The type of event which occurred e.g. click, mouseenter, mousedown etc

The event object

What information does the event object have?

type

target

timeStamp

which

data

pageX, pageY

nameSpace

The event object

target

The DOM element which fired the event e.g <a>, , <button> etc

The event object

What information does the event object have?

type

target

timeStamp

which

data

pageX, pageY

nameSpace

The event object

which

The button that was clicked or the key that was pressed to trigger the event

The event object

What information does the event object have?

type

target

timeStamp

which

data

pageX, pageY

nameSpace

The event object

pageX, pageY

The **mouse position** at the time the event occurred relative to the top left corner of the visible area

The event object

What information does the event object have?

type

target

timeStamp

which

data

pageX, pageY

nameSpace

The event object

timeStamp

The epoch time in milliseconds at which the event occurred. Epoch time is relative to Jan 1, 1970

The event object

What information does the event object have?

type

target

timeStamp

which

data

pageX, pageY

nameSpace

The event object

data

Any data that was passed in when the event was bound

The event object

What information does the event object have?

type

target

timeStamp

which

data

pageX, pageY

nameSpace

namespace

The namespace specified when the event was triggered

The event object

What information does the event object have?

type

target

timeStamp

which

data

pageX, pageY

nameSpace

The event object

type

target

which

timeStamp

pageX, pageY

data

nameSpace

What information does the event object have?

The event object also has 2 functions

The event object

type

target

which

timeStamp

pageX, pageY

data

nameSpace

The event object also has 2 functions

preventdefault()

stopPropagation()

The event object

preventdefault()

Prevent the default action of the event i.e. opening up a link

The event object

type

target

which

timeStamp

pageX, pageY

data

nameSpace

The event object also has 2 functions

preventDefault()

stopPropagation()

The event object

stopPropagation()

Stop the event **bubbling** up to other events

This is used to indicate that the event **has been handled** by the current handler and should not be passed on

The event object

type

target

which

timeStamp

pageX, pageY

data

nameSpace

The event object also has 2 functions

preventDefault()

stopPropagation()

EXAMPLE 10

EventObject.html

EventObject.html

```
<table>
    <tr>
        <td> Cuthbert Binns </td>
        <td> History Of Magic </td>
    </tr>
    <tr>
        <td> Charity Burbage </td>
        <td>
            <a href="https://evil.com" target="_blank">
                Muggle Studies
            </a>
        </td>
    </tr>
    <tr>
        <td> Firenze </td>
        <td>
            <a href="https://en.wikipedia.org/wiki/Magic_in_Harry_Potter#Divination"
               target="_blank">
                Divination
            </a>
        </td>
    </tr>
    <tr>
        <td> Pomona Sprout </td>
        <td> Herbology </td>
    </tr>
    <tr>
        <td> Rubeus Hagrid </td>
        <td>
            <a href="https://en.wikipedia.org/wiki/Magic_in_Harry_Potter#Care_of_Magical_Creatures"
               target="_blank">
                Care Of Magical Creatures
            </a>
        </td>
    </tr>
    <tr>
        <td> Aurora Sinistra </td>
        <td> Astronomy </td>
    </tr>
</table>
```

Cuthbert Binns	History Of Magic
Charity Burbage	Muggle Studies
Firenze	Divination
Pomona Sprout	Herbology
Rubeus Hagrid	Care Of Magical Creatures
Aurora Sinistra	Astronomy

EventObject.html

```
    <td> Cuthbert Binns </td>
    <td> History Of Magic </td>
</tr>
<tr>
    <td> Charity Burbage </td>
    <td>
        <a href="https://evil.com" target="_blank">
            Muggle Studies
        </a>
    </td>
</tr>
<tr>
    <td> Firenze </td>
    <td>
        <a href="https://en.wikipedia.org/wiki/Magic_in_Harry_Potter#Divination"
           target="_blank">
            Divination
        </a>
    </td>
</tr>
<tr>
    <td> Pomona Sprout </td>
    <td> Herbology </td>
</tr>
<tr>
    <td> Rubeus Hagrid </td>
    <td>
        <a href="https://en.wikipedia.org/wiki/Magic_in_Harry_Potter#Care_of_Magical_Creatures"
           target="_blank">
            Care Of Magical Creatures
        </a>
    </td>
</tr>
<tr>
    <td> Aurora Sinistra </td>
```

The Muggle Studies link points to a malicious website

This page should not open up this link

EventObject.html

Example10Demo

EventObject.html

```
$('a').click(function (e) {
    console.log("Event Type: " + e.type);
    console.log("Button or key pressed: " + e.which);
    console.log("Link clicked " + e.target.href);

    if (this.hostname.indexOf('evil') != -1) {
        e.preventDefault();
    }
});
```

Add a click event handler to any links on this page

EventObject.html

```
$('a').click(function (e) {
    console.log("Event Type: " + e.type);
    console.log("Button or key pressed: " + e.which);
    console.log("Link clicked " + e.target.href);

    if (this.hostname.indexOf('evil') != -1) {
        e.preventDefault();
    }
});
```

Log the type of event, in this case it was a click event

EventObject.html

```
$('a').click(function (e) {
    console.log("Event Type: " + e.type);
    console.log("Button or key pressed: " + e.which);
    console.log("Link clicked " + e.target.href);

    if (this.hostname.indexOf('evil') != -1) {
        e.preventDefault();
    }
});
```

The trigger button or key which was pressed to fire this event

EventObject.html

```
$('a').click(function (e) {
    console.log("Event Type: " + e.type);
    console.log("Button or key pressed: " + e.which);
    console.log("Link clicked " + e.target.href);

    if (this.hostname.indexOf('evil') != -1) {
        e.preventDefault();
    }
});
```

Access the element which fired the event, we know it is a link so has a valid href

EventObject.html

```
$('a').click(function (e) {
    console.log("Event Type: " + e.type);
    console.log("Button or key pressed: " + e.which);
    console.log("Link clicked " + e.target.href);

    if (this.hostname.indexOf('evil') != -1) {
        e.preventDefault();
    }
});
```

"this" within an event handler refers to the element which fired the event

EventObject.html

```
$('a').click(function (e) {
    console.log("Event Type: " + e.type);
    console.log("Button or key pressed: " + e.which);
    console.log("Link clicked " + e.target.href);

    if (this.hostname.indexOf('evil') != -1) {
        e.preventDefault();
    }
});
```

If the link points to a host which has "evil" in its name

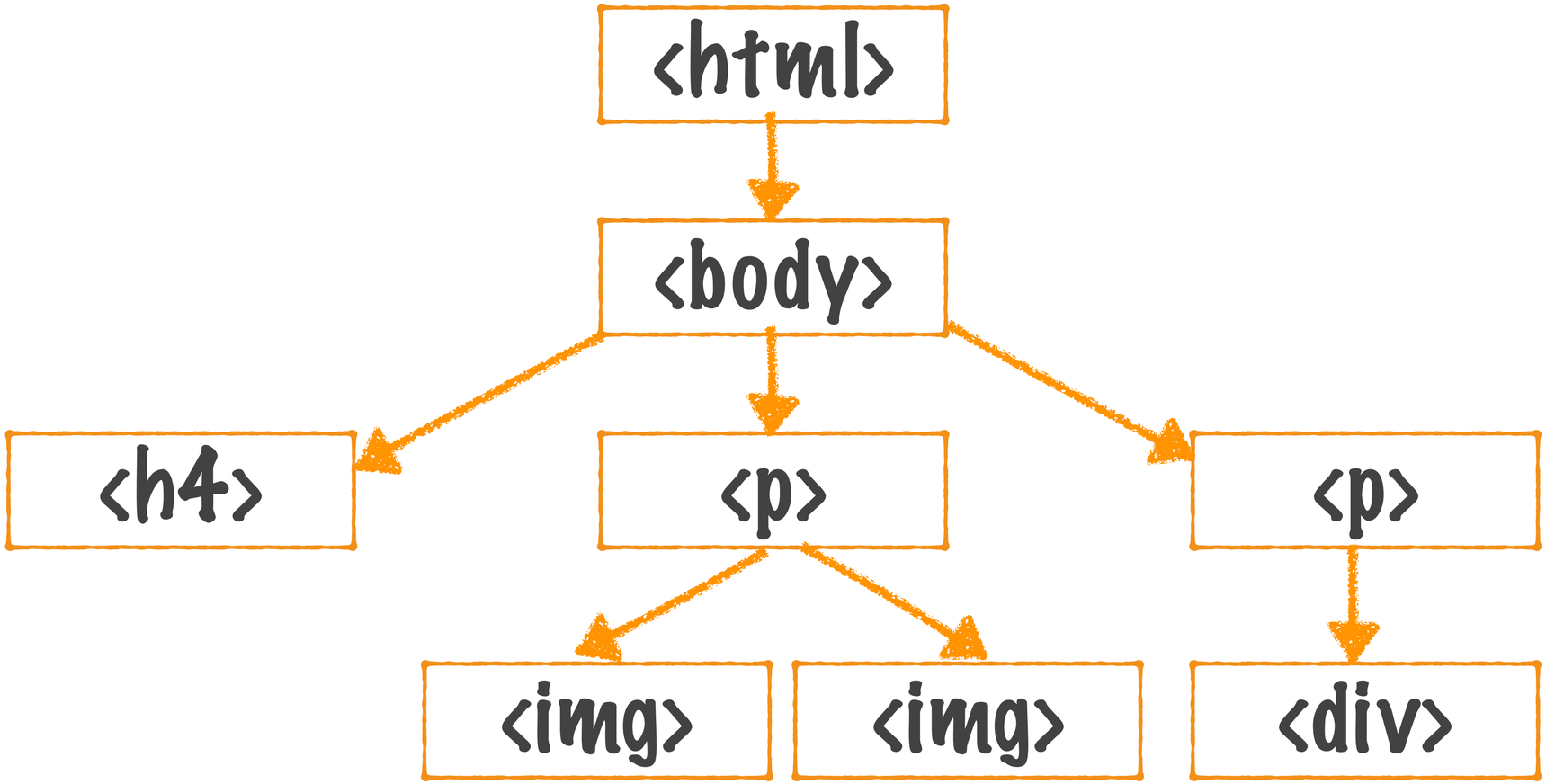
EventObject.html

```
$('a').click(function (e) {
    console.log("Event Type: " + e.type);
    console.log("Button or key pressed: " + e.which);
    console.log("Link clicked " + e.target.href);

    if (this.hostname.indexOf('evil') != -1) {
        e.preventDefault();
    }
});
```

Prevent the default action of the link i.e do not open up the page

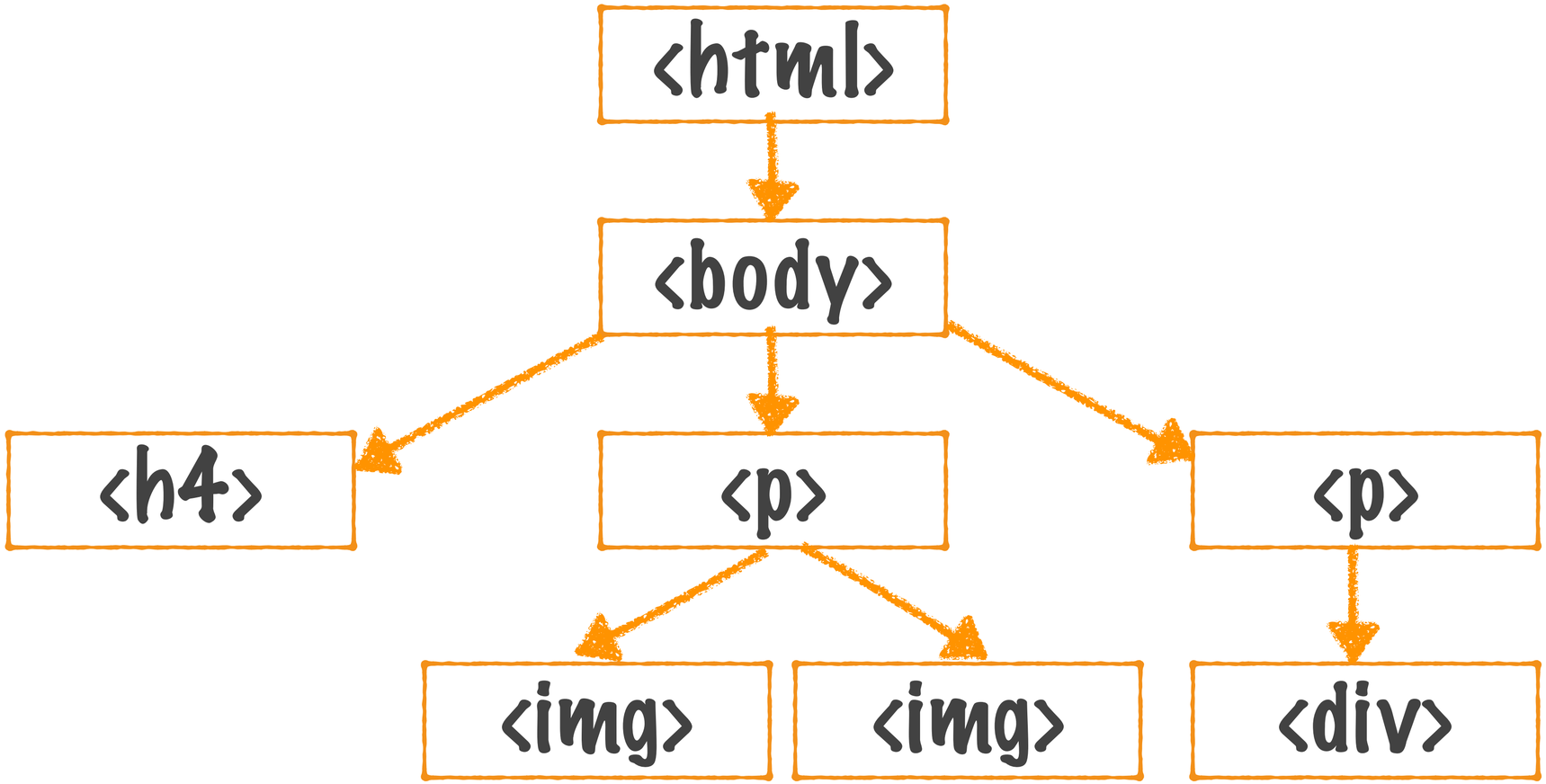
JQUERY

Capture and bubble phase



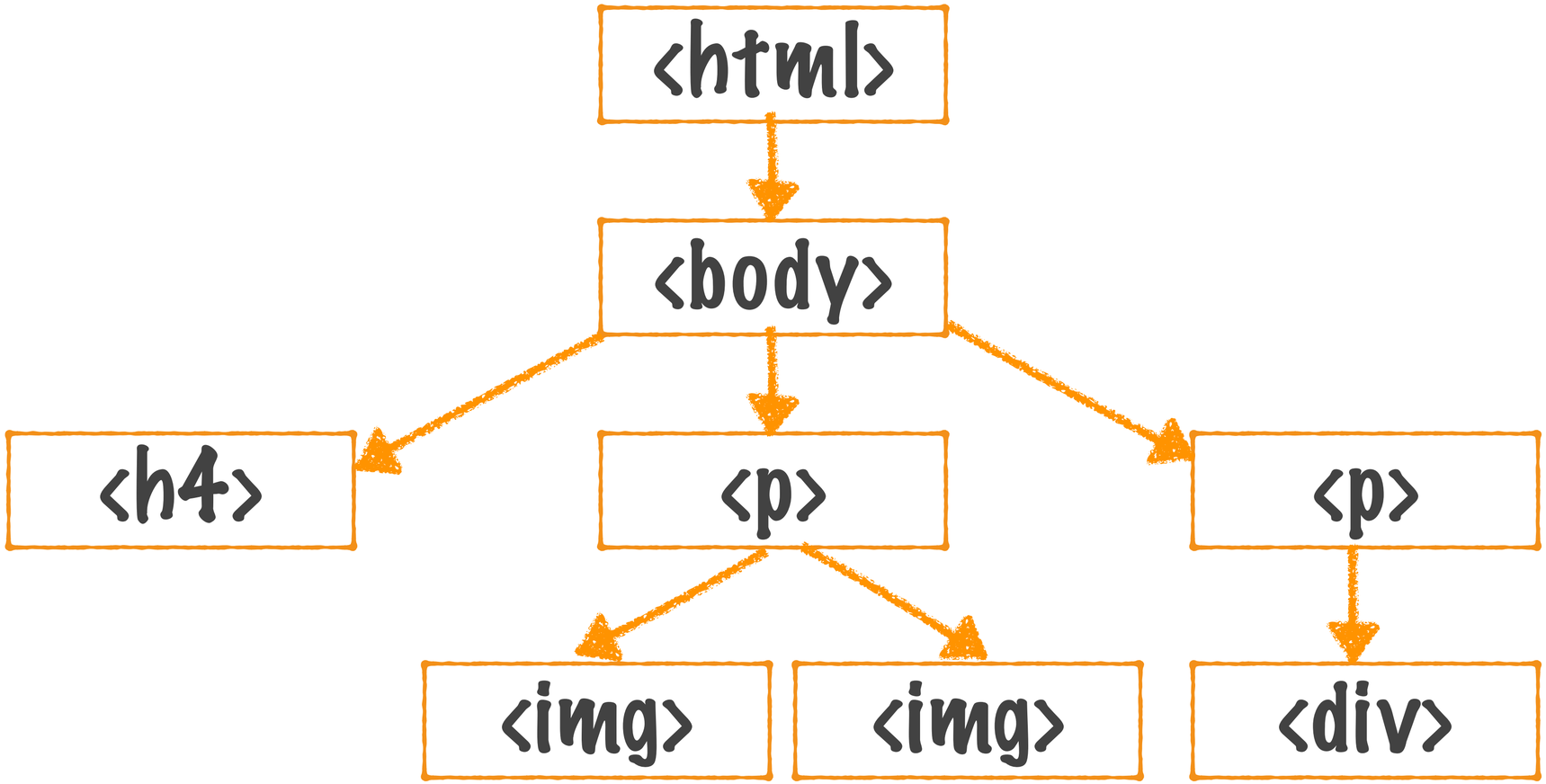
A DOM hierarchy

Capture and bubble phase



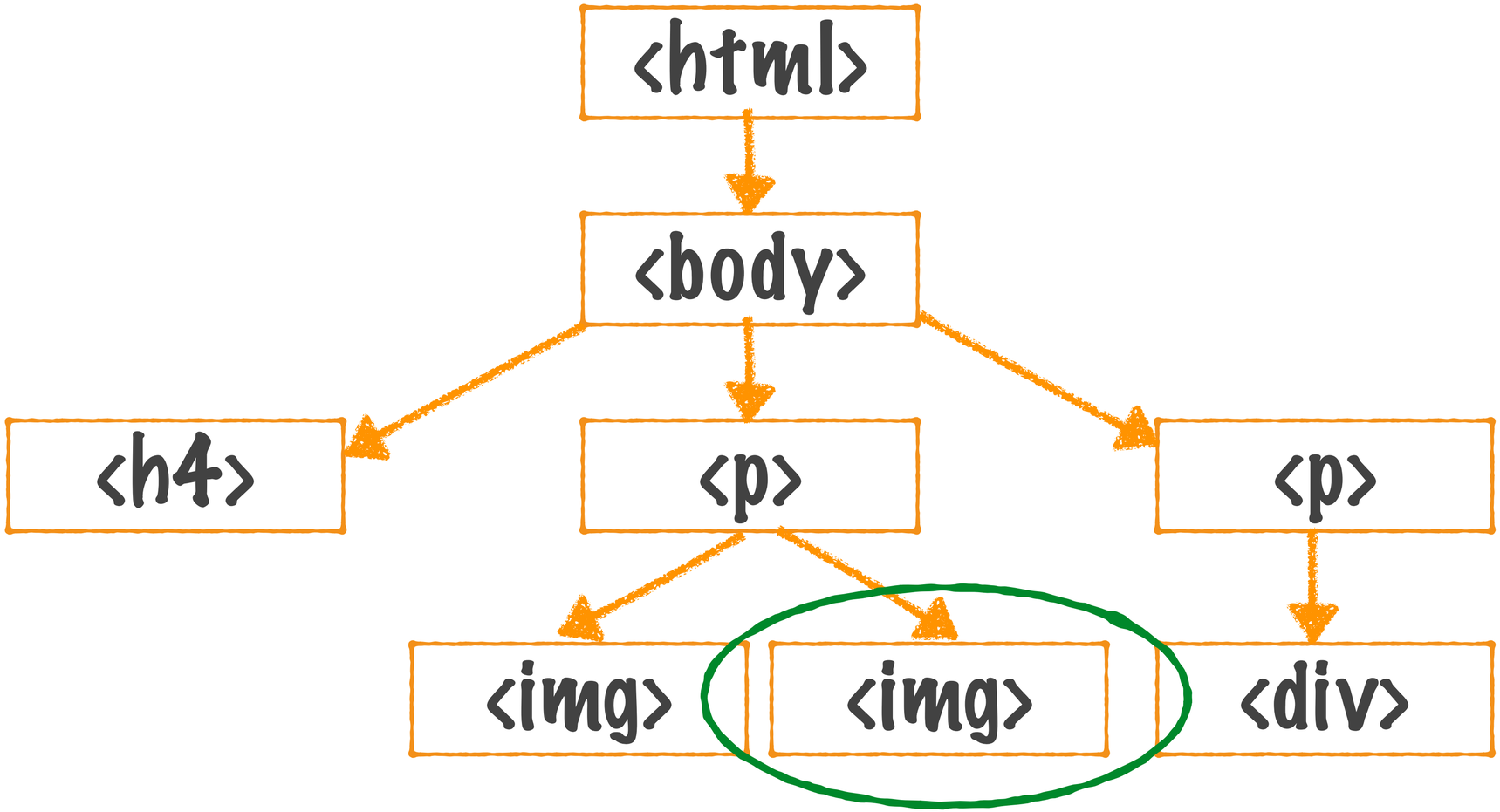
You can hook up event handlers to elements at different levels in the hierarchy

Capture and bubble phase



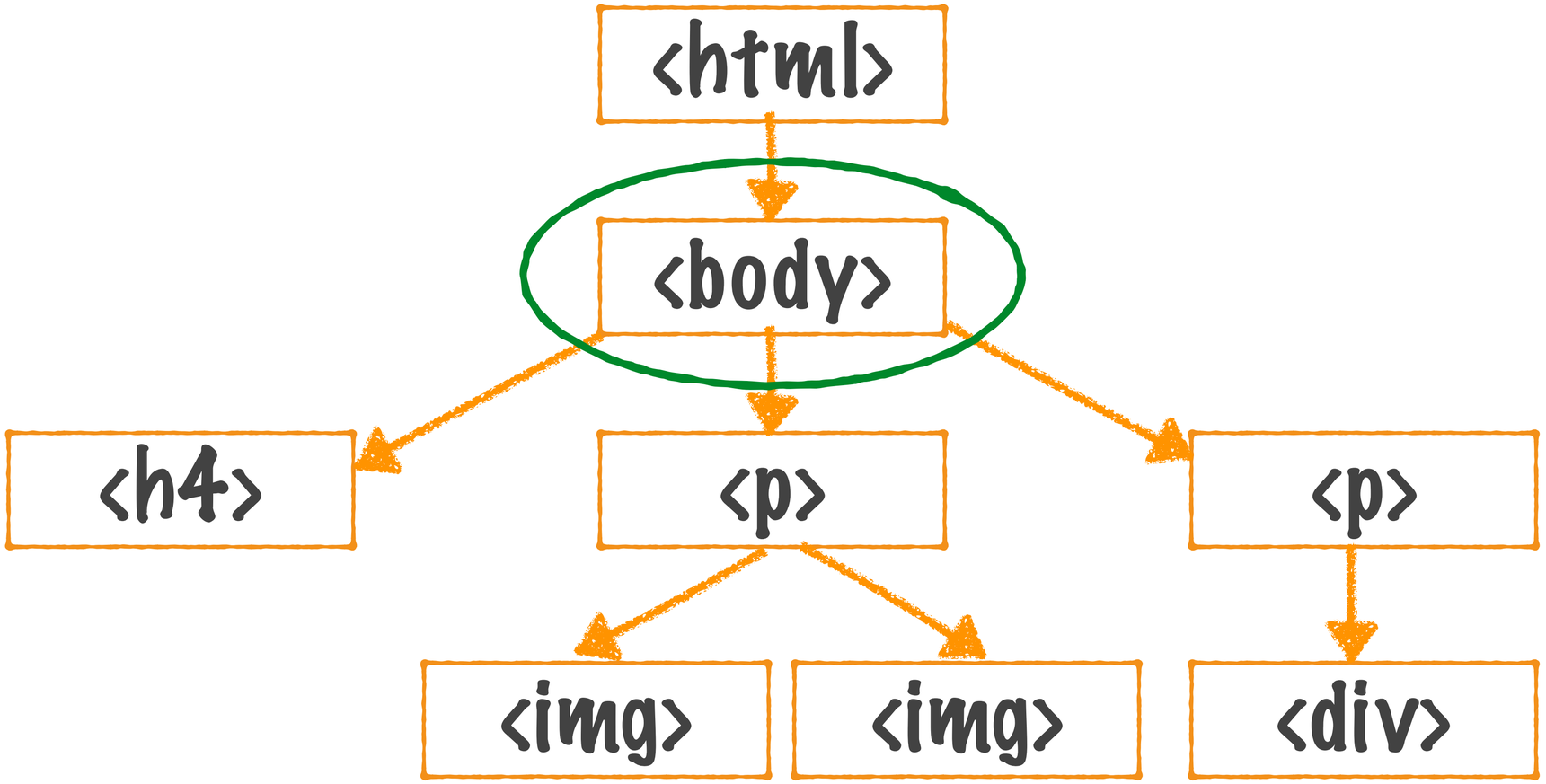
And events can be fired by any element

Capture and bubble phase



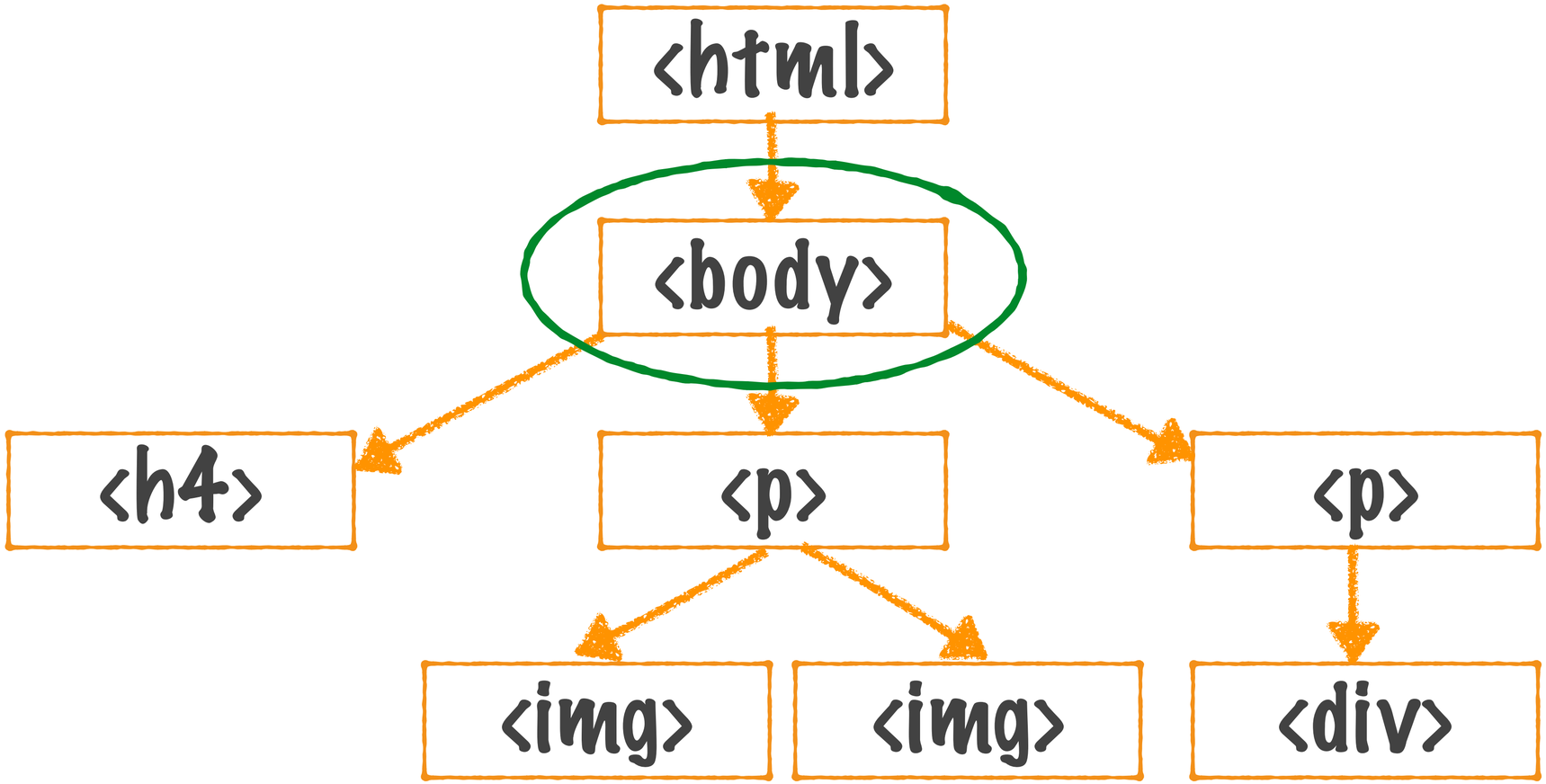
By the leaf elements

Capture and bubble phase



Or an interior element which has children as well as ancestors

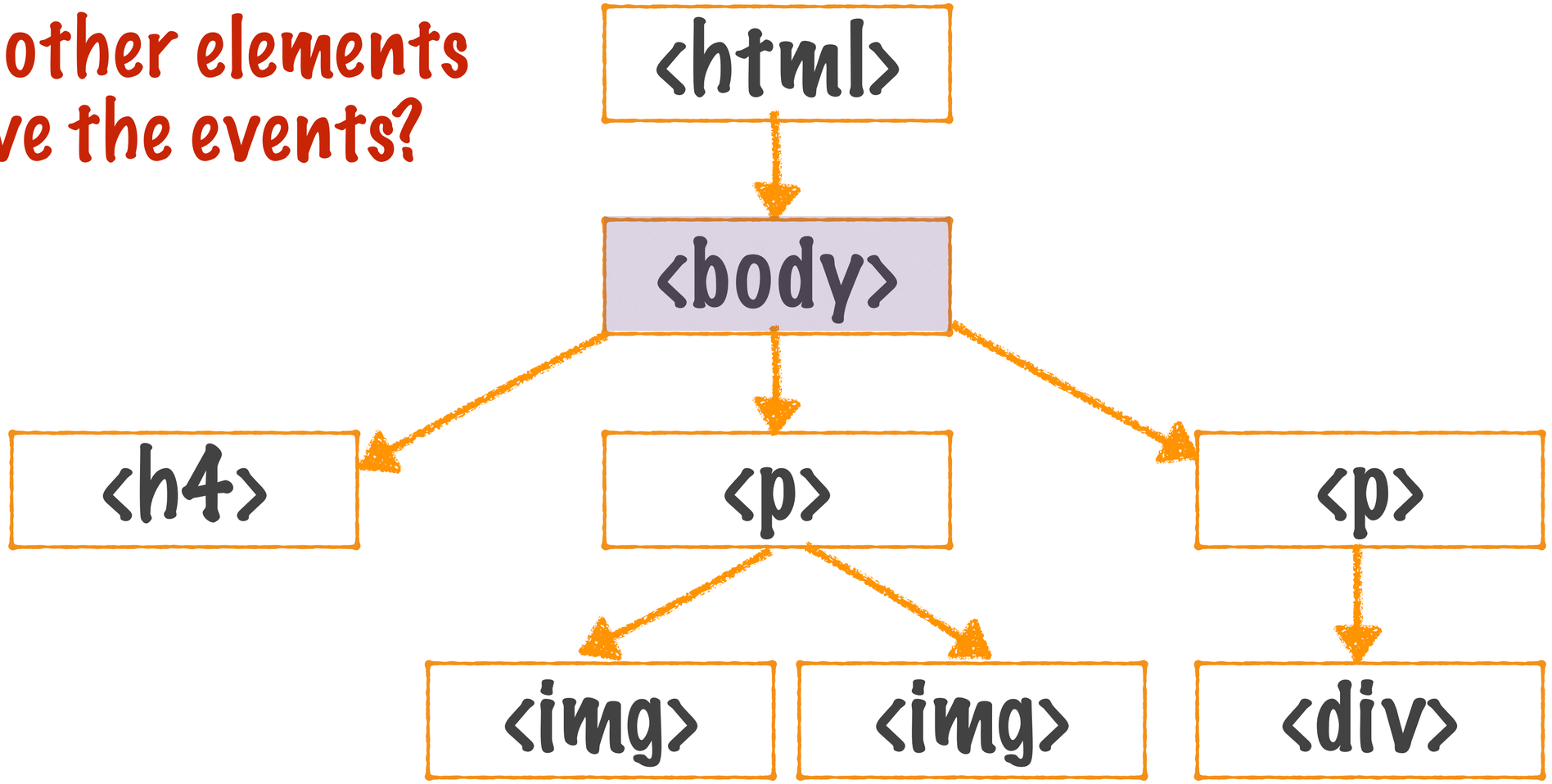
Capture and bubble phase



Which other elements receive the events?

Capture and bubble phase

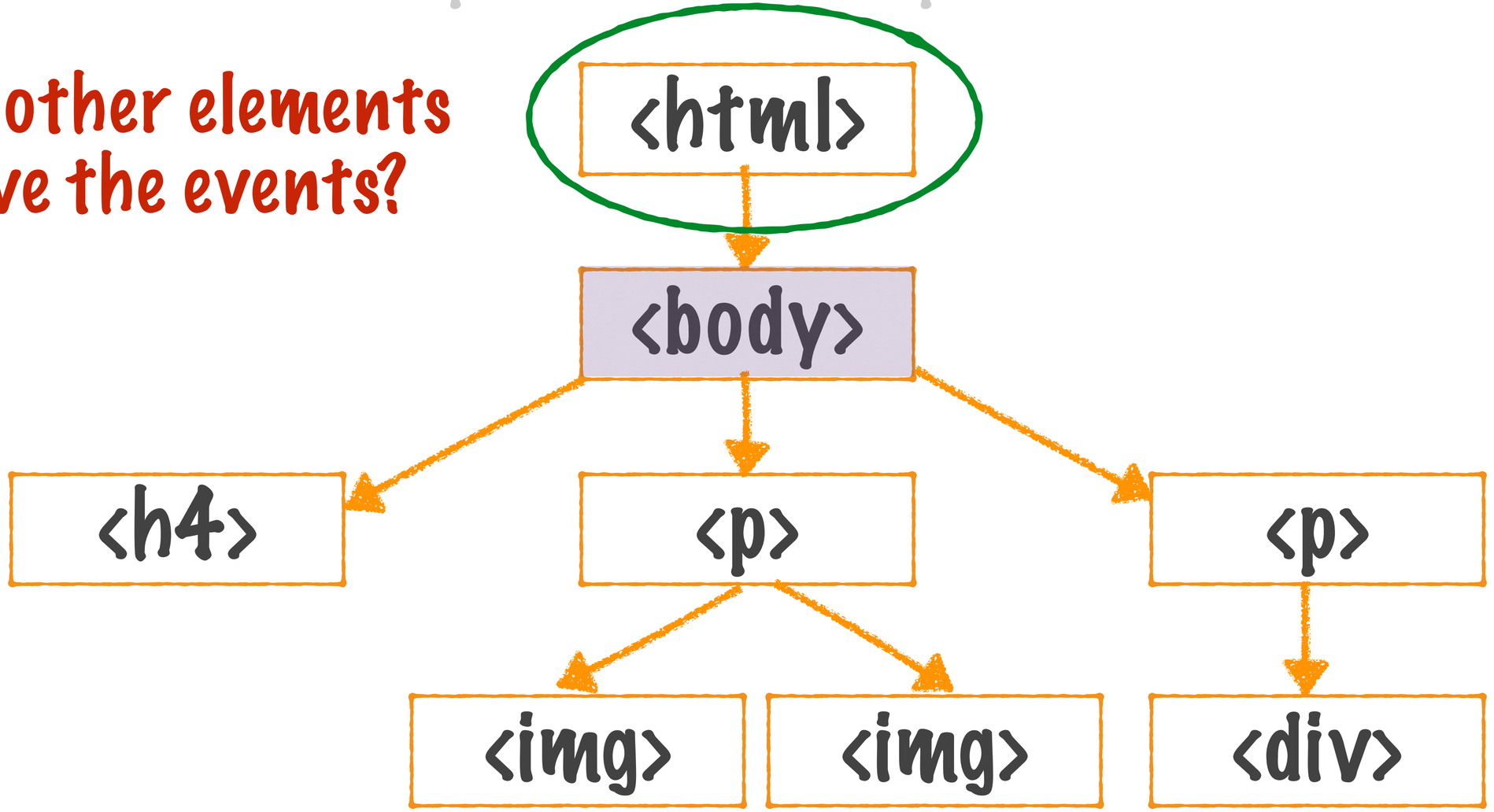
Which other elements receive the events?



The <body> element is where the event is triggered

Capture and bubble phase

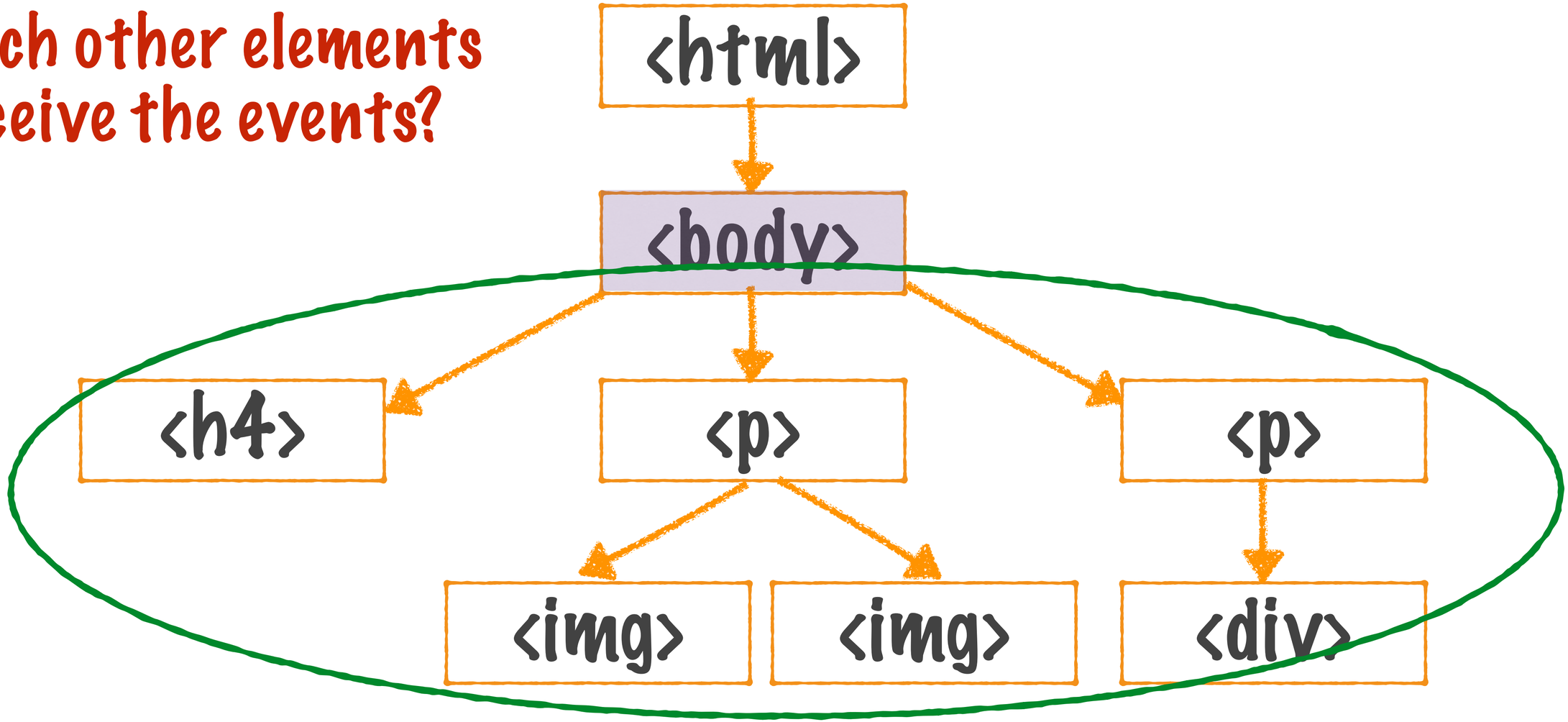
Which other elements receive the events?



You could argue that the <html> encompasses the <body> and so should receive the event as well

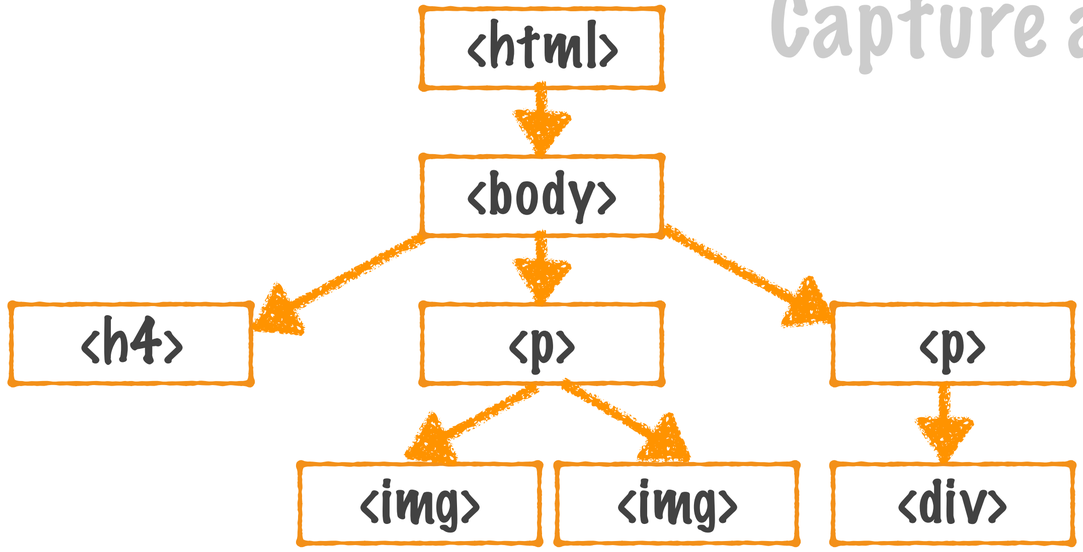
Capture and bubble phase

Which other elements receive the events?



Or you could argue that the <body> encompasses its child elements and they should receive the event

Capture and bubble phase



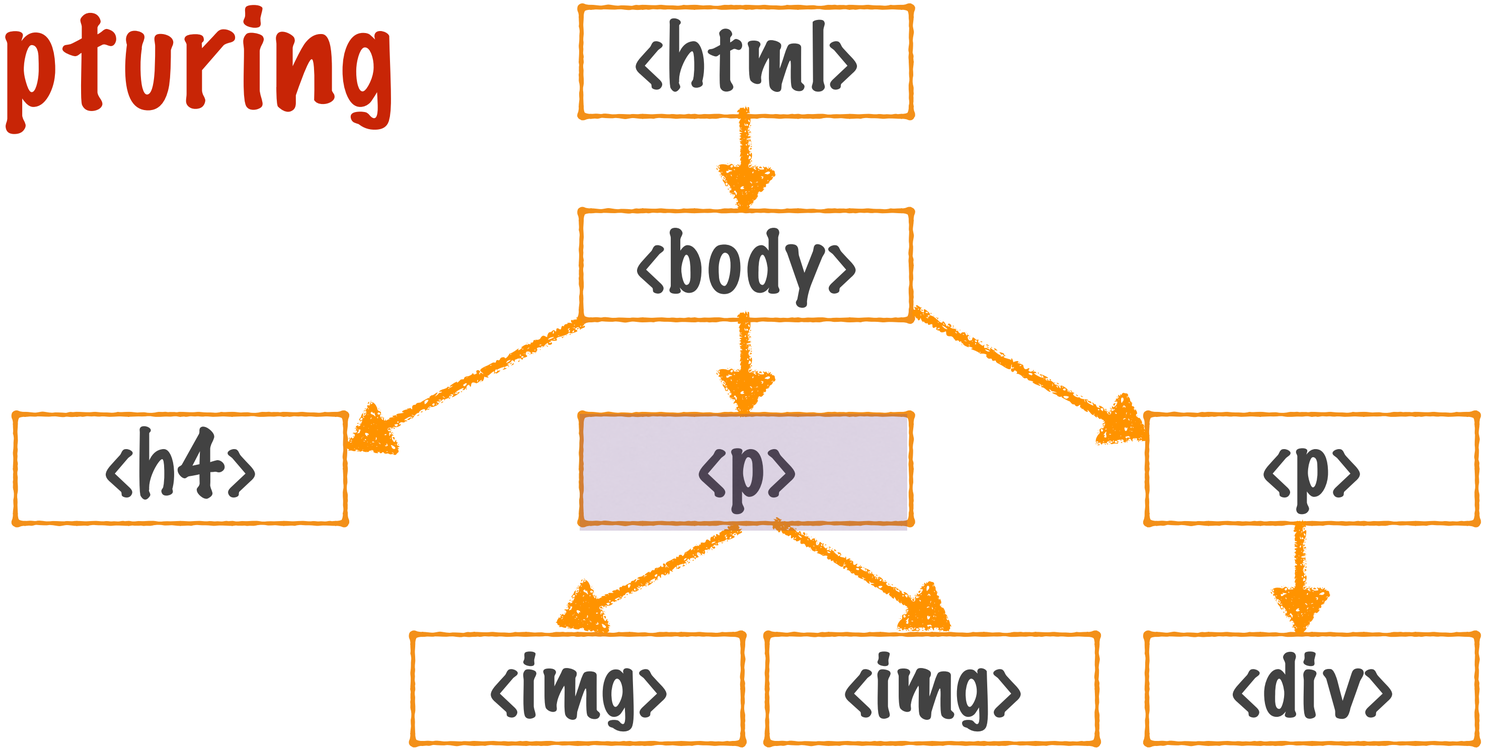
Which other elements receive the events?

These 2 ways in which events can travel are called:

Event Capturing

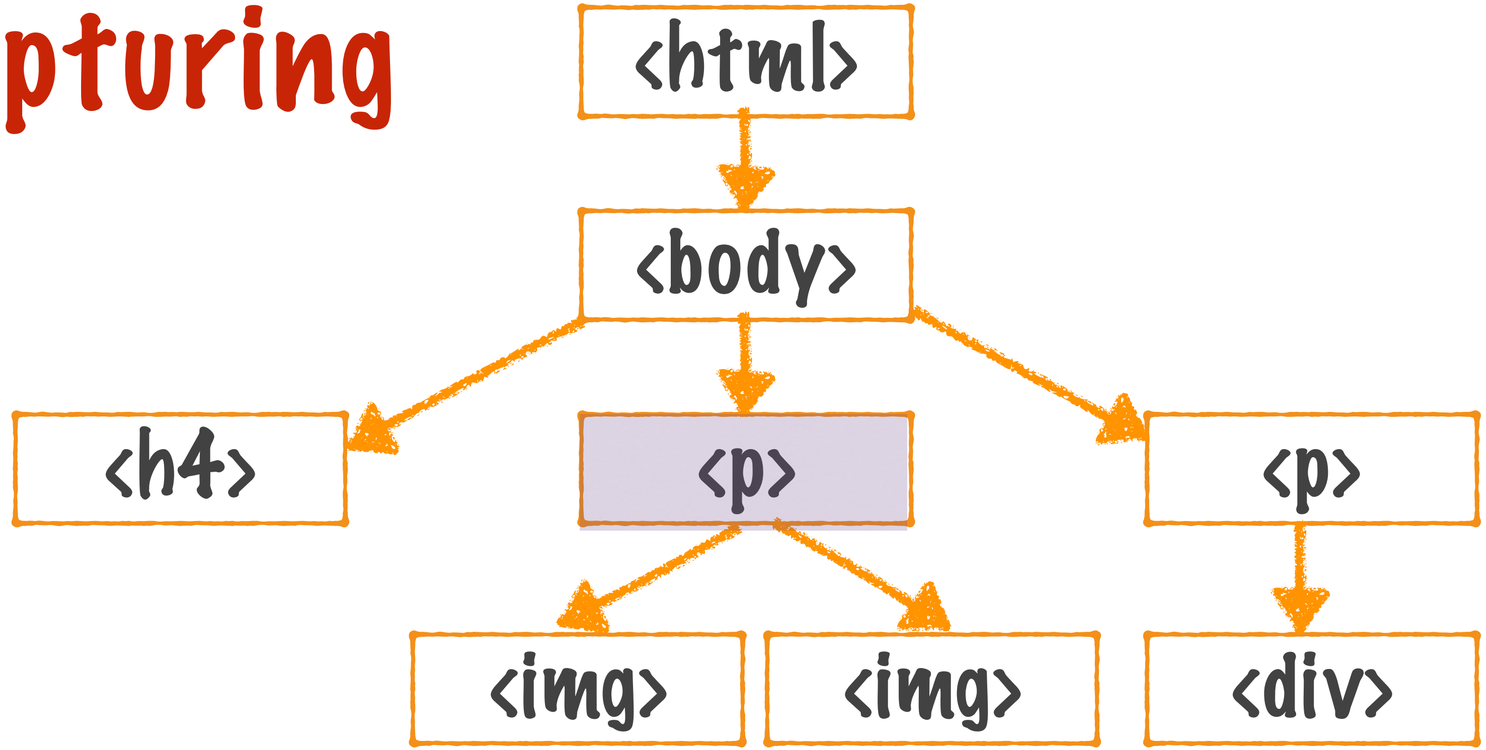
Event Bubbling

Event Capturing



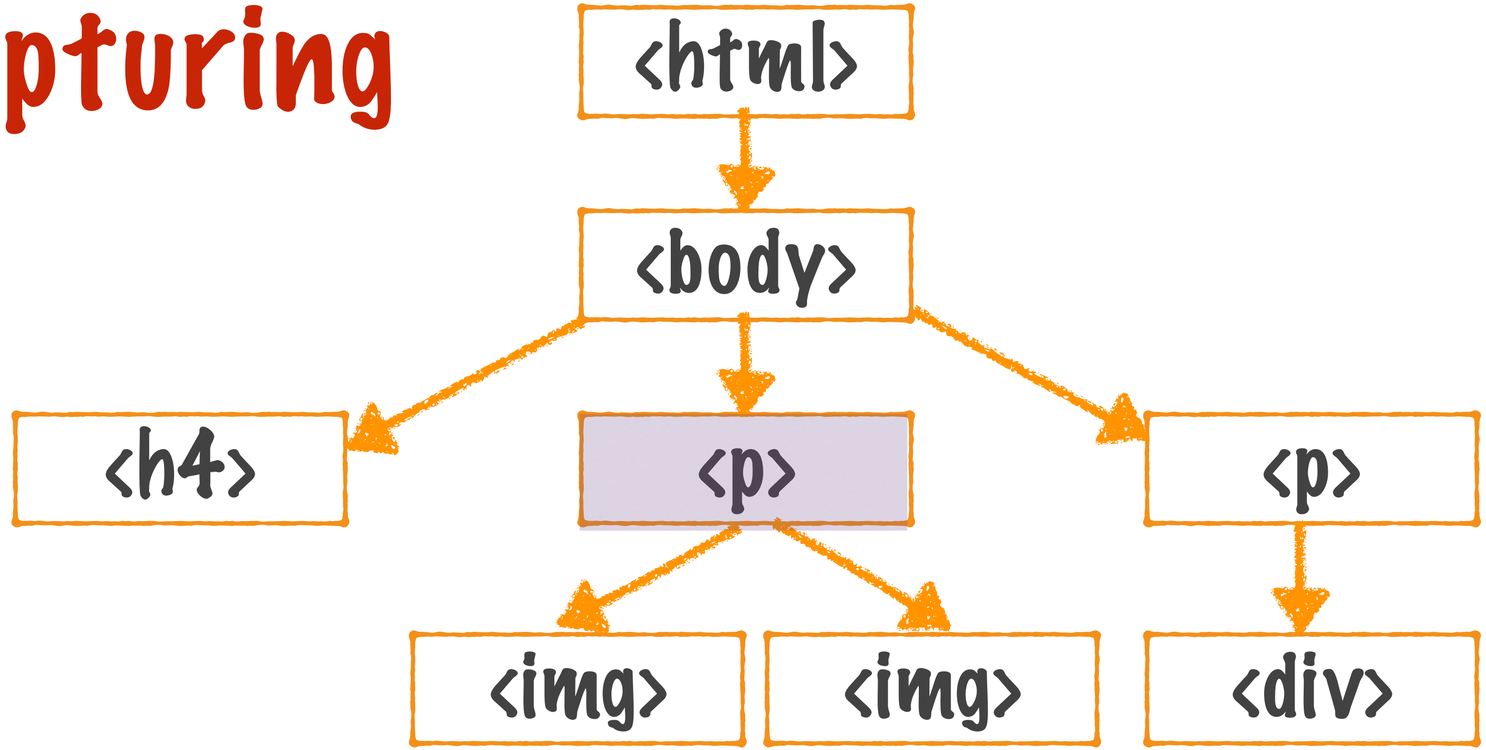
Element <p> is clicked on

Event Capturing



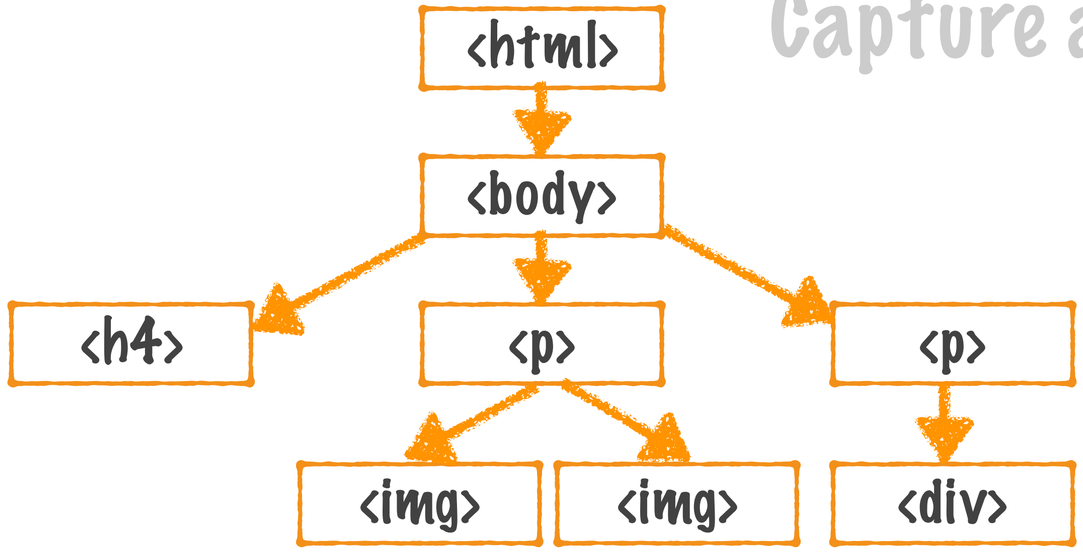
It gets a chance to react to the event

Event Capturing



The click is then propagated downwards to child elements

Capture and bubble phase



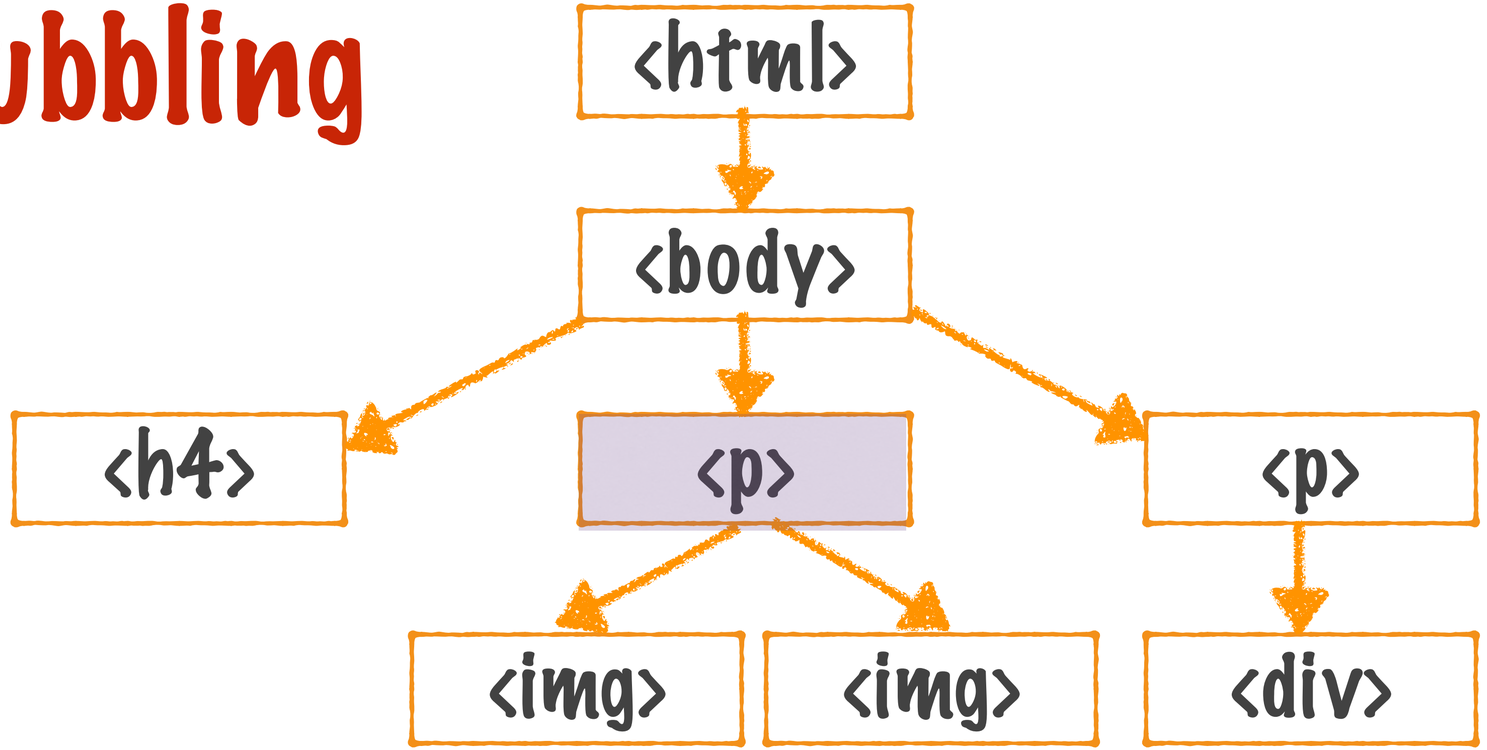
Which other elements receive the events?

These 2 ways in which events can travel are called:

Event Capturing

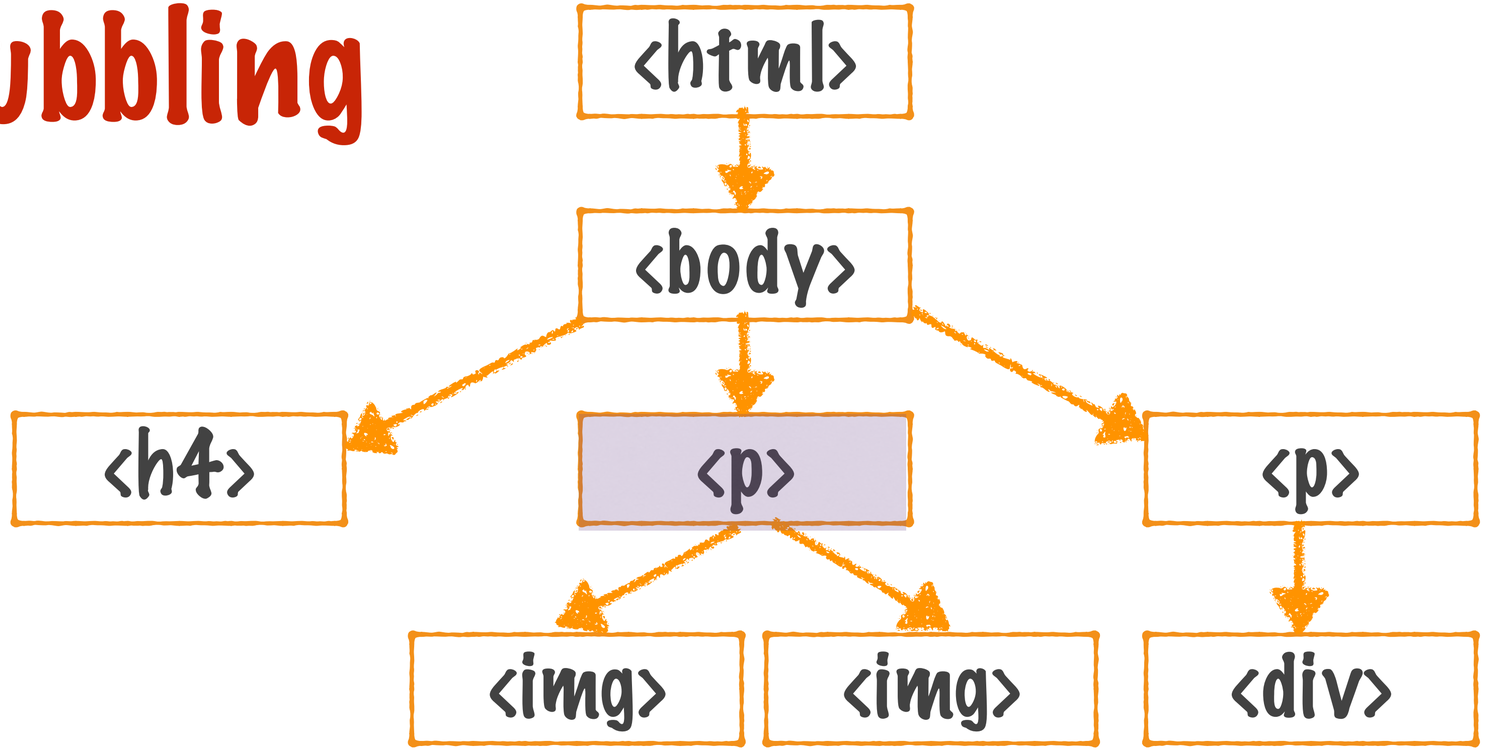
Event Bubbling

Event Bubbling



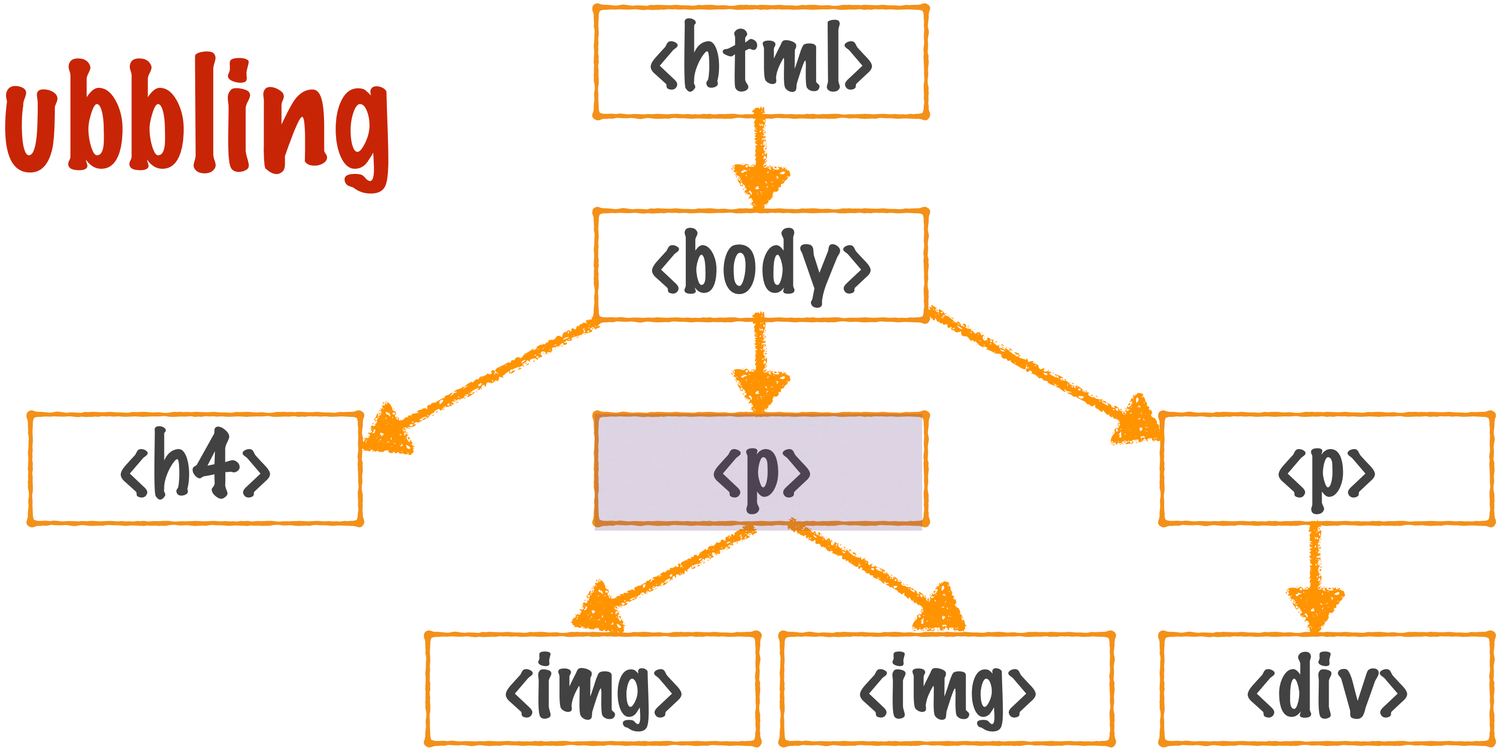
Element <p> is clicked on

Event Bubbling



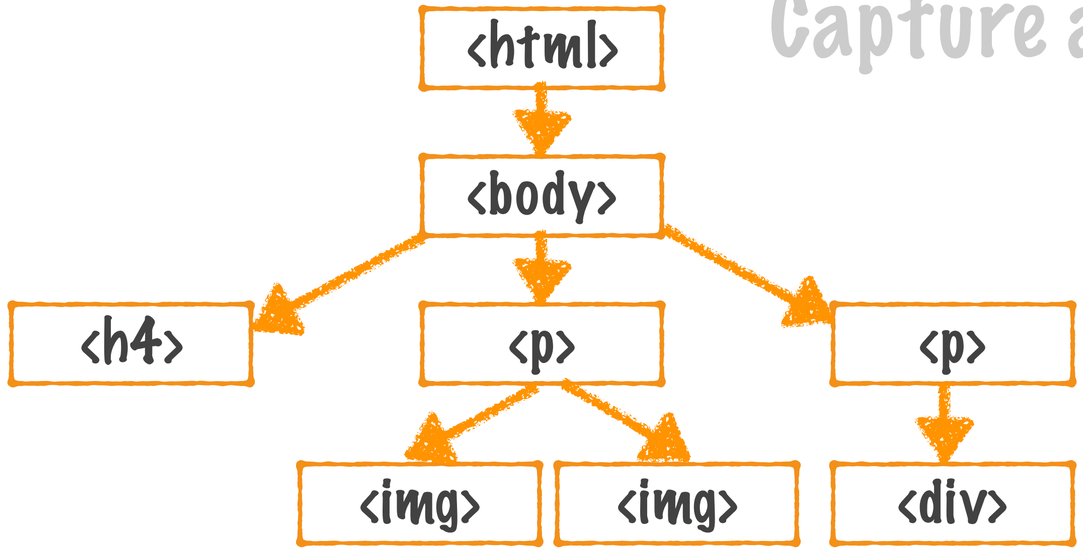
It gets a chance to react to the event

Event Bubbling



This click is them propagated upwards to parent elements

Capture and bubble phase



Which other elements receive the events?

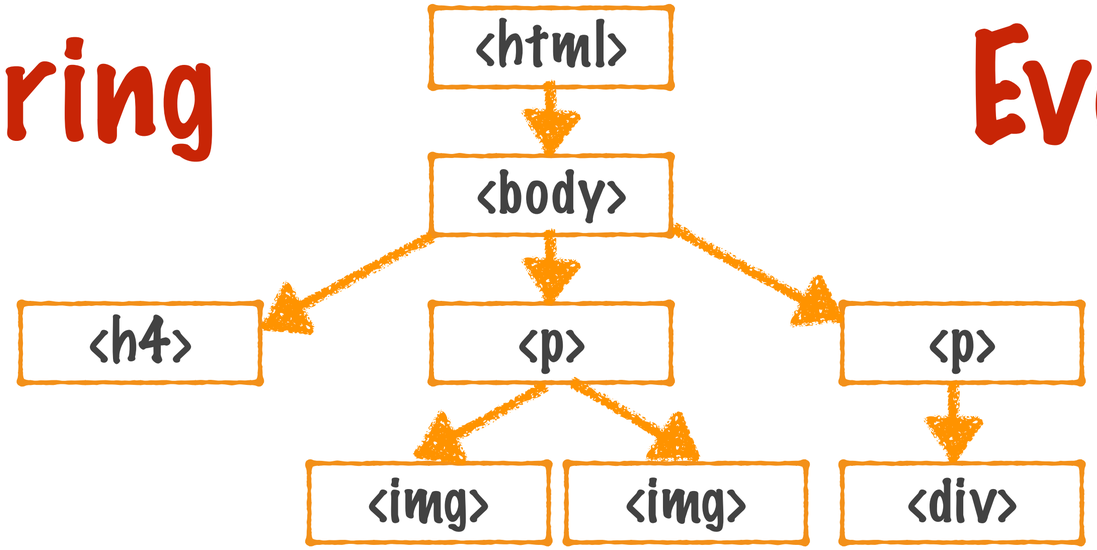
These 2 ways in which events can travel are called:

Event Capturing

Event Bubbling

Capture and bubble phase

Event Capturing



Event Bubbling

So which of these methods do browsers use?

Both!

Capture and bubble phase

So which of these methods do browsers use?

Event Capturing

Event Bubbling

Different browsers originally decided on different models for event propagation

Capture and bubble phase

So which of these methods do browsers use?

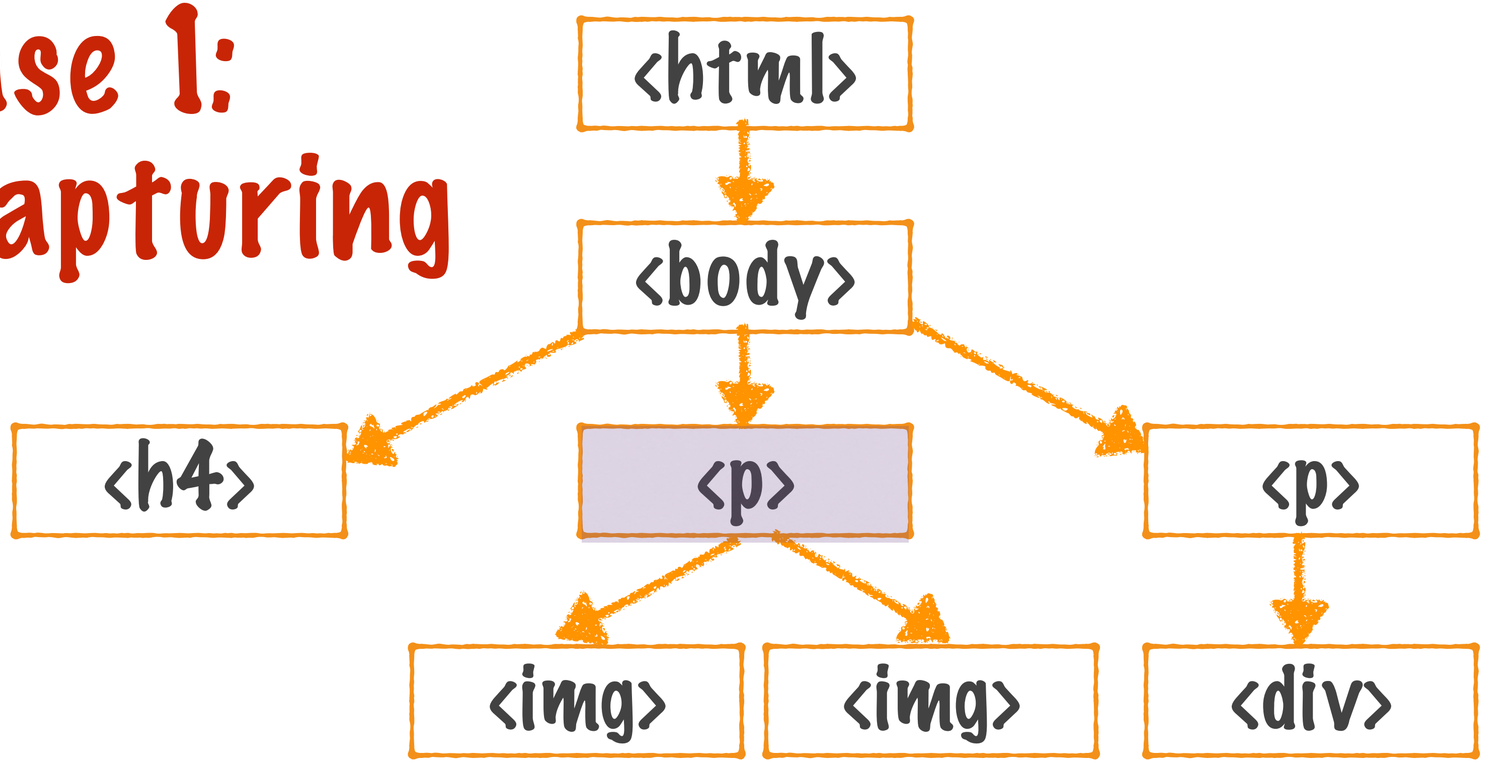
Event Capturing

Event Bubbling

The DOM standard that was ultimately adopted was that both should be supported

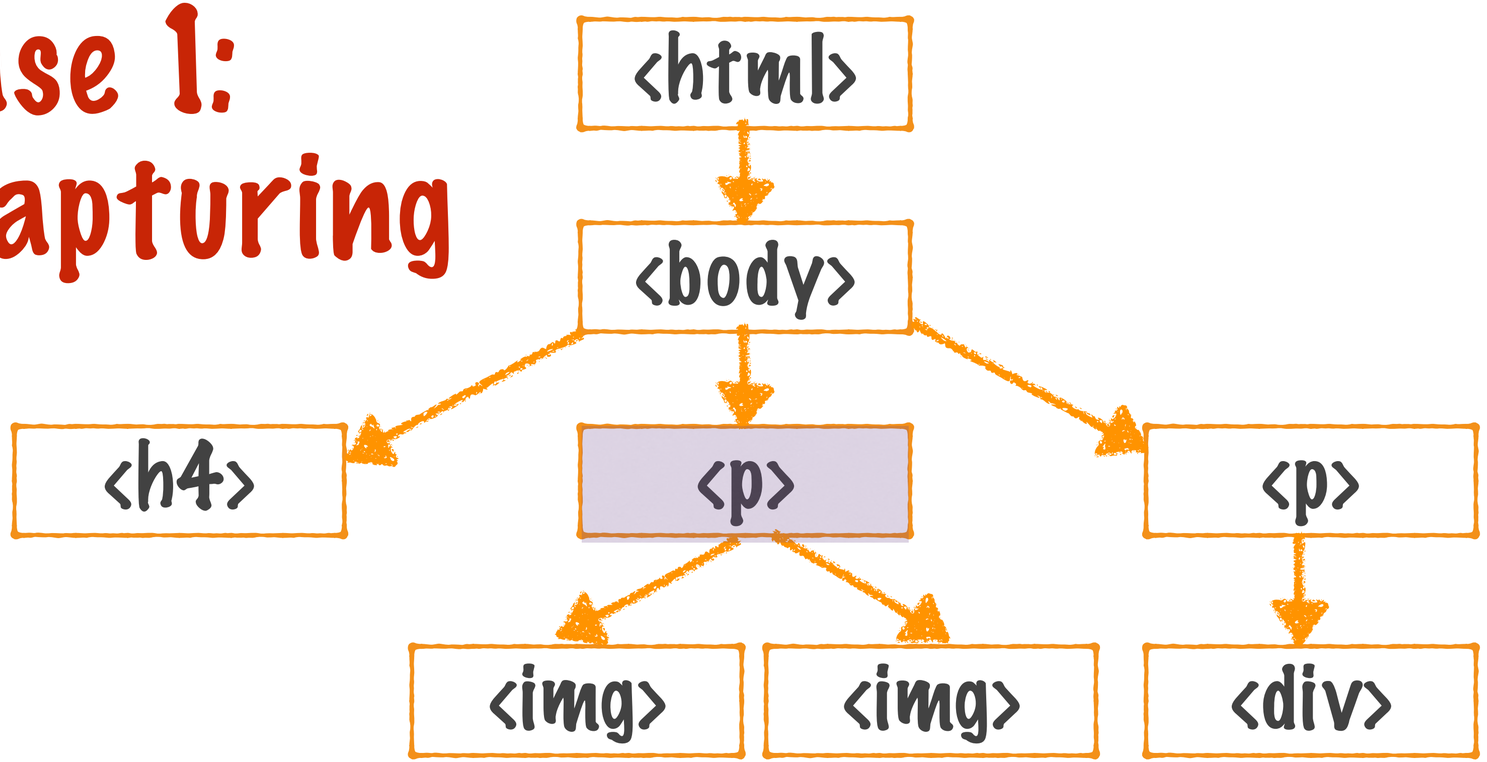
How does that work?

Phase 1: Event Capturing



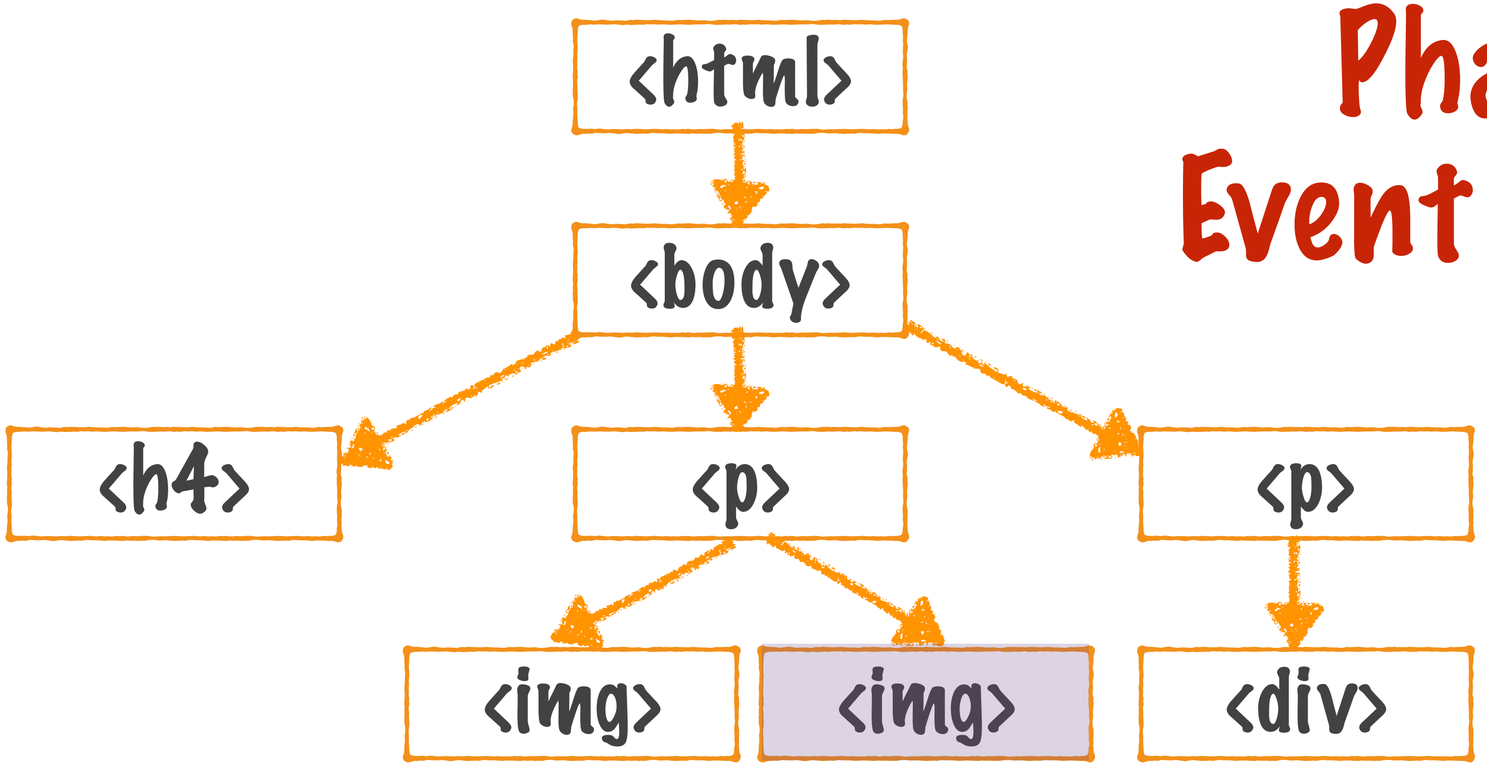
Element <p> is clicked on

Phase 1: Event Capturing



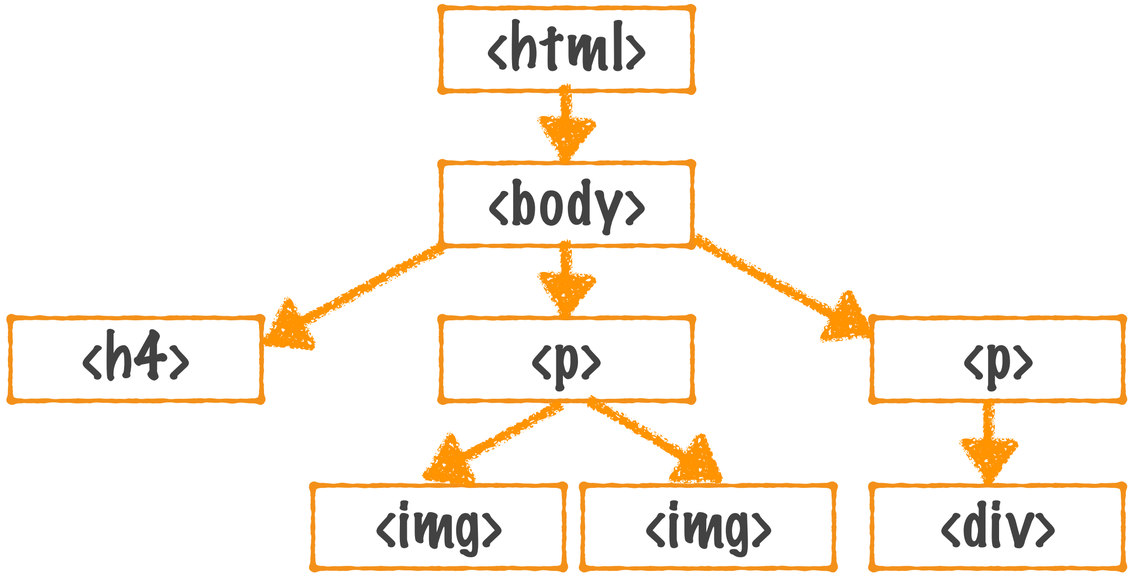
It is propagated downwards to child elements

Phase 2: Event Bubbling



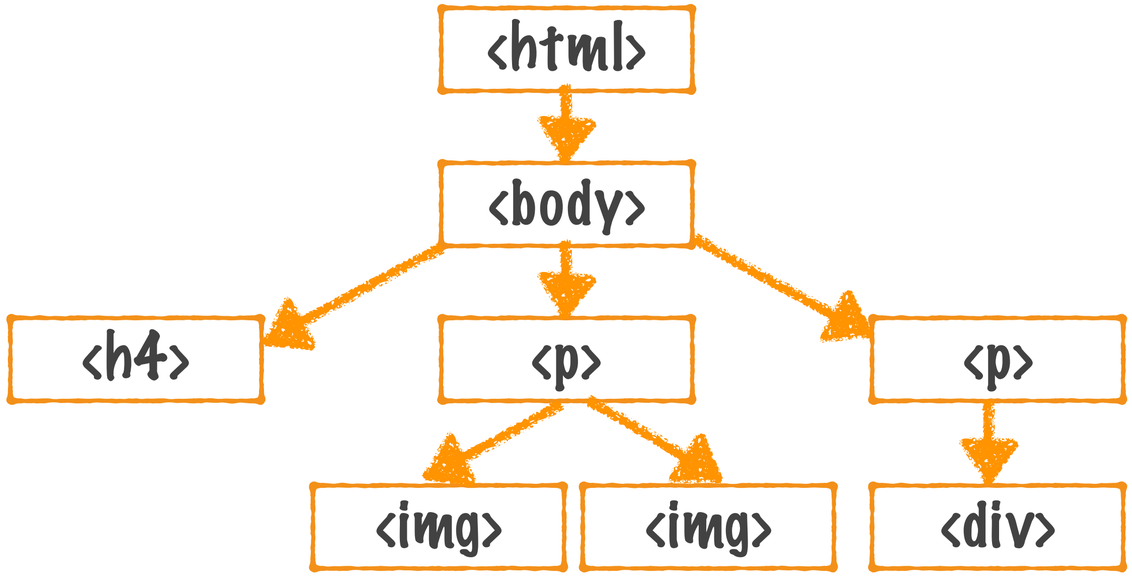
And then propagated upwards to parent elements

Capture and bubble phase



This can cause some strange behavior while handling events

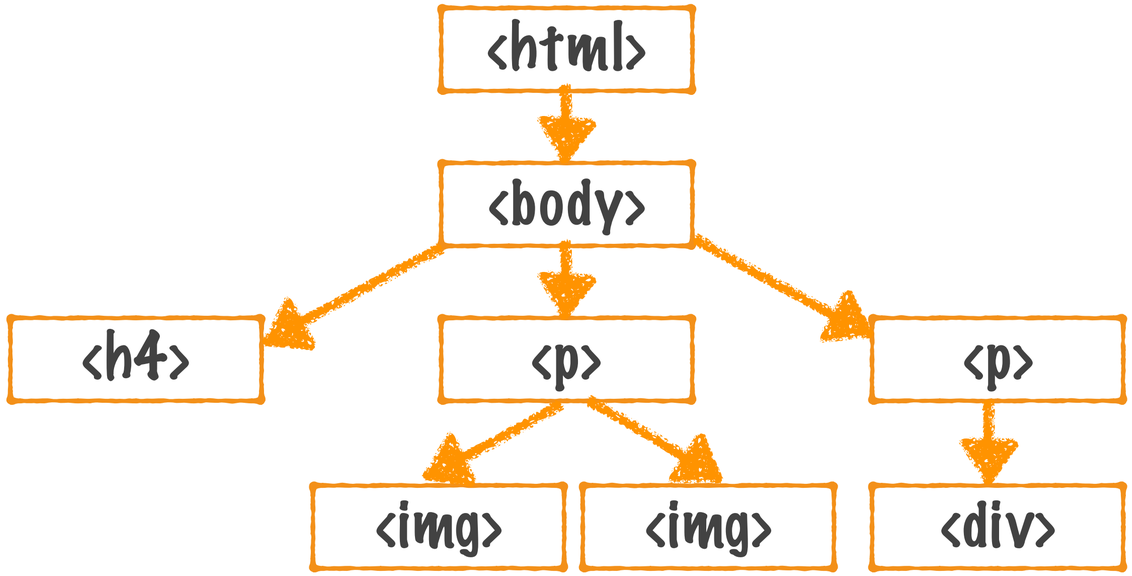
Capture and bubble phase



The <body> element will receive events from its children as well as its ancestors

When should the event be handled?

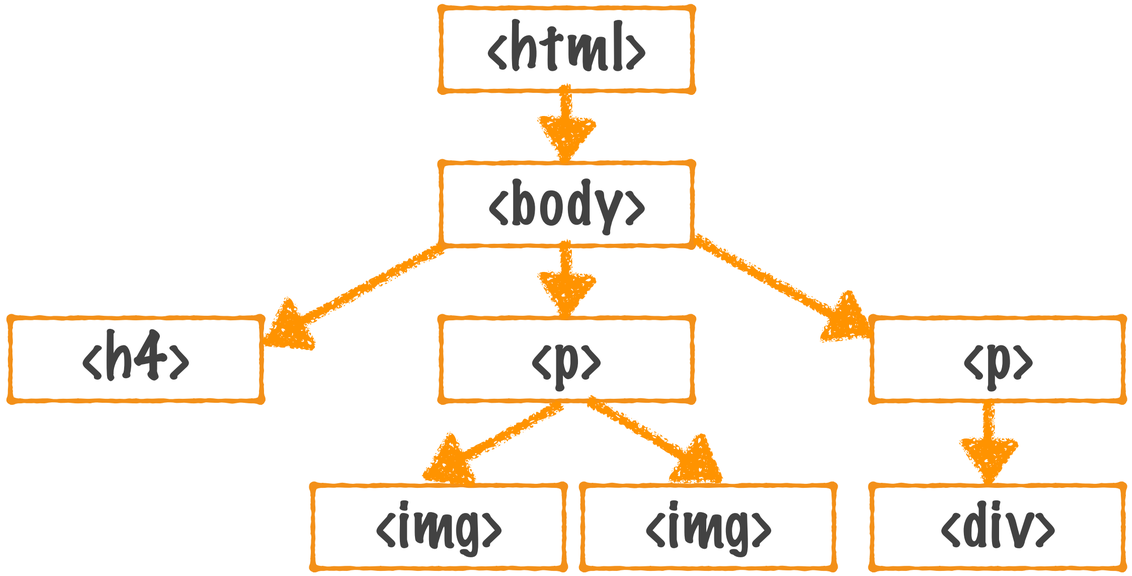
Capture and bubble phase



jQuery makes things easy for us by abstracting us from all these details

It simply registers event handlers on the bubble phase

Capture and bubble phase



It simply registers event handlers on the bubble phase

Handling events just work!

EXAMPLE 11

MultipleEventHandlers.html

MultipleEventHandlers.html

```
<body>
<button id="hello">Hello!</button>
<button id="clickMe">Click me</button>
<button id="clickMeAgain">Click me as well!</button>
</body>
```

MultipleEventHandlers.html

```
<body>
<button id="hello">Hello!</button>
<button id="clickMe">Click me</button>
<button id="clickMeAgain">Click me as well!</button>
</body>
```

2 buttons with different kinds of handlers set up on them

MultipleEventHandlers.html

```
<body>
<button id="hello">Hello!</button>
<button id="clickMe">Click me</button>
<button id="clickMeAgain">Click me as well!</button>
</body>
```

Each button has their own id

MultipleEventHandlers.html

```
$(document).ready(function () {
    $('#hello').click(function (e) {
        console.log("From listener 1");
    });

    $('#hello').click(function (e) {
        console.log("From listener 2");
    });

    $('#hello').click(function (e) {
        console.log("From listener 3");
    });

    $('#clickMe').on('mouseenter mouseleave', function(e) {
        console.log('Mouse event: ' + e.type);
    });

    $('#clickMeAgain').on({
        mouseenter: function() {
            console.log("Enter button");
        },
        mouseleave: function() {
            console.log("Leave button");
        },
        click: function() {
            console.log("Click button");
        }
    });
});
```

Here are the button handlers

MultipleEventHandlers.html

```
$(document).ready(function () {
    $('#hello').click(function (e) {
        console.log("From listener 1");
    });

    $('#hello').click(function (e) {
        console.log("From listener 2");
    });

    $('#hello').click(function (e) {
        console.log("From listener 3");
    });

    $('#clickMe').on('mouseenter mouseleave', function(e) {
        console.log('Mouse event: ' + e.type);
    });

    $('#clickMeAgain').on({
        mouseenter: function() {
            console.log("Enter button");
        },
        mouseleave: function() {
            console.log("Leave button");
        },
        click: function() {
            console.log("Click button");
        }
    });
});
```

The "hello" button has 3 event handlers attached to it

MultipleEventHandlers.html

```
$(document).ready(function () {
    $('#hello').click(function (e) {
        console.log("From listener 1");
    });

    $('#hello').click(function (e) {
        console.log("From listener 2");
    });

    $('#hello').click(function (e) {
        console.log("From listener 3");
    });

});
```

Each of these event handlers are called one after another when the button is clicked

MultipleEventHandlers.html

```
$(document).ready(function () {
    $('#hello').click(function (e) {
        console.log("From listener 1");
    });

    $('#hello').click(function (e) {
        console.log("From listener 2");
    });

    $('#hello').click(function (e) {
        console.log("From listener 3");
    });

});
```

They are called in the order in which the event handlers are set up

MultipleEventHandlers.html

Example11Demo - the first button

MultipleEventHandlers.html

```
$(document).ready(function () {
    $('#hello').click(function (e) {
        console.log("From listener 1");
    });

    $('#hello').click(function (e) {
        console.log("From listener 2");
    });

    $('#hello').click(function (e) {
        console.log("From listener 3");
    });

    $('#clickMe').on('mouseenter mouseleave', function(e) {
        console.log('Mouse event: ' + e.type);
    });

    $('#clickMeAgain').on({
        mouseenter: function() {
            console.log("Enter button");
        },
        mouseleave: function() {
            console.log("Leave button");
        },
        click: function() {
            console.log("Click button");
        }
    });
});
```

MultipleEventHandlers.html

```
$('#clickMe').on('mouseenter mouseleave', function(e) {
    console.log('Mouse event: ' + e.type);
});
```

The click me button has handlers on both the mouseenter and the mouseleave events

MultipleEventHandlers.html

```
$('#clickMe').on('mouseenter mouseleave', function(e) {
    console.log('Mouse event: ' + e.type);
});
```

You can specify the same handler for both events in one go using the on() function

MultipleEventHandlers.html

Example11Demo - the second button

MultipleEventHandlers.html

```
$(document).ready(function () {
    $('#hello').click(function (e) {
        console.log("From listener 1");
    });

    $('#hello').click(function (e) {
        console.log("From listener 2");
    });

    $('#hello').click(function (e) {
        console.log("From listener 3");
    });

    $('#clickMe').on('mouseenter mouseleave', function(e) {
        console.log('Mouse event: ' + e.type);
    });

    $('#clickMeAgain').on({
        mouseenter: function() {
            console.log("Enter button");
        },
        mouseleave: function() {
            console.log("Leave button");
        },
        click: function() {
            console.log("Click button");
        }
    });
});
```

MultipleEventHandlers.html

```
    $('#clickMeAgain').on({
        mouseenter: function() {
            console.log("Enter button");
        },
        mouseleave: function() {
            console.log("Leave button");
        },
        click: function() {
            console.log("Click button");
        }
    });
});
```

This button has multiple event handlers, a different one for each event

MultipleEventHandlers.html

```
    $('#clickMeAgain').on({
        mouseenter: function() {
            console.log("Enter button");
        },
        mouseleave: function() {
            console.log("Leave button");
        },
        click: function() {
            console.log("Click button");
        }
    });
});
```

All specified in one object which is the input argument to the on() method

MultipleEventHandlers.html

Example11Demo - the third button

EXAMPLE 12

PassingDataToEventHandlers.html

Developers can also pass **additional** data to event handlers via the event object

The `event.data` property holds this information

PassingDataToEventHandlers.html

Example12Demo

```
$(document).ready(function () {
    var ghosts = [
        'The Bloody Baron',
        'The Fat Friar',
        'The Grey Lady',
        'Nearly Headless Nick'
    ];
    $('#ghost').on('click', {
        ghostNames: ghosts
    },
    function (e) {
        console.log(e.data.ghostNames[
            Math.floor(Math.random() * e.data.ghostNames.length)]);
    });
});
```

Initialize an array of the Hogwarts house ghosts

PassingDataToEventHandlers.html

```
$(document).ready(function () {
    var ghosts = [
        'The Bloody Baron',
        'The Fat Friar',
        'The Grey Lady',
        'Nearly Headless Nick'
    ];
    $('#ghost').on('click', {
        ghostNames: ghosts
    },
    function (e) {
        console.log(e.data.ghostNames[
            Math.floor(Math.random() * e.data.ghostNames.length)]);
    });
});
```

Add a click event handler on the button

PassingDataToEventHandlers.html

```
$(document).ready(function () {
    var ghosts = [
        'The Bloody Baron',
        'The Fat Friar',
        'The Grey Lady',
        'Nearly Headless Nick'
    ];
    $('#ghost').on('click', {
        ghostNames: ghosts
    },
    function (e) {
        console.log(e.data.ghostNames[
            Math.floor(Math.random() * e.data.ghostNames.length)]);
    });
});
```

Specify an object as the second argument for the on() method

PassingDataToEventHandlers.html

```
$(document).ready(function () {
    var ghosts = [
        'The Bloody Baron',
        'The Fat Friar',
        'The Grey Lady',
        'Nearly Headless Nick'
    ];
    $('#ghost').on('click', {
        ghostNames: ghosts
    },
    function (e) {
        console.log(e.data.ghostNames[
            Math.floor(Math.random() * e.data.ghostNames.length)]);
    });
});
```

The array is assigned as a property on this object

PassingDataToEventHandlers.html

```
$(document).ready(function () {
    var ghosts = [
        'The Bloody Baron',
        'The Fat Friar',
        'The Grey Lady',
        'Nearly Headless Nick'
    ];
    $('#ghost').on('click', {
        ghostNames: ghosts
    },
    function (e) {
        console.log(e.data.ghostNames[
            Math.floor(Math.random() * e.data.ghostNames.length)]);
    });
});
```

The third argument is the event handler function

PassingDataToEventHandlers.html

```
$(document).ready(function () {
    var ghosts = [
        'The Bloody Baron',
        'The Fat Friar',
        'The Grey Lady',
        'Nearly Headless Nick'
    ];
    $('#ghost').on('click', {
        ghostNames: ghosts
    },
    function (e) {
        console.log(e.data.ghostNames[
            Math.floor(Math.random() * e.data.ghostNames.length)]);
    });
});
```

The ghostNames property can be accessed on `e.data` of the event object

PassingDataToEventHandlers.html

```
$(document).ready(function () {
    var ghosts = [
        'The Bloody Baron',
        'The Fat Friar',
        'The Grey Lady',
        'Nearly Headless Nick'
    ];
    $('#ghost').on('click', {
        ghostNames: ghosts
    },
    function (e) {
        console.log(e.data.ghostNames[
            Math.floor(Math.random() * e.data.ghostNames.length)]);
    });
});
```

Any properties we set on the object while setting up the event handler

PassingDataToEventHandlers.html

```
$(document).ready(function () {
    var ghosts = [
        'The Bloody Baron',
        'The Fat Friar',
        'The Grey Lady',
        'Nearly Headless Nick'
    ];
    $('#ghost').on('click', {
        ghostNames: ghosts
    },
    function (e) {
        console.log(e.data.ghostNames[
            Math.floor(Math.random() * e.data.ghostNames.length)]);
    });
});
```

Is accessible within the event!

EXAMPLE 13

ListenJustOnce.html

ListenJustOnce.html

It's possible to wire up the event handler to listen to exactly one occurrence of the event

ListenJustOnce.html

It's possible to wire up the event handler to listen to exactly one occurrence of the event

The event handler will execute once, and then jQuery will remove it so that it does not execute again

ListenJustOnce.html

```
$(document).ready(function () {
    var ghosts = [
        'The Bloody Baron',
        'The Fat Friar',
        'The Grey Lady',
        'Nearly Headless Nick'
    ];
    $('#ghost').one('click', {
        ghostNames: ghosts
    },
    function (e) {
        alert(e.data.ghostNames[
            Math.floor(Math.random() * e.data.ghostNames.length)]);
    });
});
```

This looks exactly like the code we saw in the last example

ListenJustOnce.html

```
$(document).ready(function () {
    var ghosts = [
        'The Bloody Baron',
        'The Fat Friar',
        'The Grey Lady',
        'Nearly Headless Nick'
    ];
    $('#ghost').one('click', {
        ghostNames: ghosts
    },
    function (e) {
        alert(e.data.ghostNames[
            Math.floor(Math.random() * e.data.ghostNames.length)]);
    });
});
```

Except we use the one() function to hook up the event handler to listen to just one occurrence of the event

ListenJustOnce.html

```
$(document).ready(function () {
    var ghosts = [
        'The Bloody Baron',
        'The Fat Friar',
        'The Grey Lady',
        'Nearly Headless Nick'
    ];
    $('#ghost').one('click', {
        ghostNames: ghosts
    },
    function (e) {
        alert(e.data.ghostNames[
            Math.floor(Math.random() * e.data.ghostNames.length)]);
    });
});
```

We also show the ghost name in an alert box rather than on the console log

ListenJustOnce.html

```
$(document).ready(function () {
    var ghosts = [
        'The Bloody Baron',
        'The Fat Friar',
        'The Grey Lady',
        'Nearly Headless Nick'
    ];
    $('#ghost').one('click', {
        ghostNames: ghosts
    },
    function (e) {
        alert(e.data.ghostNames[
            Math.floor(Math.random() * e.data.ghostNames.length)]);
    });
});
```

This makes it pretty clear that the event handler executes only the first time the button is clicked

ListenJustOnce.html

```
$(document).ready(function () {
    var ghosts = [
        'The Bloody Baron',
        'The Fat Friar',
        'The Grey Lady',
        'Nearly Headless Nick'
    ];
    $('#ghost').one('click', {
        ghostNames: ghosts
    },
    function (e) {
        alert(e.data.ghostNames[
            Math.floor(Math.random() * e.data.ghostNames.length)]);
    });
});
```

The remaining stuff remains exactly the same as the on() function - data binding to events etc.

ListenJustOnce.html

Example13Demo

EXAMPLE 14

RemovingEventHandlers.html

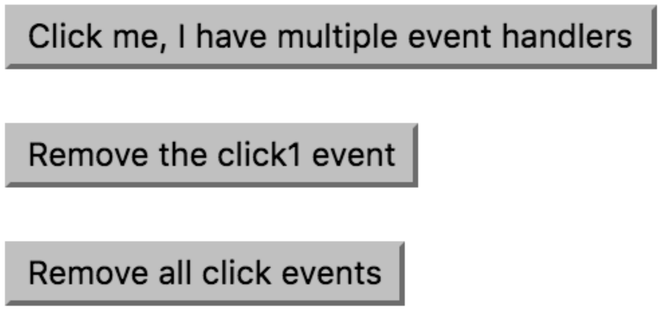
Event handlers can be unhooked or removed from elements when they are no longer needed

RemovingEventHandlers.html

Example14Demo

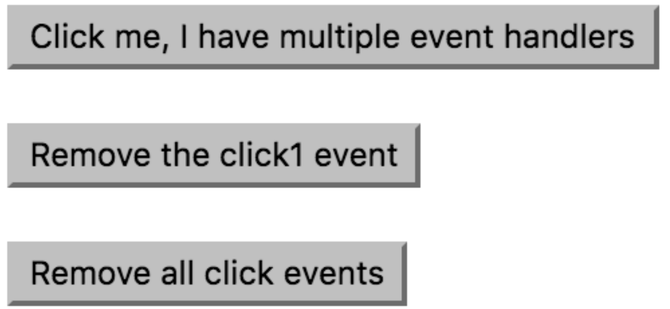
RemovingEventHandlers.html

```
<body>
<button id="clickMe">Click me, I have multiple event handlers</button>
<br>
<br>
<button id="removeClick">Remove the click1 event</button>
<br>
<br>
<button id="removeAll">Remove all click events</button>
</body>
```



RemovingEventHandlers.html

```
$(document).ready(function () {
    var click1Fn = function() {
        console.log('CLICK event handler 1');
    };
    var click2Fn = function() {
        console.log('CLICK event handler 2');
    };
    var click3Fn = function() {
        console.log('CLICK event handler 3');
    };

    $('#clickMe').on('click', click1Fn);
    $('#clickMe').on('click', click2Fn);
    $('#clickMe').on('click', click3Fn);

    $('#removeClick').on('click', function() {
        console.log('Click 1 event removed');
        $('#clickMe').off('click', click1Fn);
    });

    $('#removeAll').on('click', function() {
        console.log('All click events removed');
        $('#clickMe').off('click');
    });
});
```



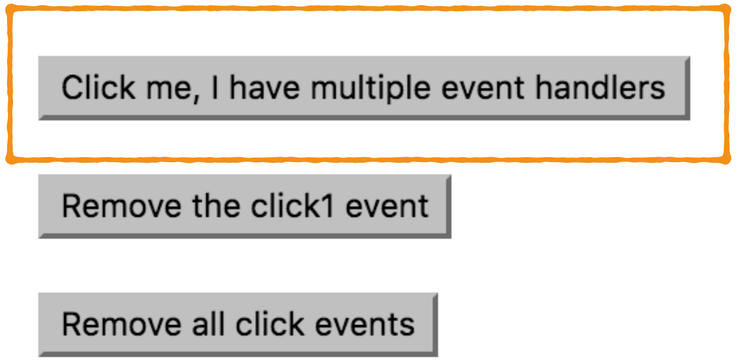
RemovingEventHandlers.html

```
$(document).ready(function () {
    var click1Fn = function() {
        console.log('CLICK event handler 1');
    };
    var click2Fn = function() {
        console.log('CLICK event handler 2');
    };
    var click3Fn = function() {
        console.log('CLICK event handler 3');
    };

    $('#clickMe').on('click', click1Fn);
    $('#clickMe').on('click', click2Fn);
    $('#clickMe').on('click', click3Fn);

    $('#removeClick').on('click', function() {
        console.log('Click 1 event removed');
        $('#clickMe').off('click', click1Fn);
    });

    $('#removeAll').on('click', function() {
        console.log('All click events removed');
        $('#clickMe').off('click');
    });
});
```

Define 3 click functions and assign each to a variable

RemovingEventHandlers.html

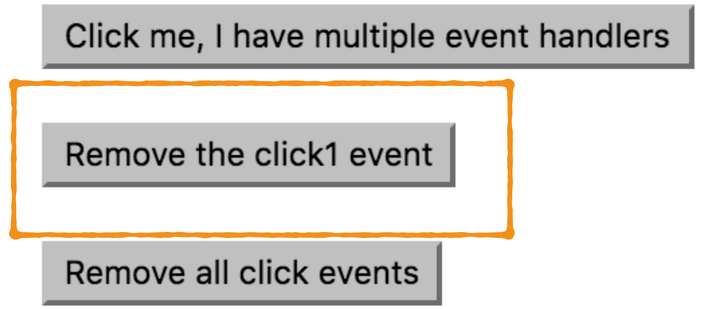
```
$(document).ready(function () {
    var click1Fn = function() {
        console.log('CLICK event handler 1');
    };
    var click2Fn = function() {
        console.log('CLICK event handler 2');
    };
    var click3Fn = function() {
        console.log('CLICK event handler 3');
    };

    $('#clickMe').on('click', click1Fn);
    $('#clickMe').on('click', click2Fn);
    $('#clickMe').on('click', click3Fn);

    $('#removeClick').on('click', function() {
        console.log('Click 1 event removed');
        $('#clickMe').off('click', click1Fn);
    });

    $('#removeAll').on('click', function() {
        console.log('All click events removed');
        $('#clickMe').off('click');
    });
});
```



Hook up the event handlers to the same button

RemovingEventHandlers.html

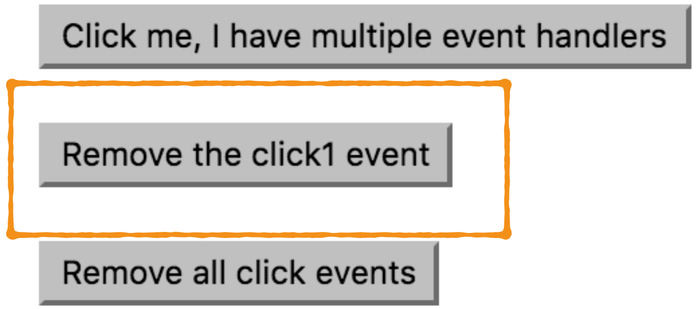
```
$(document).ready(function () {
    var click1Fn = function() {
        console.log('CLICK event handler 1');
    };
    var click2Fn = function() {
        console.log('CLICK event handler 2');
    };
    var click3Fn = function() {
        console.log('CLICK event handler 3');
    };

    $('#clickMe').on('click', click1Fn);
    $('#clickMe').on('click', click2Fn);
    $('#clickMe').on('click', click3Fn);

    $('#removeClick').on('click', function() {
        console.log('Click 1 event removed');
        $('#clickMe').off('click', click1Fn);
    });

    $('#removeAll').on('click', function() {
        console.log('All click events removed');
        $('#clickMe').off('click');
    });
});
```



Remove one of the event handlers when this button is clicked

RemovingEventHandlers.html

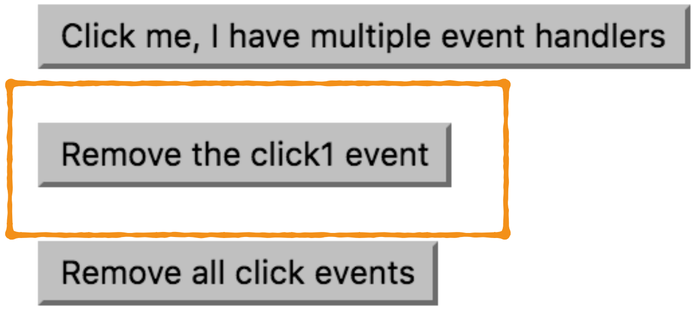
```
$(document).ready(function () {
    var click1Fn = function() {
        console.log('CLICK event handler 1');
    };
    var click2Fn = function() {
        console.log('CLICK event handler 2');
    };
    var click3Fn = function() {
        console.log('CLICK event handler 3');
    };

    $('#clickMe').on('click', click1Fn);
    $('#clickMe').on('click', click2Fn);
    $('#clickMe').on('click', click3Fn);


    $('#removeClick').on('click', function() {
        console.log('Click 1 event removed');
        $('#clickMe').off('click', click1Fn);
    });

    $('#removeAll').on('click', function() {
        console.log('All click events removed');
        $('#clickMe').off();
    });
});
```



The off() method on the button turns off the event handlers

RemovingEventHandlers.html

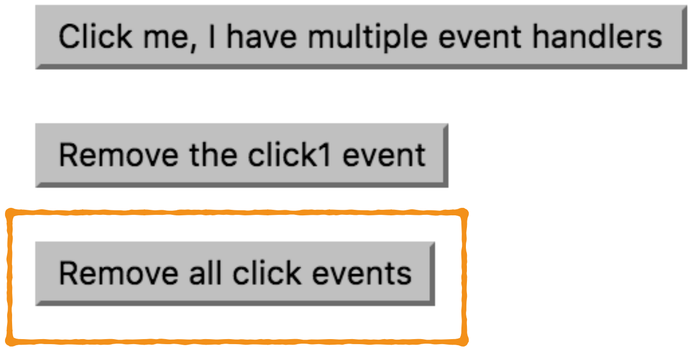
```
$(document).ready(function () {
    var click1Fn = function() {
        console.log('CLICK event handler 1');
    };
    var click2Fn = function() {
        console.log('CLICK event handler 2');
    };
    var click3Fn = function() {
        console.log('CLICK event handler 3');
    };

    $('#clickMe').on('click', click1Fn);
    $('#clickMe').on('click', click2Fn);
    $('#clickMe').on('click', click3Fn);


    $('#removeClick').on('click', function() {
        console.log('Click 1 event removed');
        $('#clickMe').off('click', click1Fn);
    });

    $('#removeAll').on('click', function() {
        console.log('All click events removed');
        $('#clickMe').off('click');
    });
});
```



Pass in the handler that you want removed

RemovingEventHandlers.html

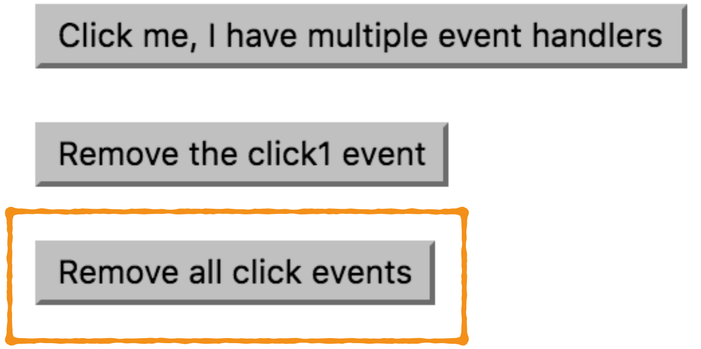
```
$(document).ready(function () {
    var click1Fn = function() {
        console.log('CLICK event handler 1');
    };
    var click2Fn = function() {
        console.log('CLICK event handler 2');
    };
    var click3Fn = function() {
        console.log('CLICK event handler 3');
    };

    $('#clickMe').on('click', click1Fn);
    $('#clickMe').on('click', click2Fn);
    $('#clickMe').on('click', click3Fn);

    $('#removeClick').on('click', function() {
        console.log('Click 1 event removed');
        $('#clickMe').off('click', click1Fn);
    });

    $('#removeAll').on('click', function() {
        console.log('All click events removed');
        $('#clickMe').off('click');
    });
});
```



You can remove all click handlers in one go as well

RemovingEventHandlers.html

```
$(document).ready(function () {
    var click1Fn = function() {
        console.log('CLICK event handler 1');
    };
    var click2Fn = function() {
        console.log('CLICK event handler 2');
    };
    var click3Fn = function() {
        console.log('CLICK event handler 3');
    };

    $('#clickMe').on('click', click1Fn);
    $('#clickMe').on('click', click2Fn);
    $('#clickMe').on('click', click3Fn);

    $('#removeClick').on('click', function() {
        console.log('Click 1 event removed');
        $('#clickMe').off('click', click1Fn);
    });

    $('#removeAll').on('click', function() {
        console.log('All click events removed');
        $('#clickMe').off('click');
    });
});
```



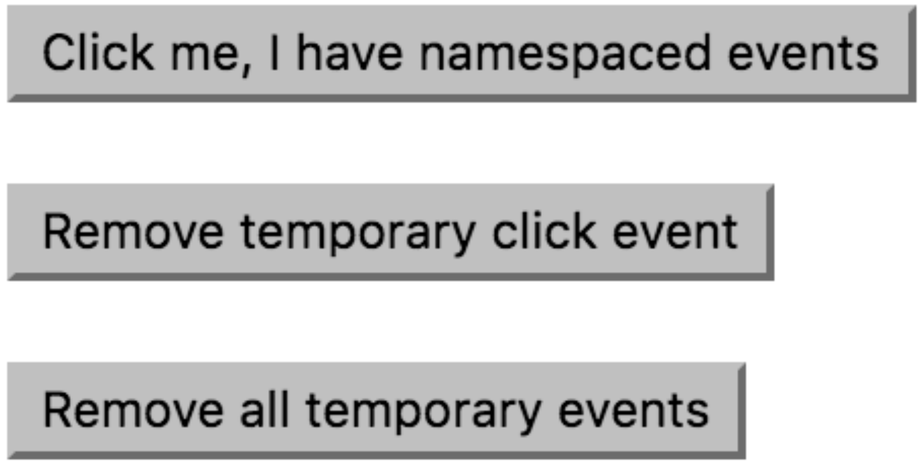
Just specify the off() function with no additional arguments!

EXAMPLE 15

NamespacingEvents.html

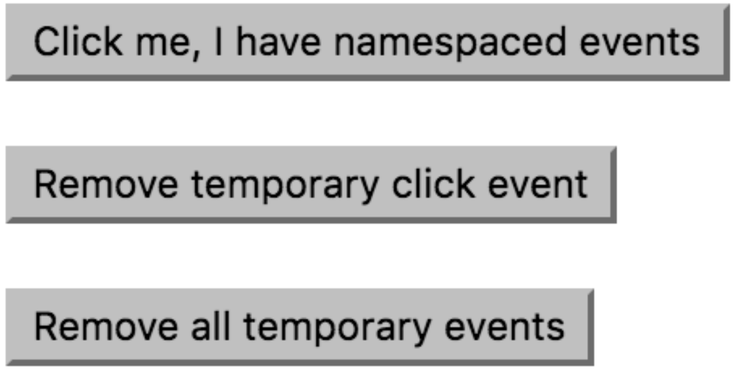
NamespacingEvents.html

```
<body>
<button id="clickMe">Click me, I have namespaced events</button>
<br>
<br>
<button id="removeClick">Remove temporary click event</button>
<br>
<br>
<button id="removeAll">Remove all temporary events</button>
</body>
```



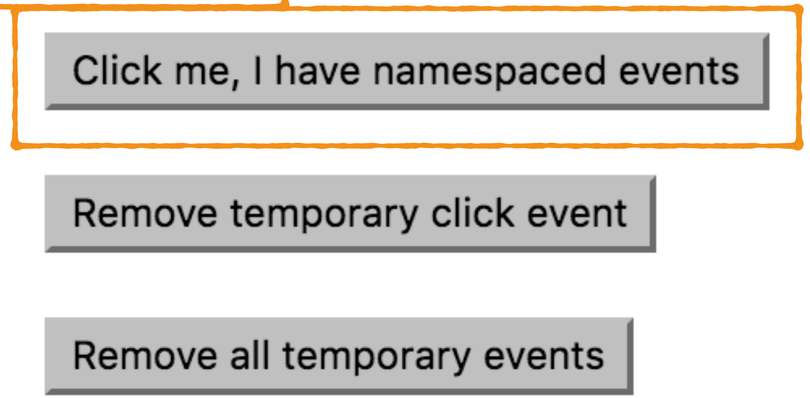
NamespacingEvents.html

```
$(document).ready(function () {
    $('#clickMe').on('click.importantNamespace', function() {
        console.log('I am a SUPER IMPORTANT event handler');
    });

    $('#clickMe').on('click.temporaryNamespace', function() {
        console.log('I am a temporary click event handler, I can be removed anytime');
    });
    $('#clickMe').on('mouseenter.temporaryNamespace', function() {
        console.log('I am a temporary mouseenter event handler');
    });
    $('#clickMe').on('mouseleave.temporaryNamespace', function() {
        console.log('I am a temporary mouseleave event handler');
    });

    $('#removeClick').on('click', function() {
        console.log('Temporary click events turned off');
        $('#clickMe').off('click.temporaryNamespace');
    });

    $('#removeAll').on('click', function() {
        console.log('All temporary events turned off');
        $('#clickMe').off('.temporaryNamespace');
    });
});
```



NamespacingEvents.html

```
$(document).ready(function () {
    $('#clickMe').on('click.importantNamespace', function() {
        console.log('I am a SUPER IMPORTANT event handler');
    });

    $('#clickMe').on('click.temporaryNamespace', function() {
        console.log('I am a temporary click event handler, I can be removed anytime');
    });
    $('#clickMe').on('mouseenter.temporaryNamespace', function() {
        console.log('I am a temporary mouseenter event handler');
    });
    $('#clickMe').on('mouseleave.temporaryNamespace', function() {
        console.log('I am a temporary mouseleave event handler');
    });

    $('#removeClick').on('click', function() {
        console.log('Temporary click events turned off');
        $('#clickMe').off('click.temporaryNamespace');
    });

    $('#removeAll').on('click', function() {
        console.log('All temporary events turned off');
        $('#clickMe').off('.temporaryNamespace');
    });
});
```



There are 4 events handlers on this button

NamespacingEvents.html

```
$(document).ready(function () {
    $('#clickMe').on('click.importantNamespace', function() {
        console.log('I am a SUPER IMPORTANT event handler');
    });

    $('#clickMe').on('click.temporaryNamespace', function() {
        console.log('I am a temporary click event handler, I can be removed anytime');
    });
    $('#clickMe').on('mouseenter.temporaryNamespace', function() {
        console.log('I am a temporary mouseenter event handler');
    });
    $('#clickMe').on('mouseleave.temporaryNamespace', function() {
        console.log('I am a temporary mouseleave event handler');
    });

    $('#removeClick').on('click', function() {
        console.log('Temporary click events turned off');
        $('#clickMe').off('click.temporaryNamespace');
    });

    $('#removeAll').on('click', function() {
        console.log('Temporary events turned off');
        $('#clickMe').off('.temporaryNamespace');
    });
});
```

The first one is an important handler so it has its own namespace

NamespacingEvents.html

```
$(document).ready(function () {
    $('#clickMe').on('click.importantNamespace', function() {
        console.log('I am a SUPER IMPORTANT event handler');
    });

    $('#clickMe').on('click.temporaryNamespace', function() {
        console.log('I am a temporary click event handler, I can be removed anytime');
    });
    $('#clickMe').on('mouseenter.temporaryNamespace', function() {
        console.log('I am a temporary mouseenter event handler');
    });
    $('#clickMe').on('mouseleave.temporaryNamespace', function() {
        console.log('I am a temporary mouseleave event handler');
    });

    $('#removeClick').on('click', function() {
        console.log('Temporary click events turned off');
        $('#clickMe').off('click.temporaryNamespace');
    });

    $('#removeAll').on('click', function() {
        console.log('All temporary events turned off');
        $('#clickMe').off('.temporaryNamespace');
    });
});
```

The click event is in the "importantNamespace"

NamespacingEvents.html

```
$(document).ready(function () {
    $('#clickMe').on('click.importantNamespace', function() {
        console.log('I am a SUPER IMPORTANT event handler');
    });

    $('#clickMe').on('click.temporaryNamespace', function() {
        console.log('I am a temporary click event handler, I can be removed anytime');
    });
    $('#clickMe').on('mouseenter.temporaryNamespace', function() {
        console.log('I am a temporary mouseenter event handler');
    });
    $('#clickMe').on('mouseleave.temporaryNamespace', function() {
        console.log('I am a temporary mouseleave event handler');
    });

    $('#removeClick').on('click', function() {
        console.log('Temporary click events turned off');
        $('#clickMe').off('click.temporaryNamespace');
    });

    $('#removeAll').on('click', function() {
        console.log('Temporary events turned off');
        $('#clickMe').off('.temporaryNamespace');
    });
});
```

The other 3 handlers are temporary and have their own namespace

NamespacingEvents.html

```
$(document).ready(function () {
    $('#clickMe').on('click.importantNamespace', function() {
        console.log('I am a SUPER IMPORTANT event handler');
    });

    $('#clickMe').on('click.temporaryNamespace', function() {
        console.log('I am a temporary click event handler, I can be removed anytime');
    });
    $('#clickMe').on('mouseenter.temporaryNamespace', function() {
        console.log('I am a temporary mouseenter event handler');
    });
    $('#clickMe').on('mouseleave.temporaryNamespace', function() {
        console.log('I am a temporary mouseleave event handler');
    });

    $('#removeClick').on('click', function() {
        console.log('Temporary click events turned off');
        $('#clickMe').off('click.temporaryNamespace');
    });

    $('#removeAll').on('click', function() {
        console.log('All temporary events turned off');
        $('#clickMe').off('.temporaryNamespace');
    });
});
```

The "temporaryNamespace"

NamespacingEvents.html

```
$(document).ready(function () {
    $('#clickMe').on('click.importantNamespace', function() {
        console.log('I am a SUPER IMPORTANT event handler');
    });

    $('#clickMe').on('click.temporaryNamespace', function() {
        console.log('I am a temporary click event handler, I can be removed anytime');
    });
    $('#clickMe').on('mouseenter.temporaryNamespace', function() {
        console.log('I am a temporary mouseenter event handler');
    });
    $('#clickMe').on('mouseleave.temporaryNamespace', function() {
        console.log('I am a temporary mouseleave event handler');
    });

    $('#removeClick').on('click', function() {
        console.log('Temporary click events turned off');
        $('#clickMe').off('click.temporaryNamespace');
    });

    $('#removeAll').on('click', function() {
        console.log('All temporary events removed');
        $('#clickMe').off('.temporaryNamespace');
    });
});
```

The namespace serves as a logical grouping of event handlers

NamespacingEvents.html

```
$(document).ready(function () {
    $('#clickMe').on('click.importantNamespace', function() {
        console.log('I am a SUPER IMPORTANT event handler');
    });

    $('#clickMe').on('click.temporaryNamespace', function() {
        console.log('I am a temporary click event handler, I can be removed anytime');
    });
    $('#clickMe').on('mouseenter.temporaryNamespace', function() {
        console.log('I am a temporary mouseenter event handler');
    });
    $('#clickMe').on('mouseleave.temporaryNamespace', function() {
        console.log('I am a temporary mouseleave event handler');
    });

    $('#removeClick').on('click', function() {
        console.log('Temporary click events turned off');
        $('#clickMe').off('click.temporaryNamespace');
    });

    $('#removeAll').on('click', function() {
        console.log('All temporary events turned off');
        $('#clickMe').off('.temporaryNamespace');
    });
});
```

You can identity a handler by the name e.g. "click.temporaryNamespace"

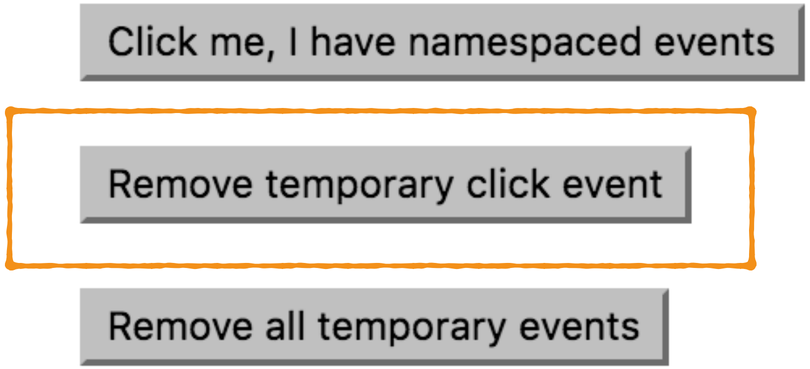
NamespacingEvents.html

```
$(document).ready(function () {
    $('#clickMe').on('click.importantNamespace', function() {
        console.log('I am a SUPER IMPORTANT event handler');
    });

    $('#clickMe').on('click.temporaryNamespace', function() {
        console.log('I am a temporary click event handler, I can be removed anytime');
    });
    $('#clickMe').on('mouseenter.temporaryNamespace', function() {
        console.log('I am a temporary mouseenter event handler');
    });
    $('#clickMe').on('mouseleave.temporaryNamespace', function() {
        console.log('I am a temporary mouseleave event handler');
    });

    $('#removeClick').on('click', function() {
        console.log('Temporary click events turned off');
        $('#clickMe').off('click.temporaryNamespace');
    });

    $('#removeAll').on('click', function() {
        console.log('All temporary events turned off');
        $('#clickMe').off('.temporaryNamespace');
    });
});
```

Or by a group e.g. ".temporaryNamespace"

NamespacingEvents.html

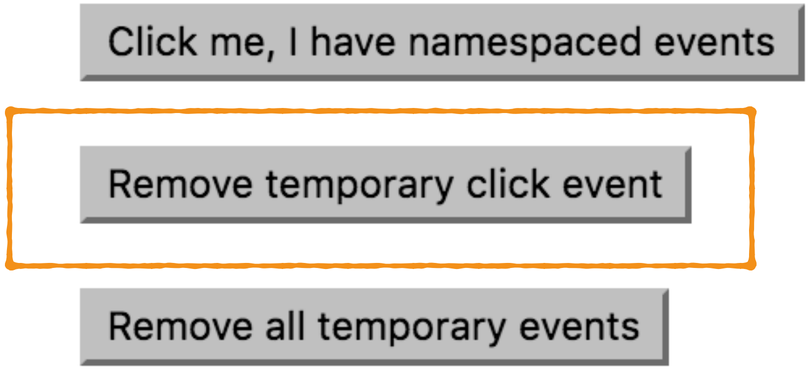
```
    $('#removeClick').on('click', function() {
        console.log('Temporary click events turned off');
        $('#clickMe').off('click.temporaryNamespace');
    });

    $('#removeAll').on('click', function() {
        console.log('All temporary events turned off');
        $('#clickMe').off('.temporaryNamespace');
    });
});
```



This removes only the temporary click event on the button

NamespacingEvents.html

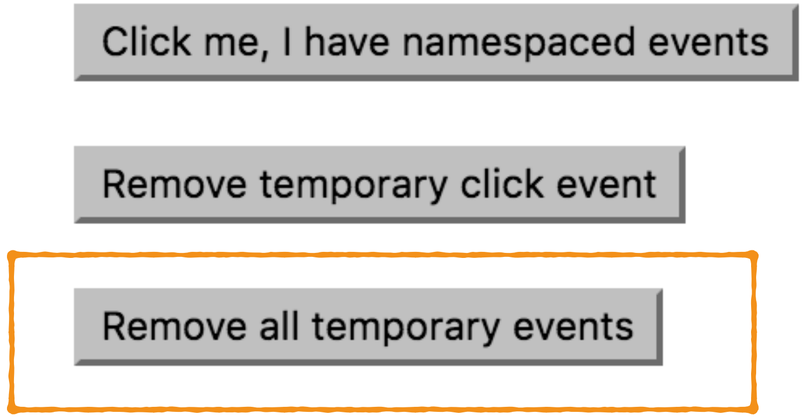
```
    $('#removeClick').on('click', function() {
        console.log('Temporary click events turned off');
        $('#clickMe').off('click.temporaryNamespace');
    });

    $('#removeAll').on('click', function() {
        console.log('All temporary events turned off');
        $('#clickMe').off('.temporaryNamespace');
    });
});
```



Just specify the right namespace for the event handler you want to unhook

NamespacingEvents.html

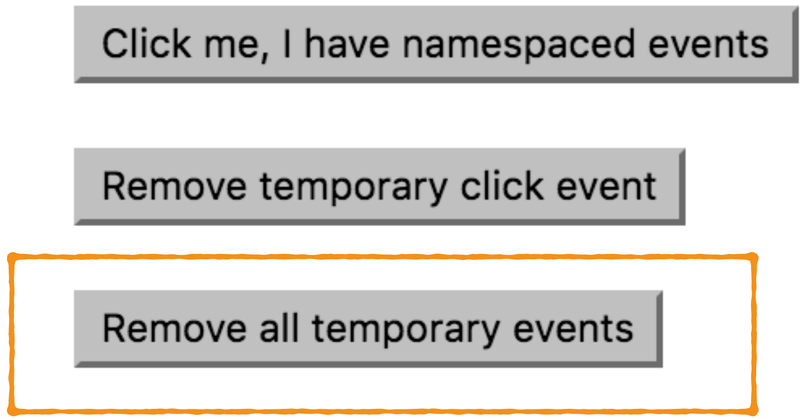
```
    $('#removeClick').on('click', function() {
        console.log('Temporary click events turned off');
        $('#clickMe').off('click.temporaryNamespace');
    });

    $('#removeAll').on('click', function() {
        console.log('All temporary events turned off');
        $('#clickMe').off('.temporaryNamespace');
    });
});
```



You can remove all temporary event handlers in one go as well

NamespacingEvents.html

```
    $('#removeClick').on('click', function() {
        console.log('Temporary click events turned off');
        $('#clickMe').off('click.temporaryNamespace');
    });

    $('#removeAll').on('click', function() {
        console.log('All temporary events turned off');
        $('#clickMe').off('.temporaryNamespace');
    });
});
```



Just make sure the namespace matches!

NamespacingEvents.html

Example15Demo

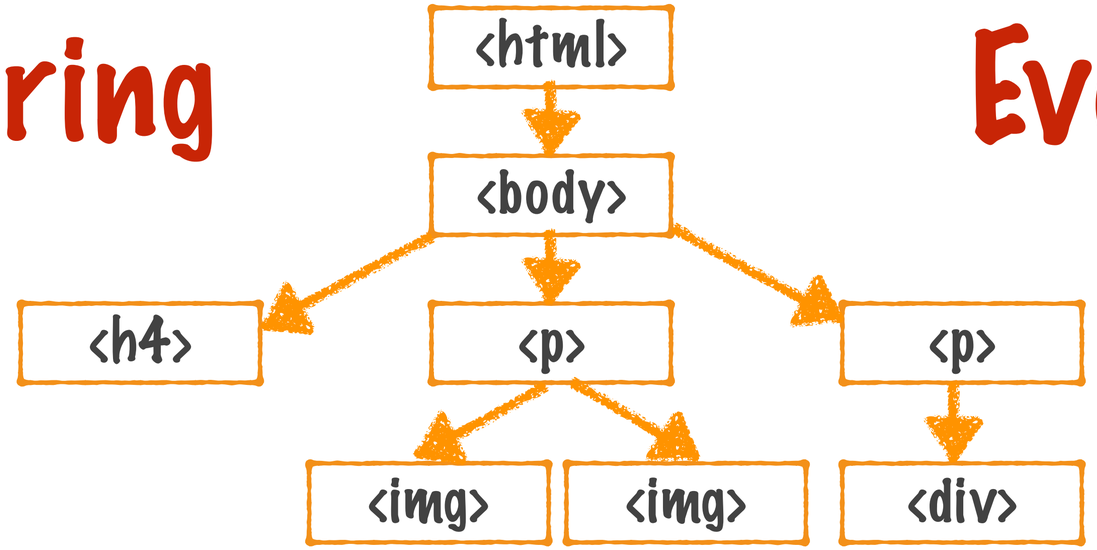
JQUERY

Event delegation

Recap

Capture and bubble phase

Event Capturing



Event Bubbling

So which of these methods do browsers use?

Both!

Recap

Capture and bubble phase

So which of these methods do browsers use?

Event Capturing

Event Bubbling

Different browsers originally decided on different models for event propagation

Recap

So which of these methods do browsers use?

Event Capturing

Event Bubbling

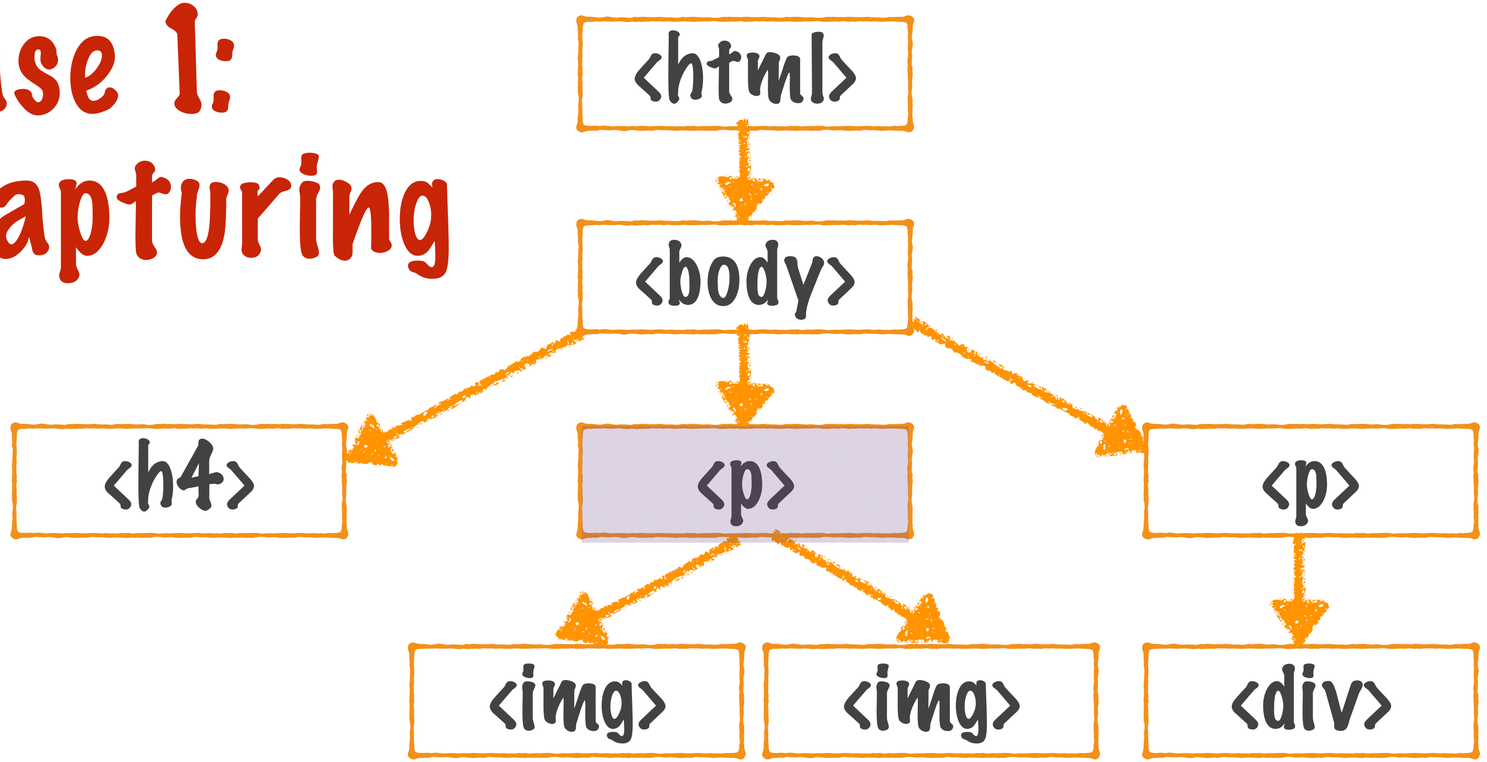
The DOM standard that was ultimately adopted was that both should be supported

How does that work?

Recap

Capture and bubble phase

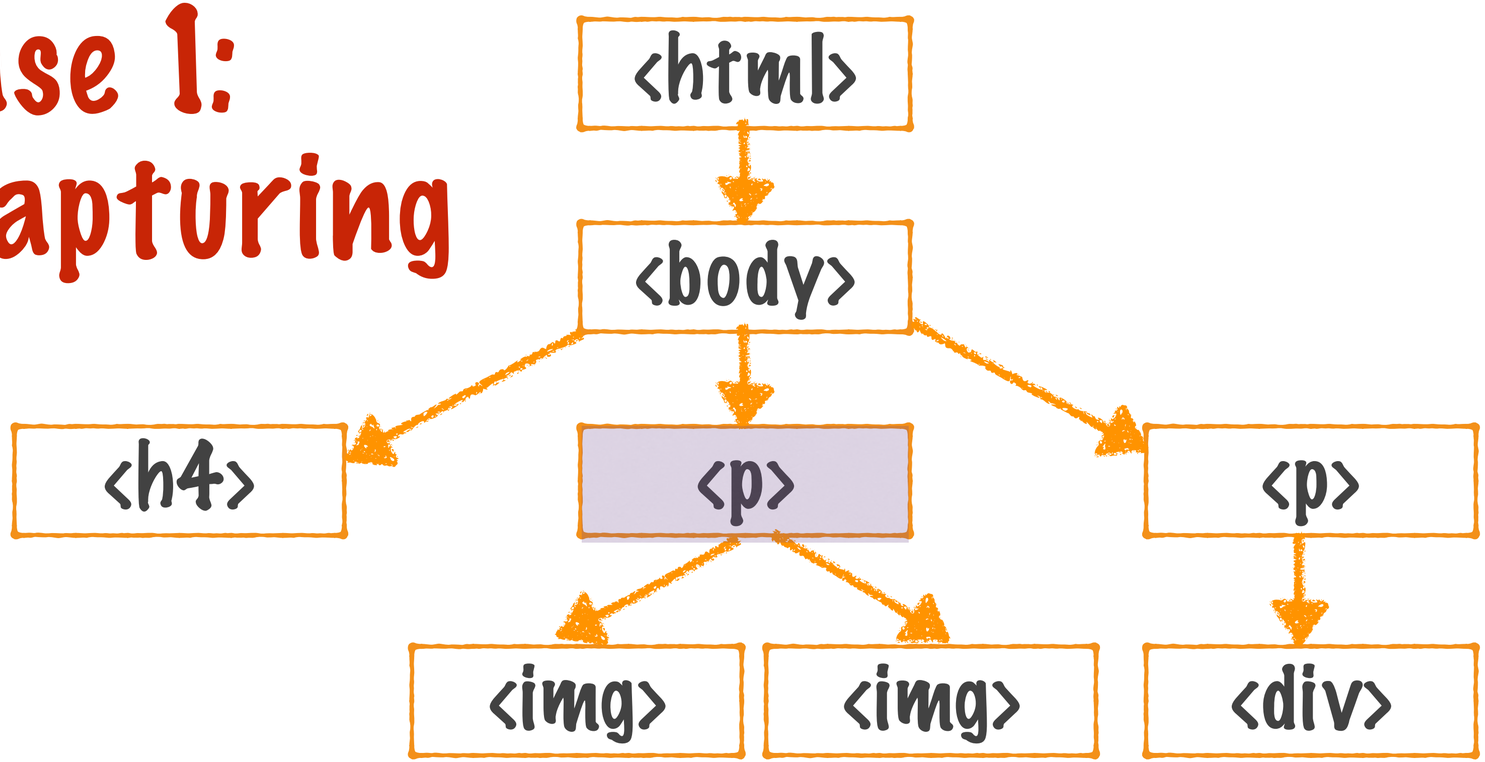
Phase 1: Event Capturing



Element <p> is clicked on

Recap

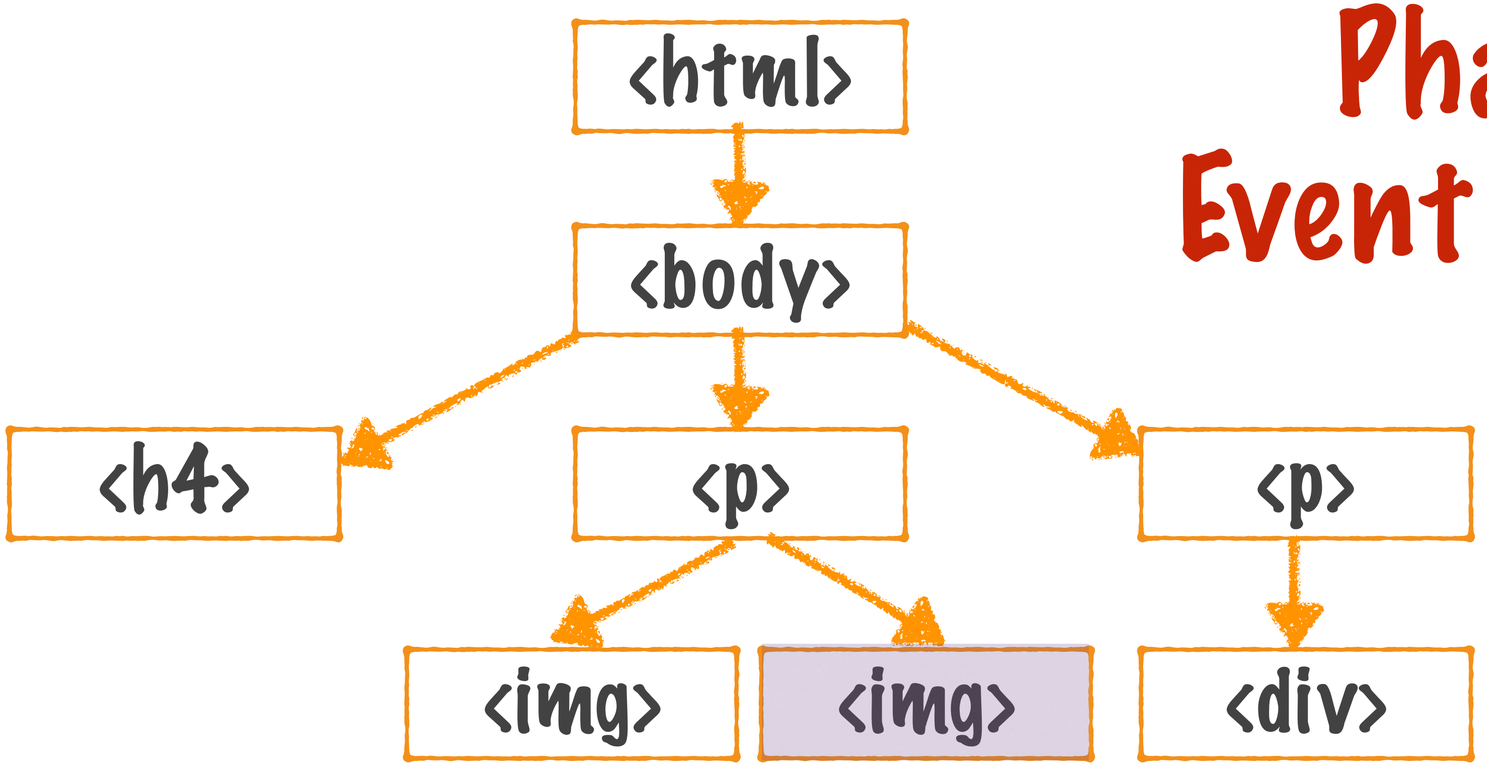
Phase 1: Event Capturing



It is propagated downwards to child elements

Recap

Capture and bubble phase



Phase 2: Event Bubbling

And then propagated upwards to parent elements

Event delegation

The event handlers we've set up so far only work when the element already exists in the DOM

You cannot hook up a listener to an element which does not exist!

Event delegation

You cannot hook up a listener to an element which does not exist!

These kind of events are called **direct events** - you directly listen to the event on an element

Event delegation

You cannot hook up a listener to
an element which does not exist!

This can be limiting though...

What if you want to listen to
clicks on **all buttons** within a page?

Event delegation

What if you want to listen to clicks on all buttons within a page?

Each time you dynamically add a button you'll also have to add an event handler

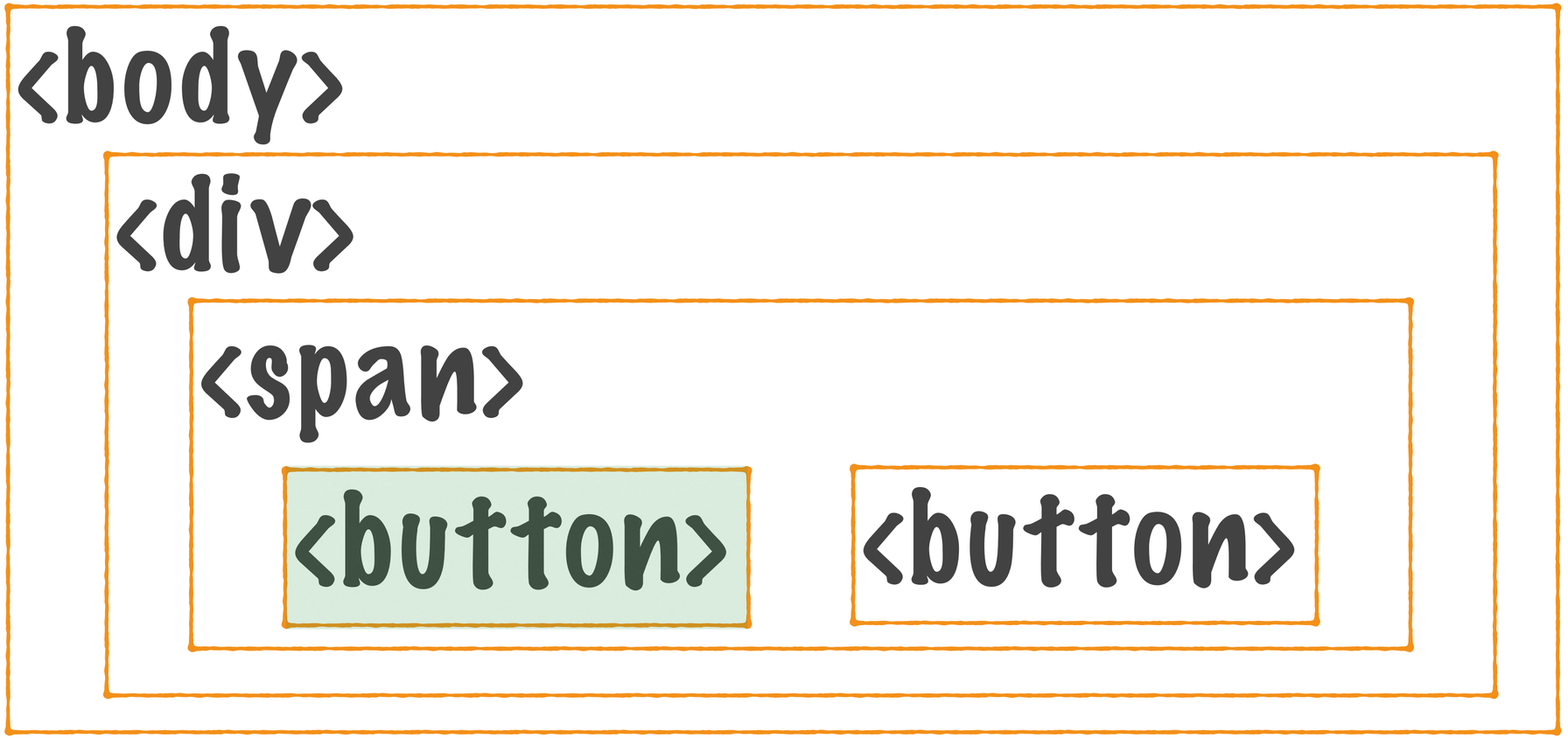
Event delegation

What if you want to listen to clicks on all buttons within a page?

Event delegation to the rescue!

Event delegation

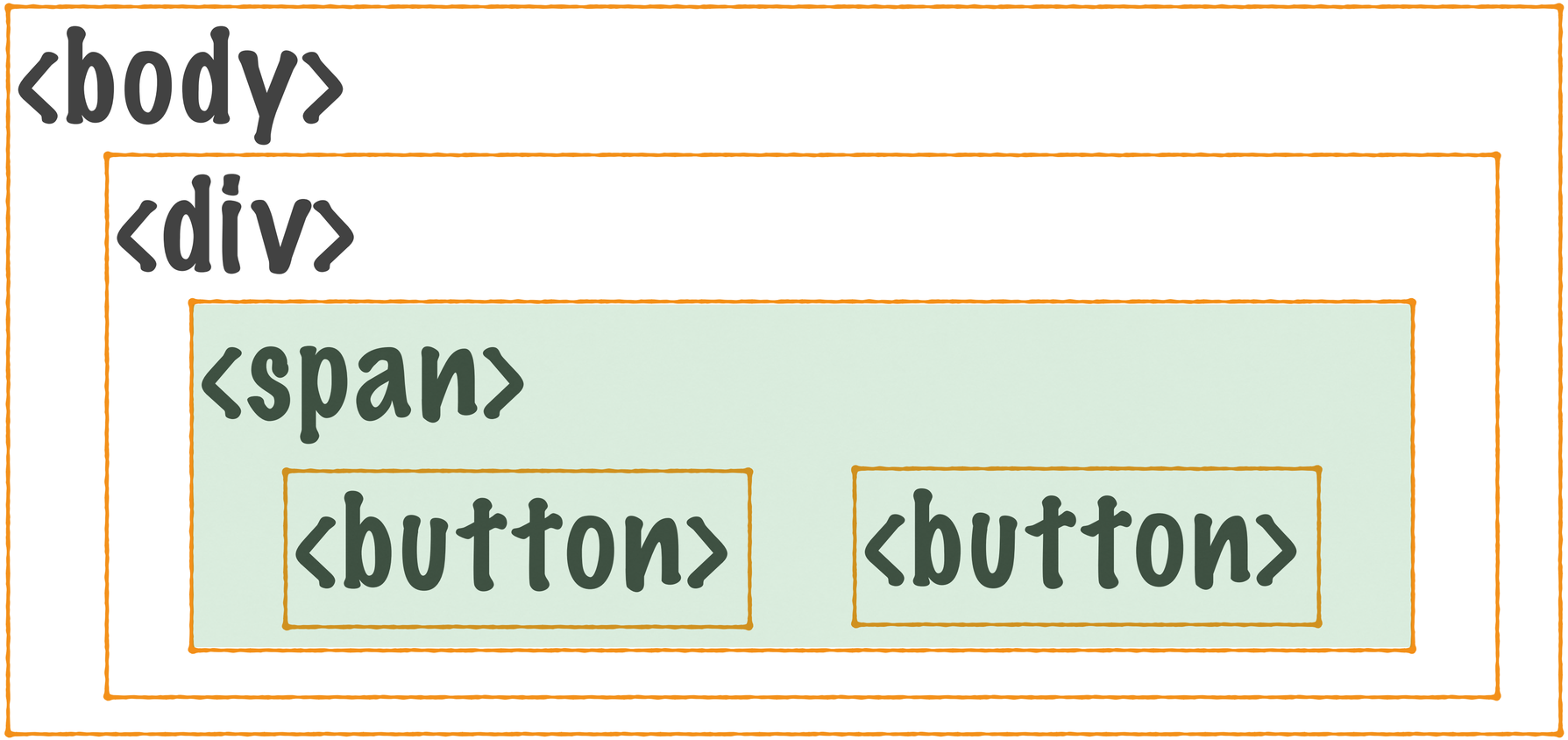
Event delegation to the rescue!



A button is clicked

Event delegation

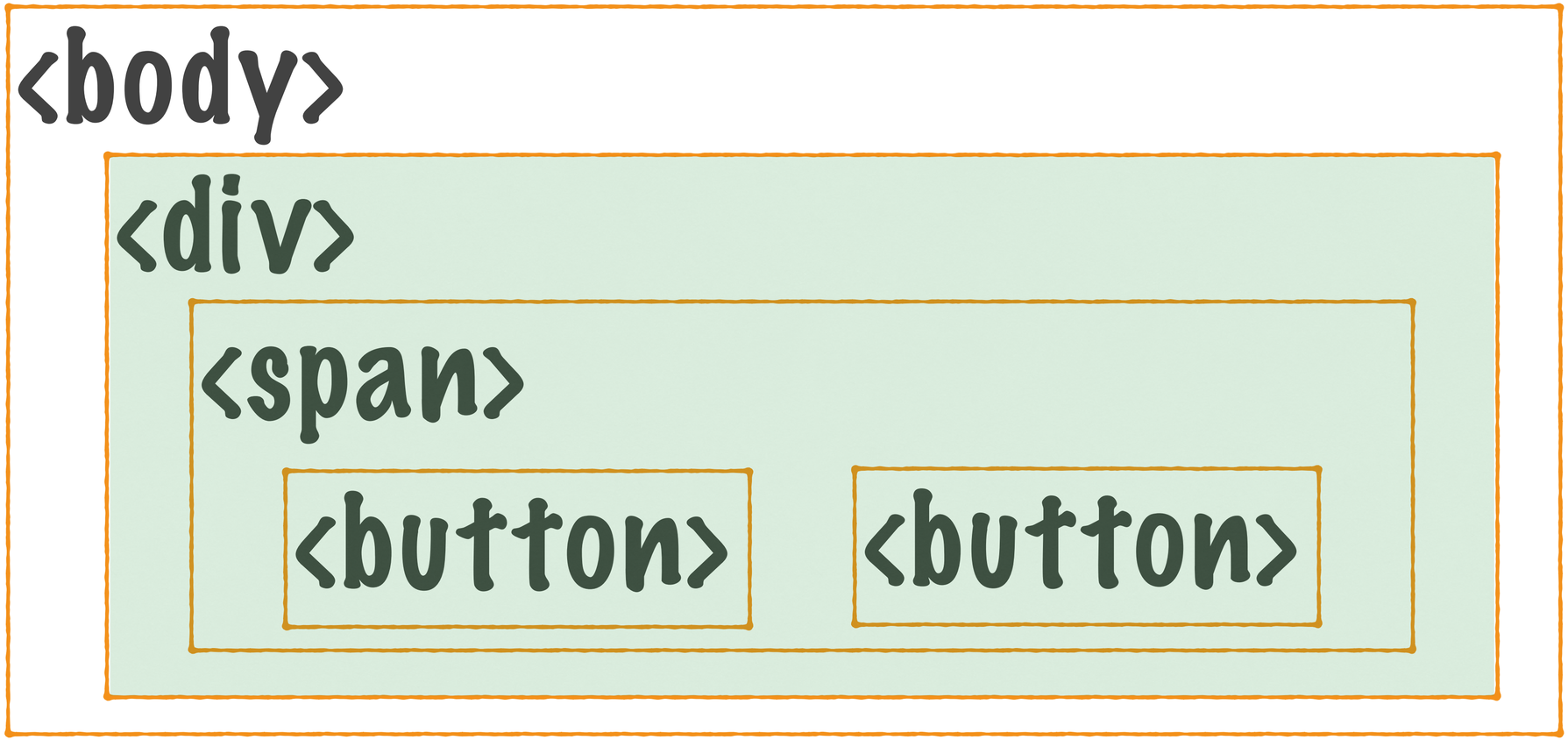
Event delegation to the rescue!



The event will be bubbled upwards

Event delegation

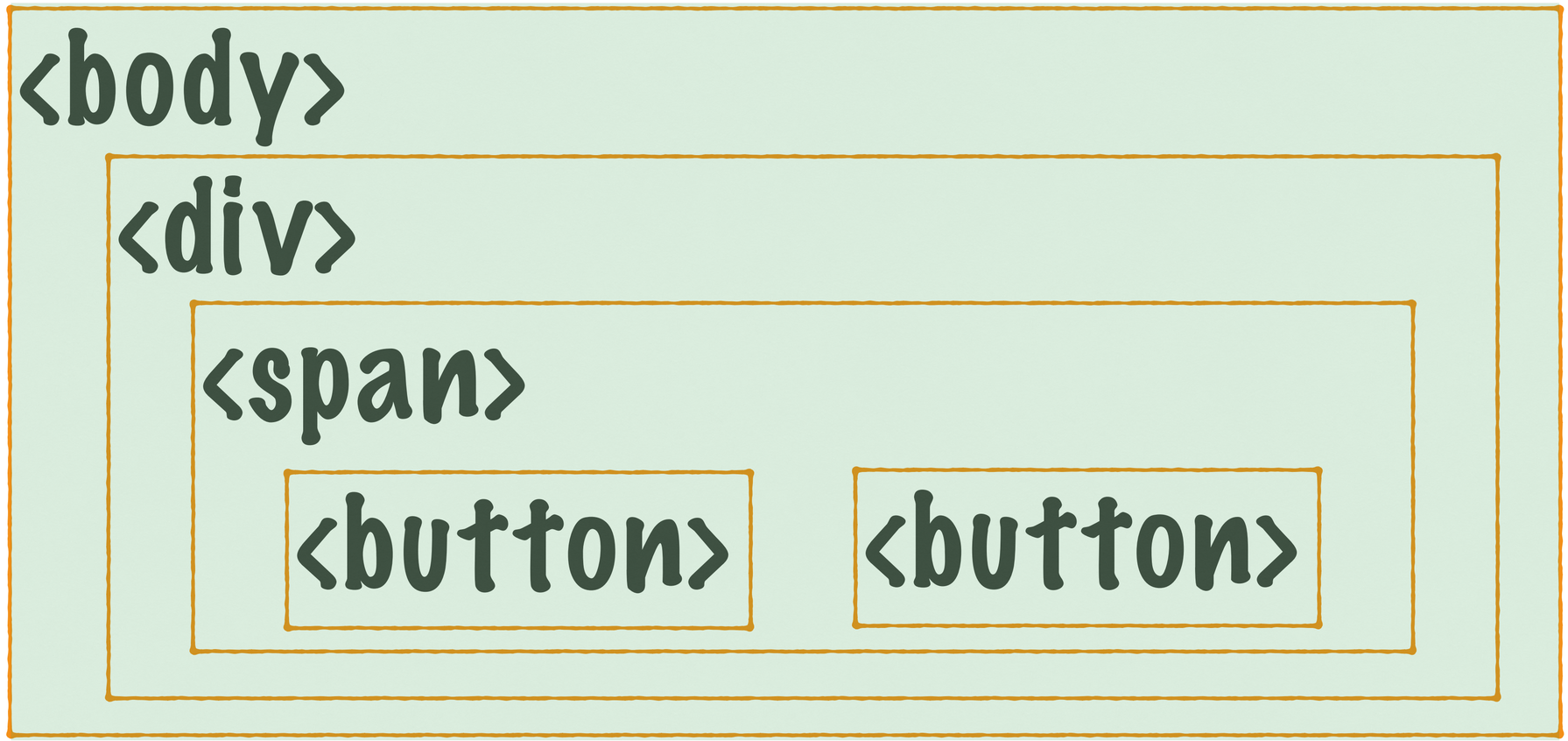
Event delegation to the rescue!



The event will be bubbled upwards

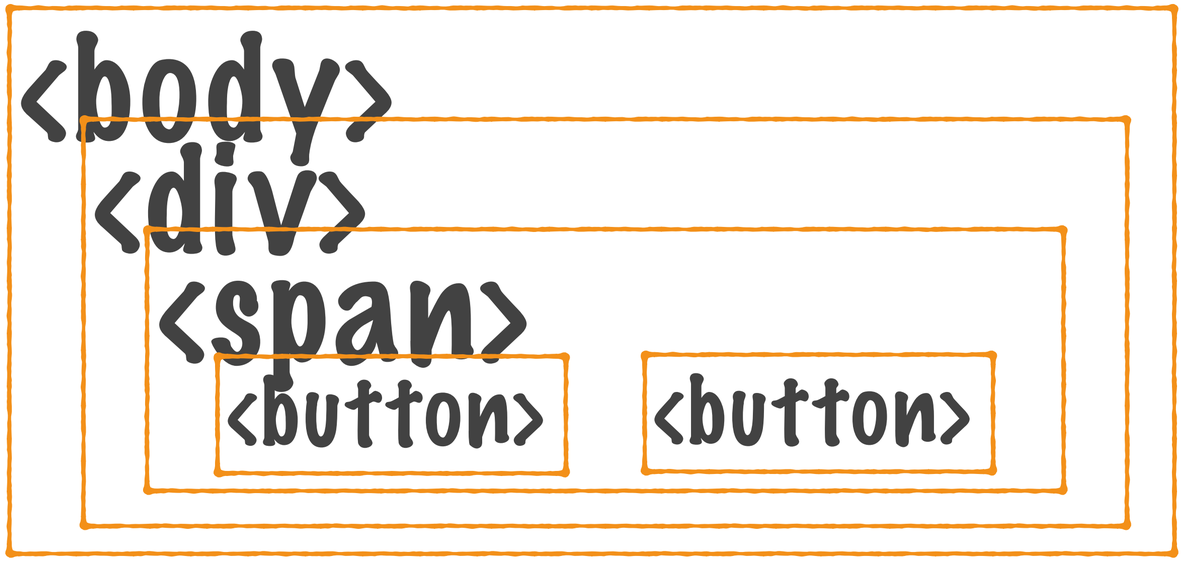
Event delegation

Event delegation to the rescue!



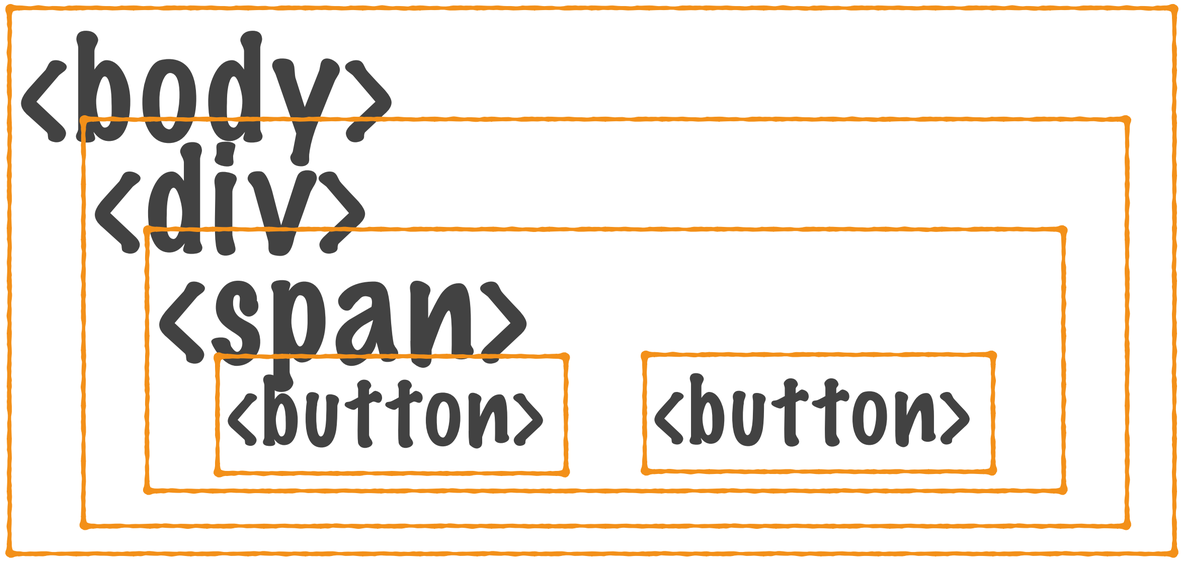
Till it reaches the <body> element

Event delegation



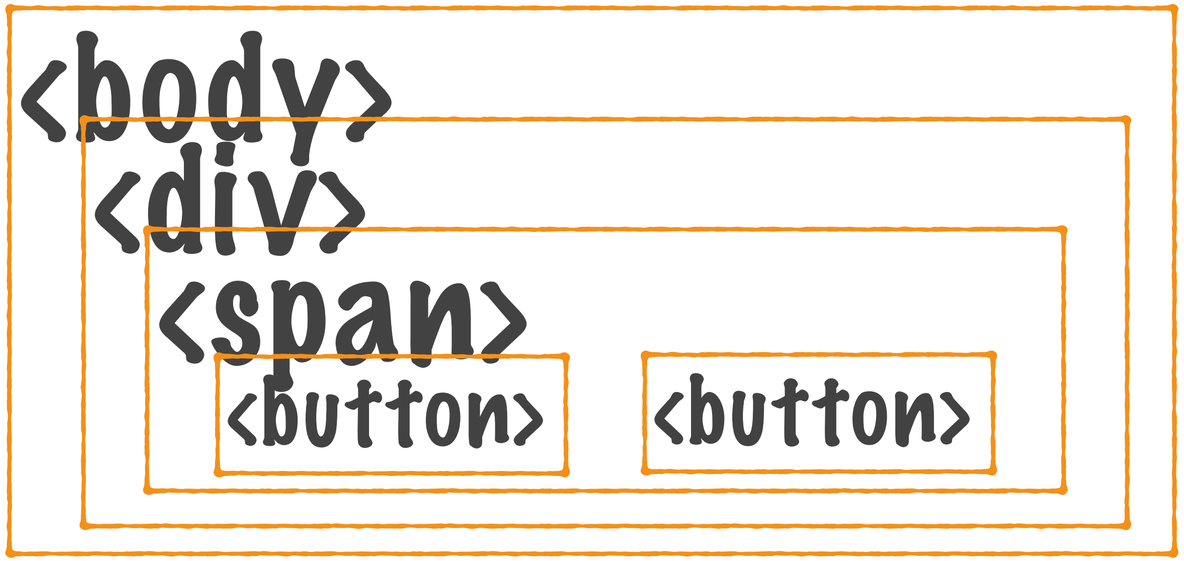
A listener on the <body> element will receive the events for all buttons within the body!

Event delegation



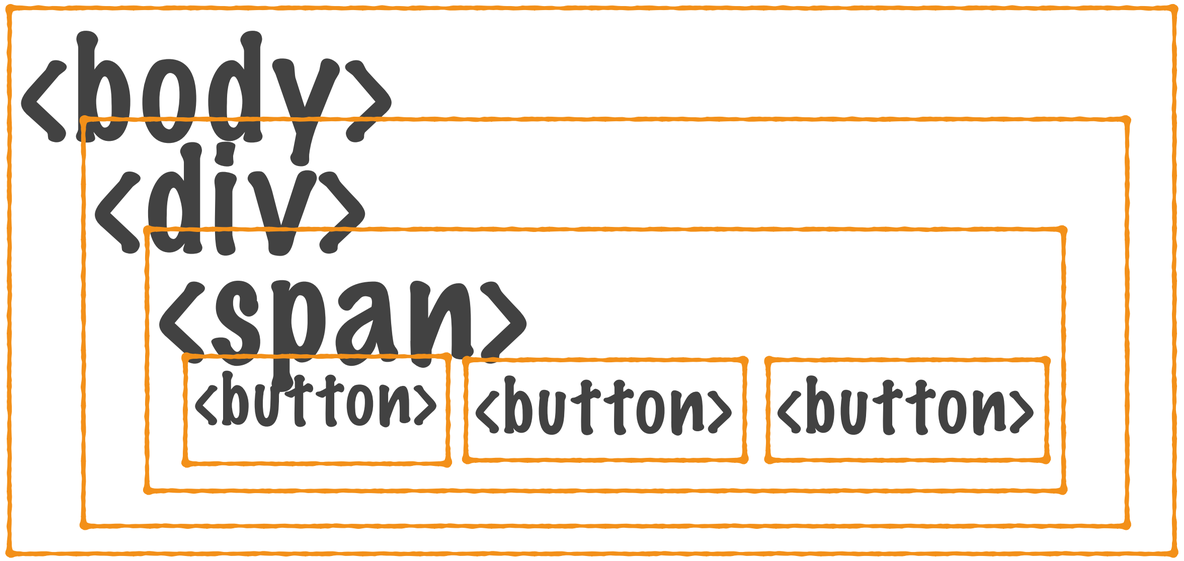
Delegate the event to be handled at a higher level than the button

Event delegation



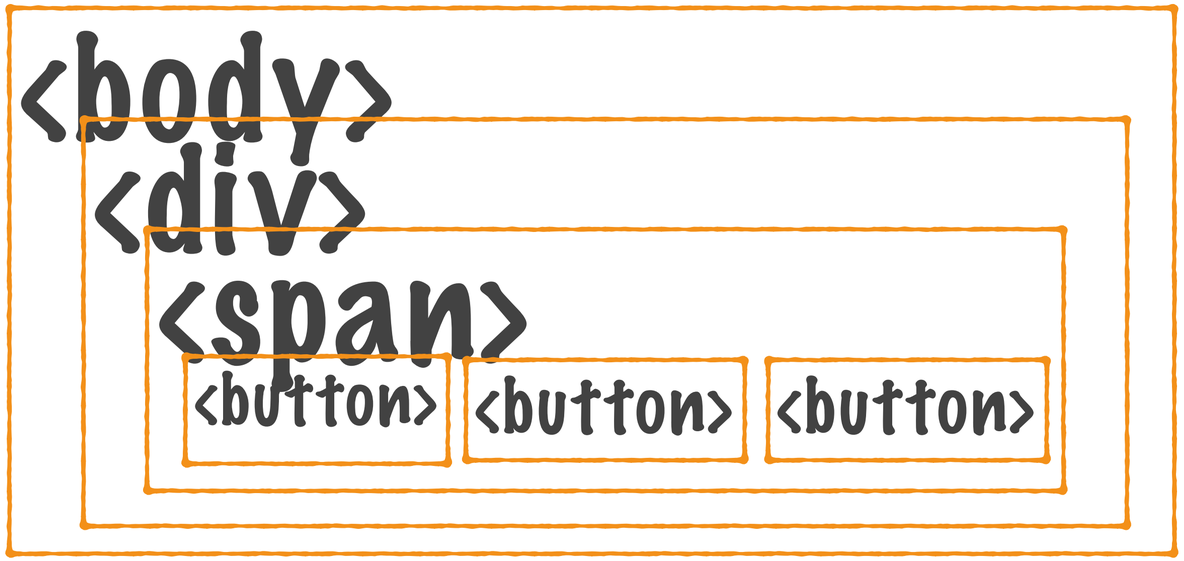
Any new buttons added within the <body> will automatically be hooked up with listeners!

Event delegation



Events from all buttons will reach the body element and can be handled

Event delegation



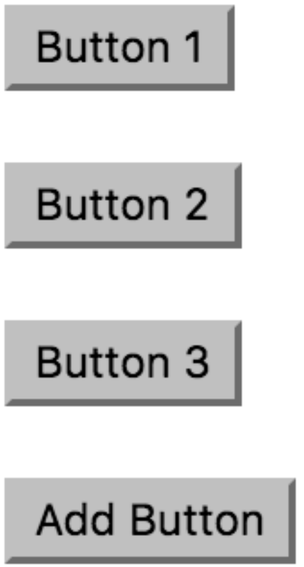
The buttons need not all exist when the delegated event handler is added

EXAMPLE 16

EventDelegation.html

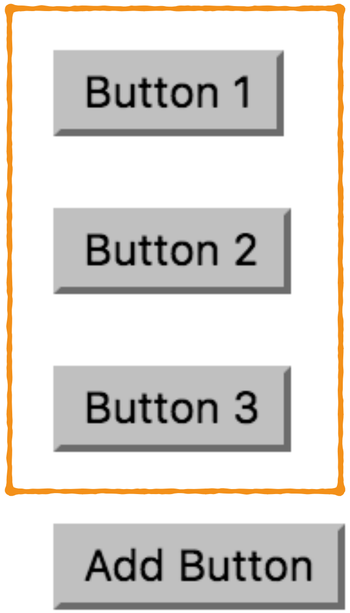
EventDelegation.html

```
<body>
<div>
    <button id="button1">Button 1</button>
    <br>
    <br>
    <button id="button2">Button 2</button>
    <br>
    <br>
    <button id="button3">Button 3</button>
    <br>
    <br>
</div>
<button id="addButton">Add Button</button>
<br>
<br>
</body>
```



EventDelegation.html

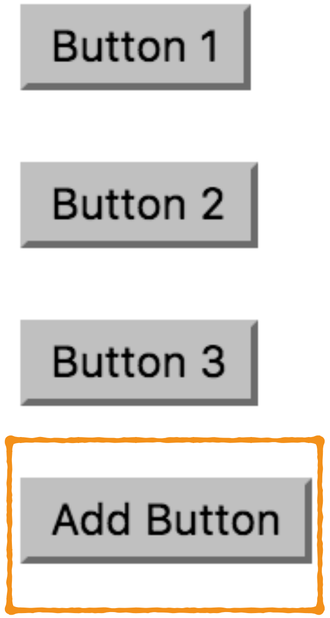
```
<body>
<div>
    <button id="button1">Button 1</button>
    <br>
    <br>
    <button id="button2">Button 2</button>
    <br>
    <br>
    <button id="button3">Button 3</button>
    <br>
    <br>
</div>
<button id="addButton">Add Button</button>
<br>
<br>
</body>
```



3 buttons within a

EventDelegation.html

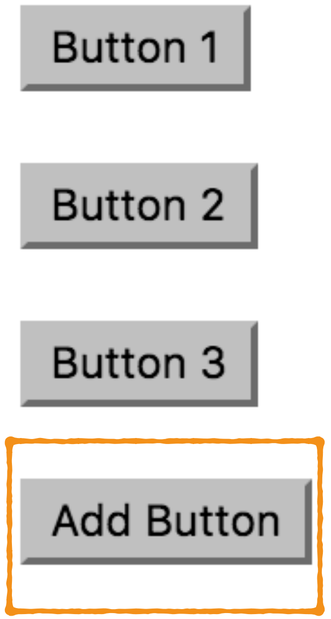
```
<body>
<div>
    <button id="button1">Button 1</button>
    <br>
    <br>
    <button id="button2">Button 2</button>
    <br>
    <br>
    <button id="button3">Button 3</button>
    <br>
    <br>
</div>
<button id="addButton">Add Button</button>
<br>
<br>
</body>
```



More buttons can be added dynamically

EventDelegation.html

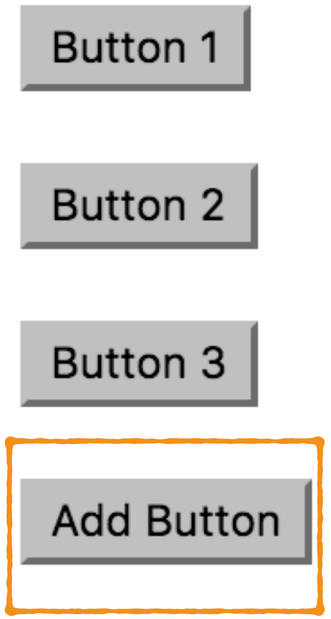
```
<body>
<div>
    <button id="button1">Button 1</button>
    <br>
    <br>
    <button id="button2">Button 2</button>
    <br>
    <br>
    <button id="button3">Button 3</button>
    <br>
    <br>
</div>
<button id="addButton">Add Button</button>
<br>
<br>
</body>
```



All buttons are hooked up to two event handlers

EventDelegation.html

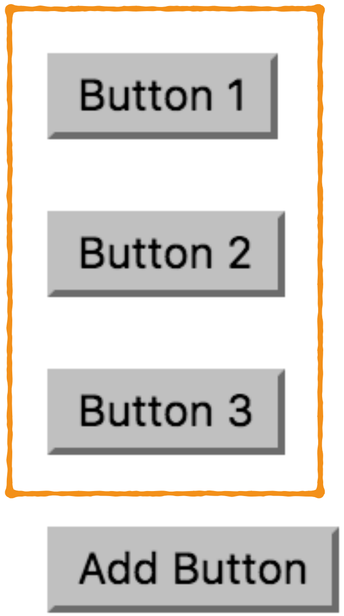
```
<body>
<div>
    <button id="button1">Button 1</button>
    <br>
    <br>
    <button id="button2">Button 2</button>
    <br>
    <br>
    <button id="button3">Button 3</button>
    <br>
    <br>
</div>
<button id="addButton">Add Button</button>
<br>
<br>
</body>
```



Even if the buttons did not exist at the time the event handlers are set up!

EventDelegation.html

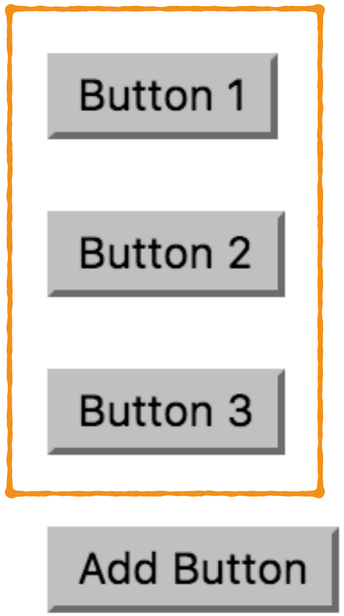
```
$(document).ready(function () {
    $('div').on('click', 'button', function() {
        console.log('You clicked on a button!');
    });

    $('div').on('click', 'button', function() {
        console.log('You clicked on this button: ' + $(this).text());
    });

    var index = 4;
    $('#addButton').on('click', function() {
        var buttonText = 'Button ' + index++;
        $('div').append("<button>" + buttonText + "</button><br><br>");

        console.log('Added button: ' + buttonText);
    });
});
```



This adds a delegated event listener on the outermost parent

EventDelegation.html

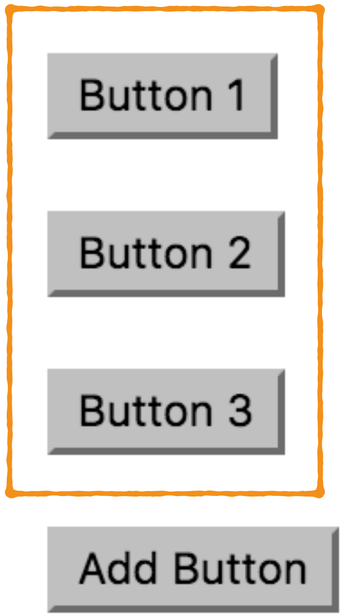
```
$(document).ready(function () {
    $('div').on('click', 'button', function() {
        console.log('You clicked on a button!');
    });

    $('div').on('click', 'button', function() {
        console.log('You clicked on this button: ' + $(this).text());
    });

    var index = 4;
    $('#addButton').on('click', function() {
        var buttonText = 'Button ' + index++;
        $('div').append("<button>" + buttonText + "</button><br><br>");

        console.log('Added button: ' + buttonText);
    });
});
```



Listen to all click events which bubble up to this

EventDelegation.html

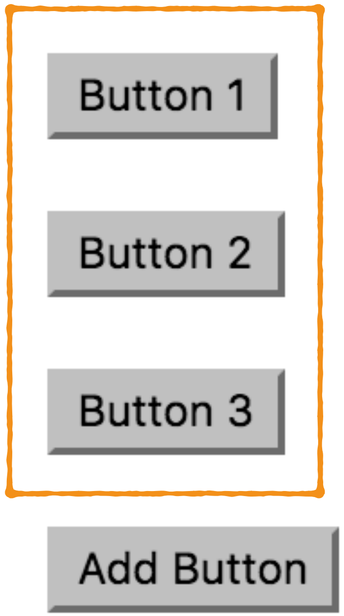
```
$(document).ready(function () {
    $('div').on('click', 'button', function() {
        console.log('You clicked on a button!');
     });

    $('div').on('click', 'button', function() {
        console.log('You clicked on this button: ' + $(this).text());
    });

    var index = 4;
    $('#addButton').on('click', function() {
        var buttonText = 'Button ' + index++;
        $('div').append("<button>" + buttonText + "</button><br><br>");

        console.log('Added button: ' + buttonText);
    });
});
```



The click events should originate from a button element

EventDelegation.html

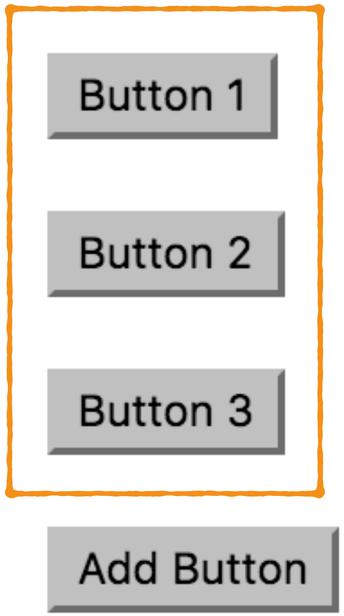
```
$(document).ready(function () {
    $('div').on('click', 'button', function() {
        console.log('You clicked on a button!');
    });

    $('div').on('click', 'button', function() {
        console.log('You clicked on this button: ' + $(this).text());
    });

    var index = 4;
    $('#addButton').on('click', function() {
        var buttonText = 'Button ' + index++;
        $('div').append("<button>" + buttonText + "</button><br><br>");

        console.log('Added button: ' + buttonText);
    });
});
```



Add one more click event on all buttons within the parent

EventDelegation.html

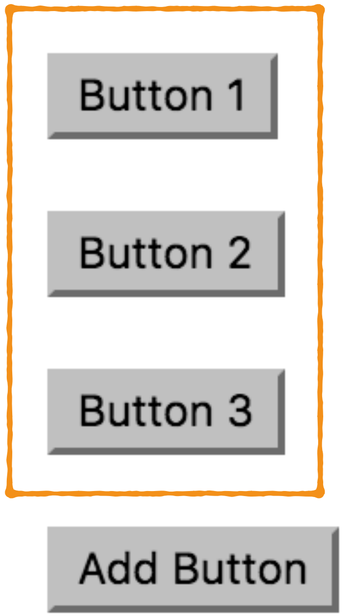
```
$(document).ready(function () {
    $('div').on('click', 'button', function() {
        console.log('You clicked on a button!');
    });

     $('div').on('click', 'button', function() {
         console.log('You clicked on this button: ' + $(this).text());
     });

    var index = 4;
    $('#addButton').on('click', function() {
        var buttonText = 'Button ' + index++;
        $('div').append("<button>" + buttonText + "</button><br><br>");

        console.log('Added button: ' + buttonText);
    });
});
```



Print out the text() on the button where the click originated

EventDelegation.html

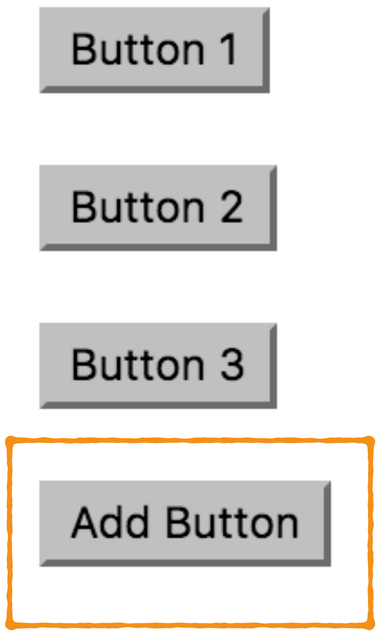
```
$(document).ready(function () {
    $('div').on('click', 'button', function() {
        console.log('You clicked on a button!');
    });

     $('div').on('click', 'button', function() {
         console.log('You clicked on this button: ' + $(this).text());
     });

    var index = 4;
    $('#addButton').on('click', function() {
        var buttonText = 'Button ' + index++;
        $('div').append("<button>" + buttonText + "</button><br><br>");

        console.log('Added button: ' + buttonText);
    });
});
```



$(this) will refer to the originator button

EventDelegation.html

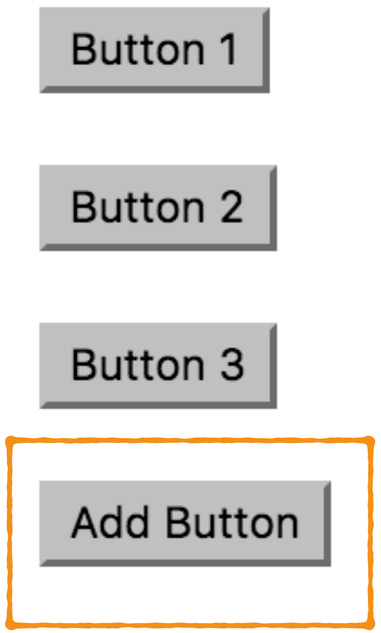
```
$(document).ready(function () {
    $('div').on('click', 'button', function() {
        console.log('You clicked on a button!');
    });

    $('div').on('click', 'button', function() {
        console.log('You clicked on this button: ' + $(this).text());
    });

    var index = 4;
    $('#addButton').on('click', function() {
        var buttonText = 'Button ' + index++;
        $('div').append("<button>" + buttonText + "</button><br><br>");

        console.log('Added button: ' + buttonText);
    });
});
```



This adds a new button to the same parent

EventDelegation.html

```
$(document).ready(function () {
    $('div').on('click', 'button', function() {
        console.log('You clicked on a button!');
    });

    $('div').on('click', 'button', function() {
        console.log('You clicked on this button: ' + $(this).text());
    });

    var index = 4;
    $('#addButton').on('click', function() {
        var buttonText = 'Button ' + index++;
        $('div').append("<button>" + buttonText + "</button><br><br>");

        console.log('Added button: ' + buttonText);
    });
});
```



All new buttons will automatically have the two previous event listeners hooked up

EventDelegation.html

Example16Demo

EXAMPLE 17

KeyEvents.html

KeyEvents.html

Examl17Demo

KeyEvents.html

```
<body>
    <h3>Last key pressed</h3>
    <div class="big">
    </div>
</body>
```

```
$(document).ready(function() {
    $(document).keyup(function(event) {
        var ch = String.fromCharCode(event.which);
        $('div').append(ch);
    });
});
```

KeyEvents.html

```
$(document).ready(function() {
    $(document).keyup(function(event) {
        var ch = String.fromCharCode(event.which);
        $('div').append(ch);
    });
});
```

Listen to the keyup event on the entire document

KeyEvents.html

```
$(document).ready(function() {
    $(document).keyup(function(event) {
        var ch = String.fromCharCode(event.which);
        $('div').append(ch);
    });
});
```

This is fired when you release a key

KeyEvents.html

```
$(document).ready(function() {
    $(document).keyup(function(event) {
        var ch = String.fromCharCode(event.which);
        $('div').append(ch);
    });
});
```

keydown, keypress are other events that you can listen to

KeyEvents.html

```
$(document).ready(function() {
    $(document).keyup(function(event) {
        var ch = String.fromCharCode(event.which);
        $('div').append(ch);
    });
});
```

Information about the key pressed is present in the **which** property of the event

KeyEvents.html

```
$(document).ready(function() {
    $(document).keyup(function(event) {
        var ch = String.fromCharCode(event.which);
        $('div').append(ch);
    });
});
```

This contains the Unicode values of the character pressed

KeyEvents.html

```
$(document).ready(function() {
    $(document).keyup(function(event) {
        var ch = String.fromCharCode(event.which);
        $('div').append(ch);
    });
});
```

Extract the character as a string using this static method on the JS string

KeyEvents.html

```
$(document).ready(function() {
    $(document).keyup(function(event) {
        var ch = String.fromCharCode(event.which);
        $('div').append(ch);
    });
});
```

And append() it to the where you want it displayed

EXAMPLE 18

EventTriggers.html

EventTriggers.html

Any event that is added via jQuery functions such as on(), click() etc can be triggered by jQuery

jQuery stores a reference to every event handler that it adds

EventTriggers.html

jQuery stores a reference to every event handler that it adds

Using the trigger() method allows you to invoke this event handler

EventTriggers.html

jQuery stores a reference to every event handler that it adds

trigger() cannot be used for native browser events like clicking on links

...only for events added via jQuery

EventTriggers.html

jQuery stores a reference to every event handler that it adds

Triggering allows you to test that your handler works correctly

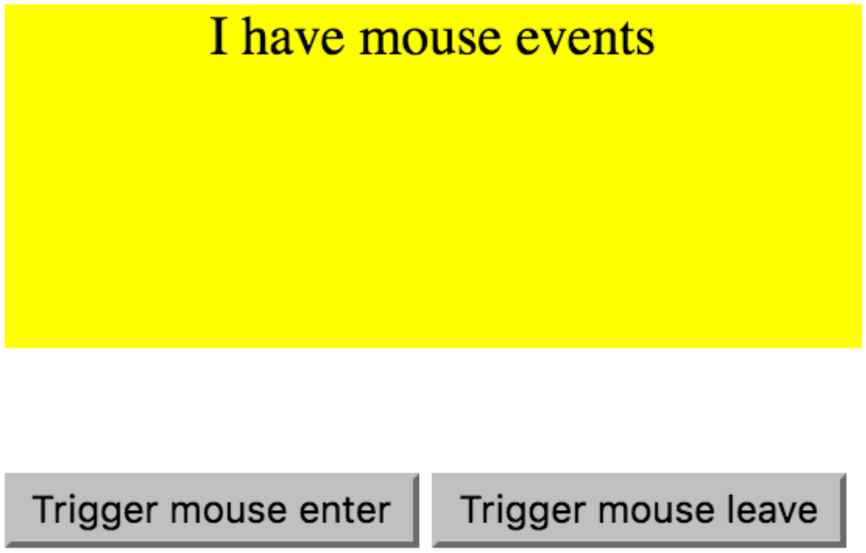
Testability is a huge reason to use this trigger() method

EventTriggers.html

Example18Demo

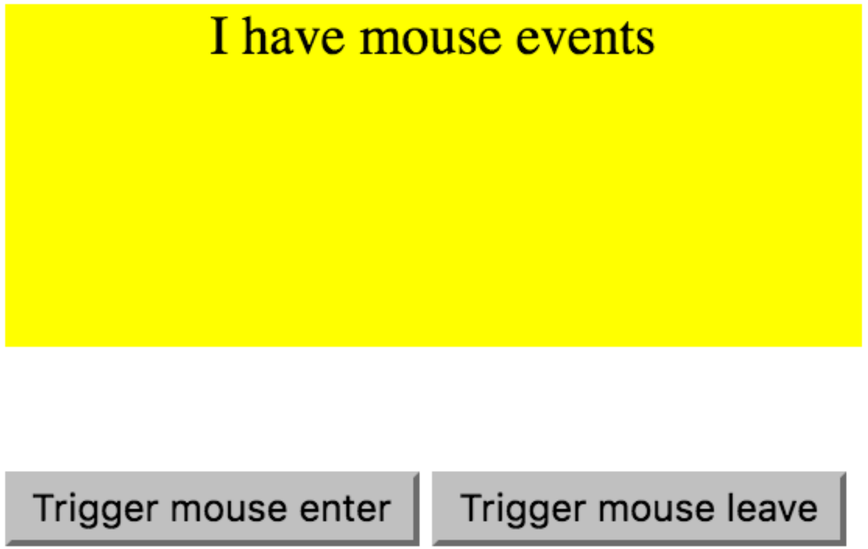
EventTriggers.html

```
<div class="big">I have mouse events</div>
<br>
<br>
<button id="triggerButton1">Trigger mouse enter</button>
<button id="triggerButton2">Trigger mouse leave</button>
<br>
```



EventTriggers.html

```
$(document).ready(function () {
    $('.big').on('mouseenter', function() {
        console.log('Mouseenter!');
    });
    $('.big').on('mouseleave', function() {
        console.log('Mouseleave!');
    });

    $('#triggerButton1').on('click', function() {
        $('.big').trigger('mouseenter');
    });
    $('#triggerButton2').on('click', function() {
        $('.big').trigger('mouseleave');
    });
});
```



EventTriggers.html

```
$(document).ready(function () {
    $('.big').on('mouseenter', function() {
        console.log('Mouseenter!');
    });
    $('.big').on('mouseleave', function() {
        console.log('Mouseleave!');
    });

    $('#triggerButton1').on('click', function() {
        $('.big').trigger('mouseenter');
    });
    $('#triggerButton2').on('click', function() {
        $('.big').trigger('mouseleave');
    });
});
```

Add 2 event handlers for mouseenter and mouseleave

EventTriggers.html

```
$(document).ready(function () {
    $('.big').on('mouseenter', function() {
        console.log('Mouseenter!');
    });
    $('.big').on('mouseleave', function() {
        console.log('Mouseleave!');
    });

    $('#triggerButton1').on('click', function() {
        $('.big').trigger('mouseenter');
    });
    $('#triggerButton2').on('click', function() {
        $('.big').trigger('mouseleave');
    });
});
```

Use the trigger() on the element to invoke the handler

EventTriggers.html

```
$(document).ready(function () {
    $('.big').on('mouseenter', function() {
        console.log('Mouseenter!');
    });
    $('.big').on('mouseleave', function() {
        console.log('Mouseleave!');
    });

    $('#triggerButton1').on('click', function() {
        $('.big').trigger('mouseenter');
    });
    $('#triggerButton2').on('click', function() {
        $('.big').trigger('mouseleave');
    });
});
```

Specify the event you want to trigger

EventTriggers.html

```
$(document).ready(function () {
    $('.big').on('mouseenter', function() {
        console.log('Mouseenter!');
    });
    $('.big').on('mouseleave', function() {
        console.log('Mouseleave!');
    });

    $('#triggerButton1').on('click', function() {
        $('.big').trigger('mouseenter');
    });
    $('#triggerButton2').on('click', function() {
        $('.big').trigger('mouseleave');
    });
});
```

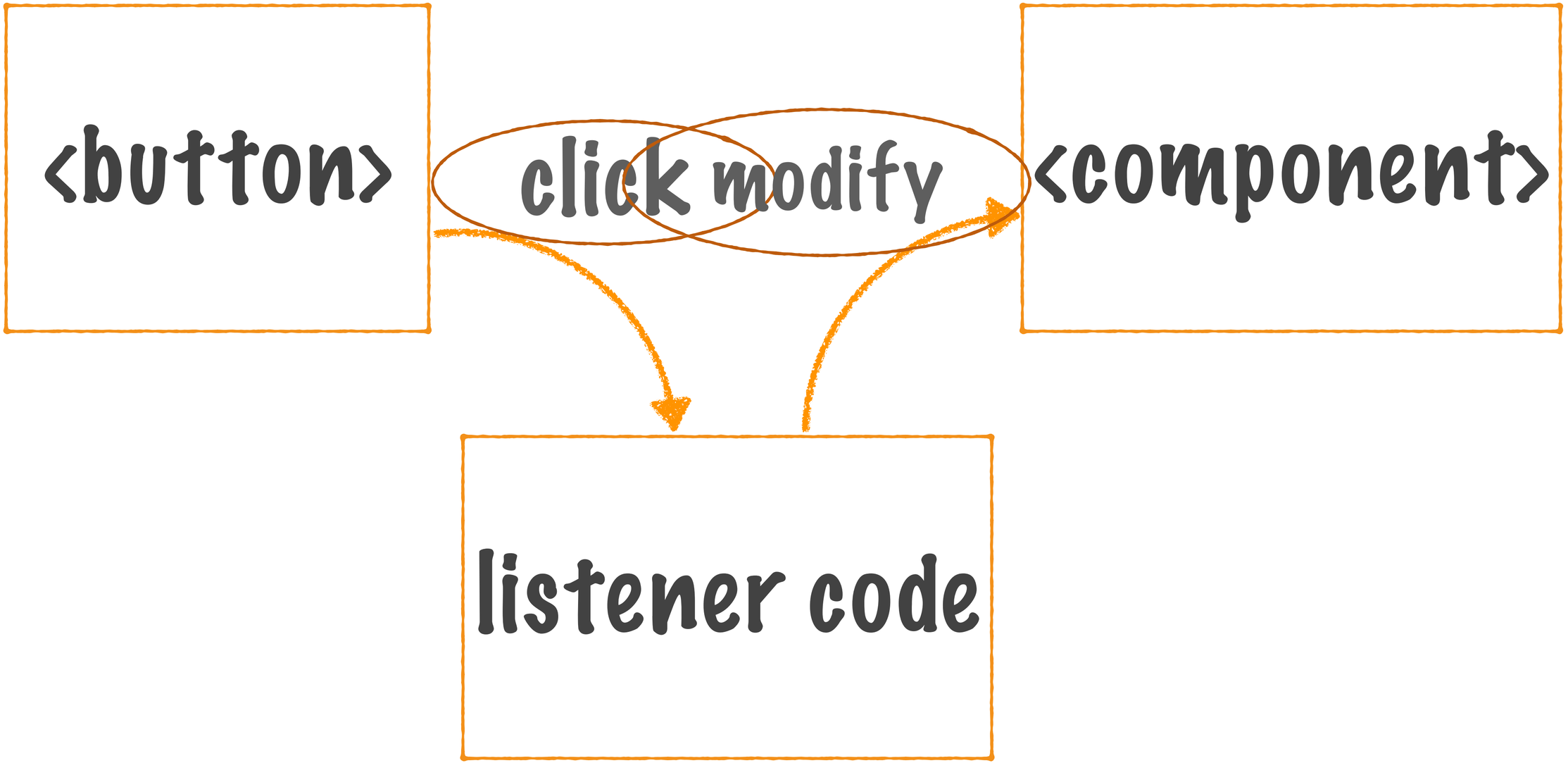
Note that both the handlers were previously added using jQuery

JQUERY

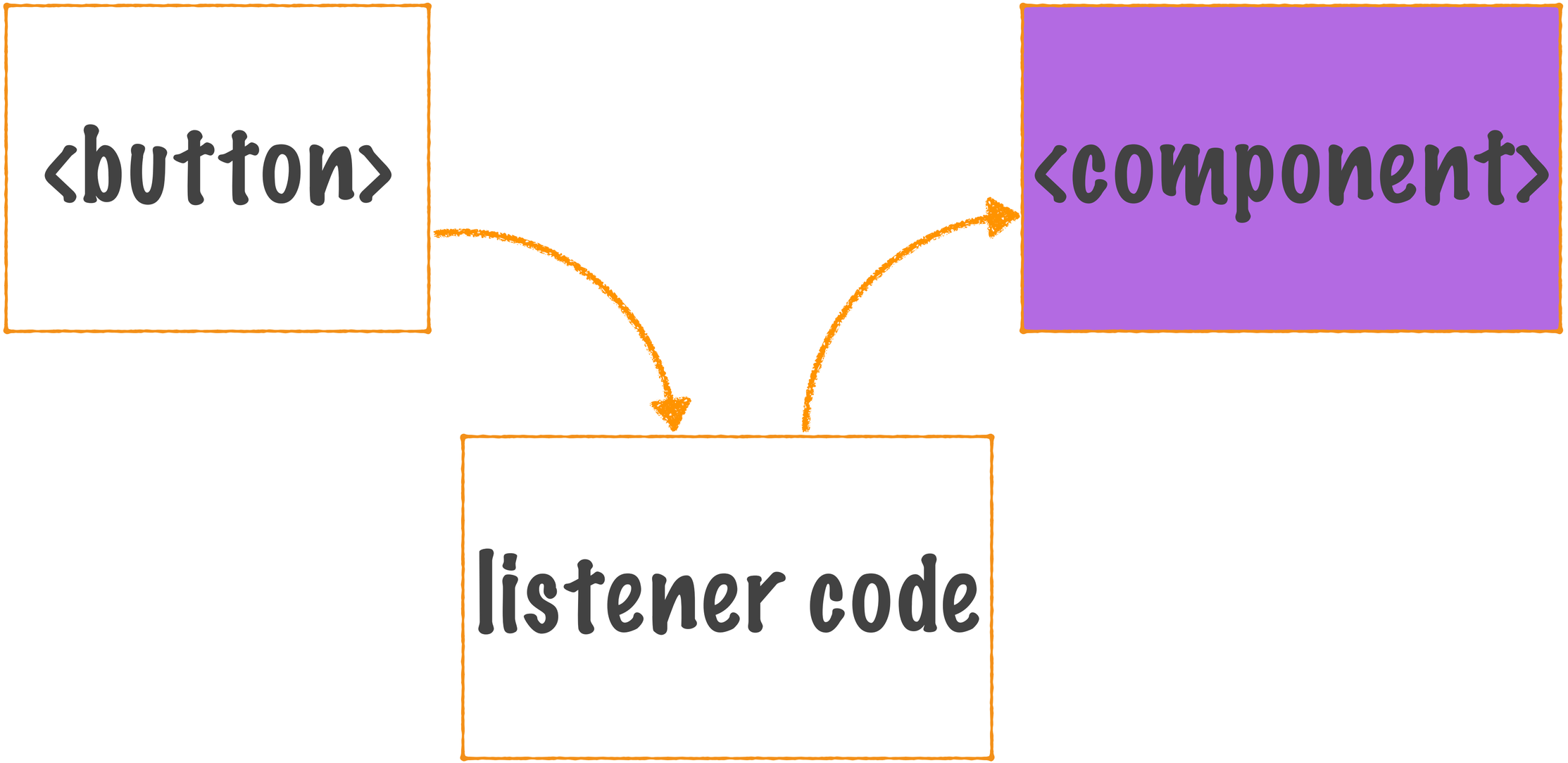
Custom Events

Let's see how normal event handlers work

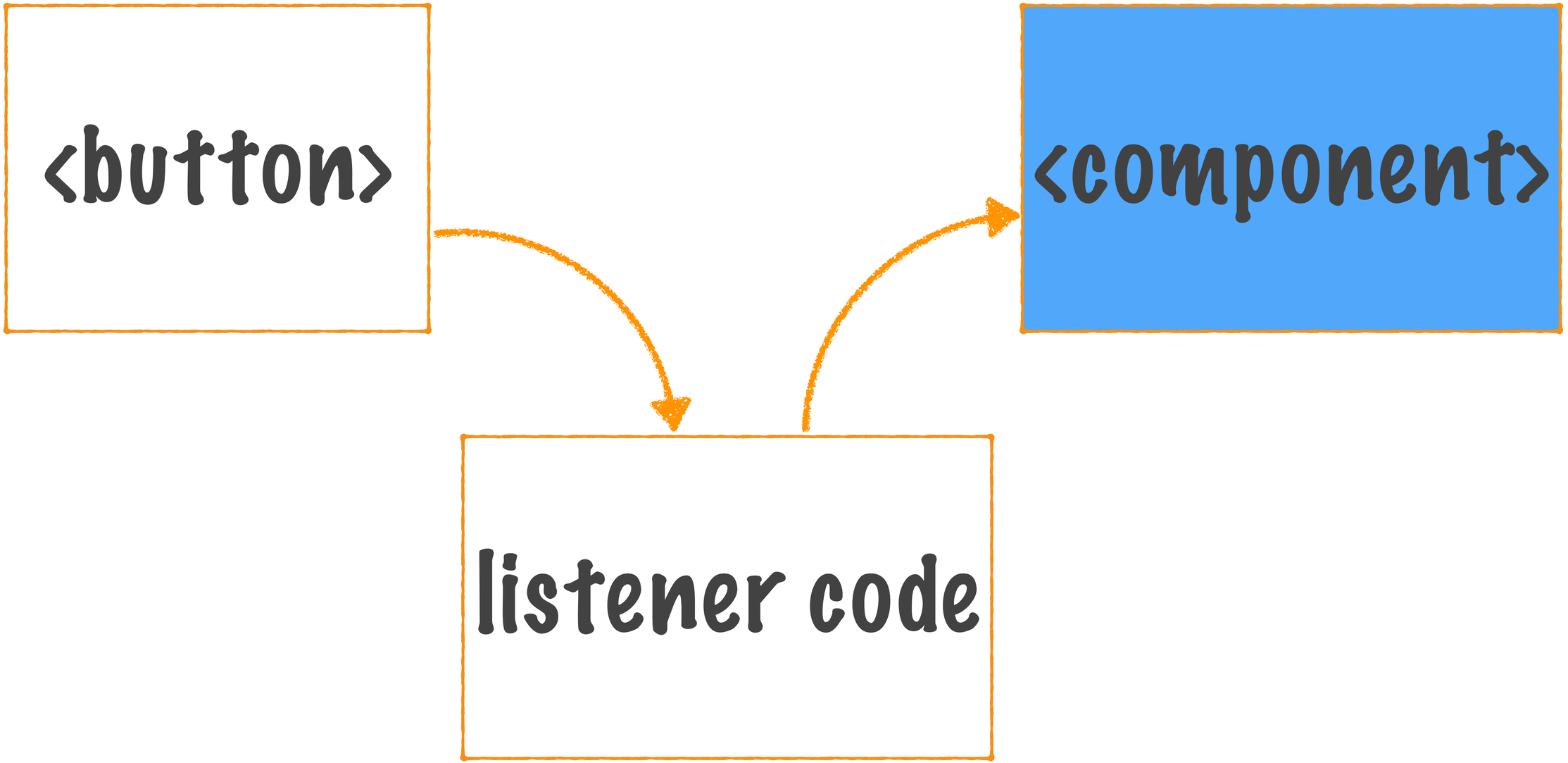
Custom Events



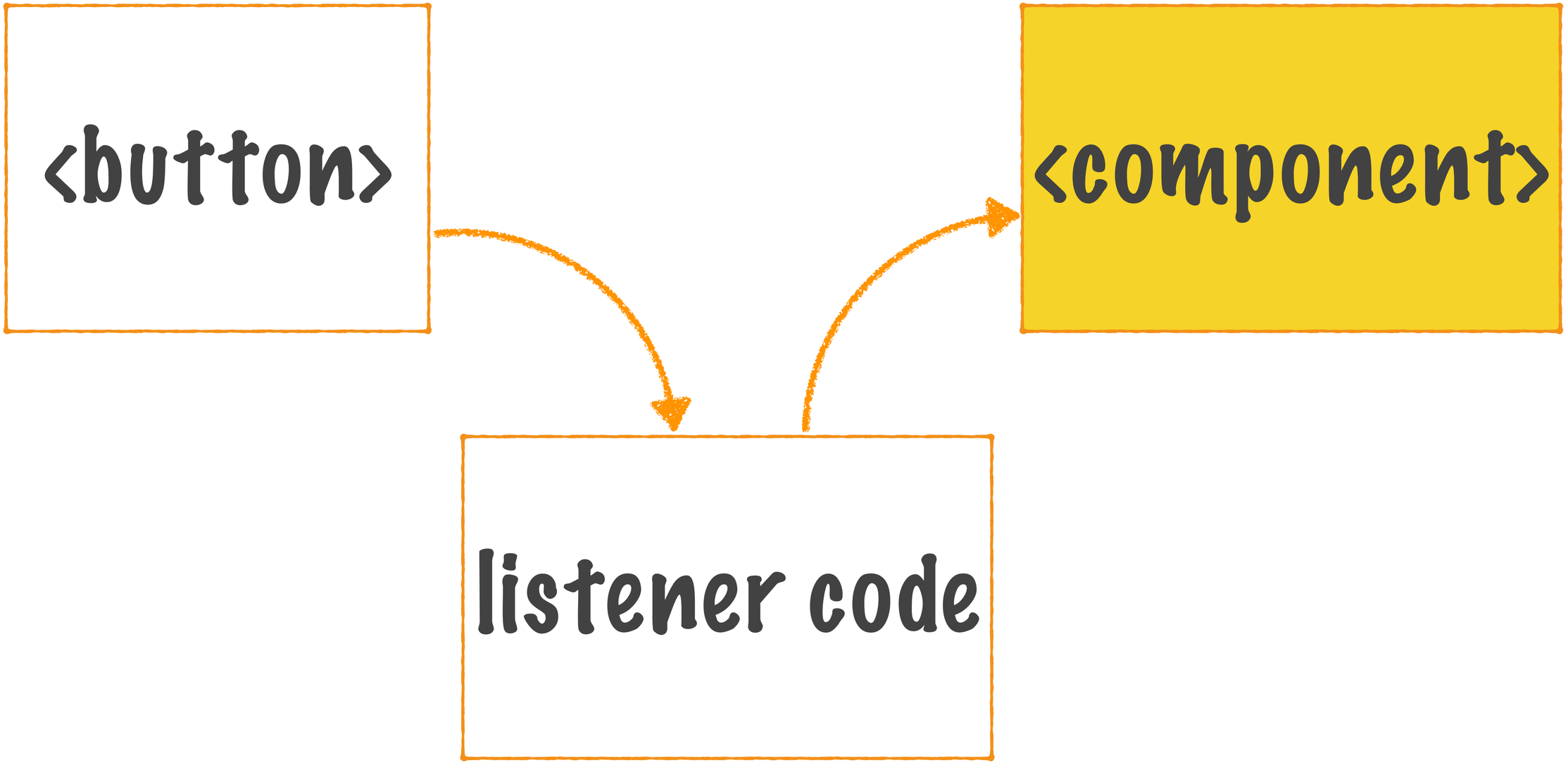
Custom Events



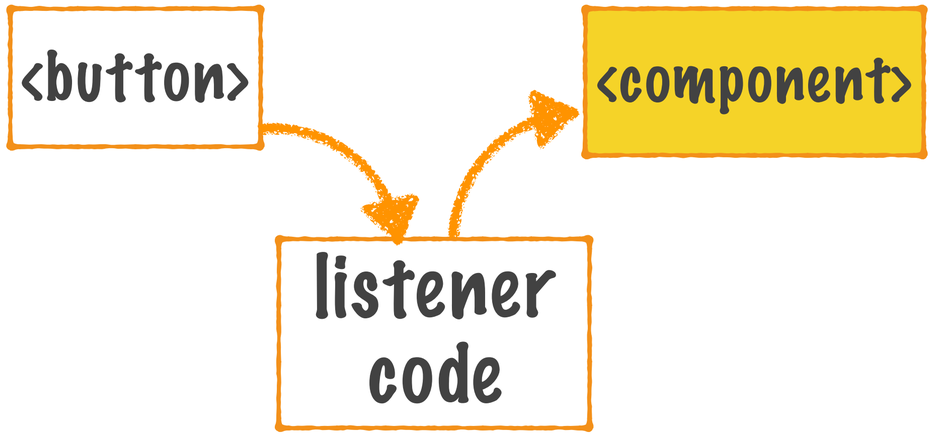
Custom Events



Custom Events

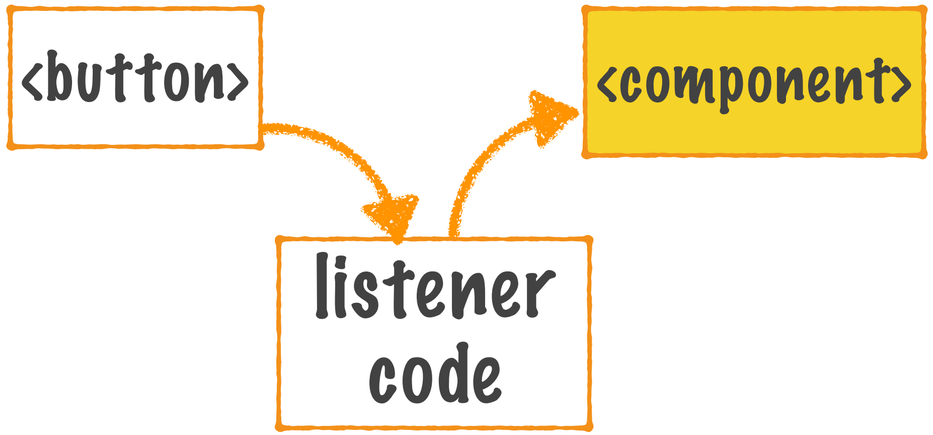


Custom Events



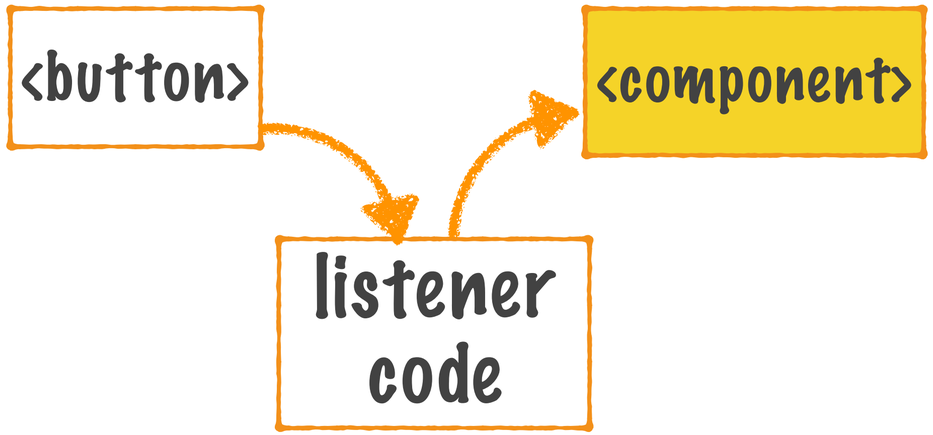
This pattern focuses attention on the element which triggers the listener

Custom Events



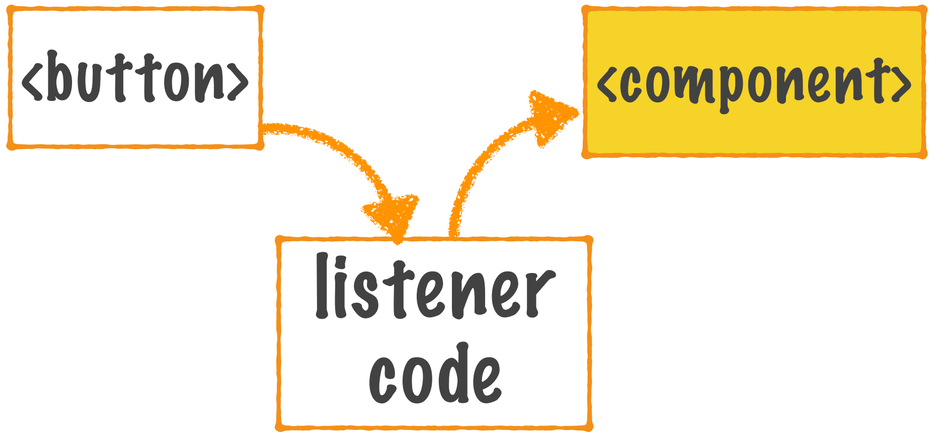
Rather than the element which is affected

Custom Events



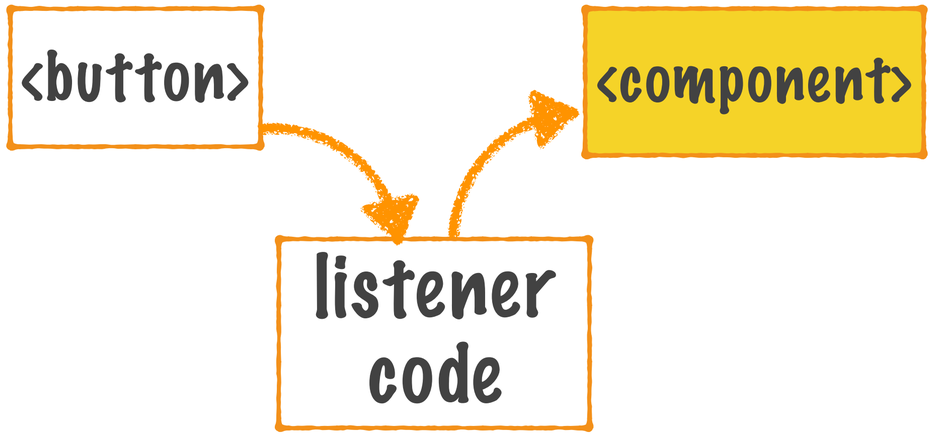
Let's turn this on its head

Custom Events



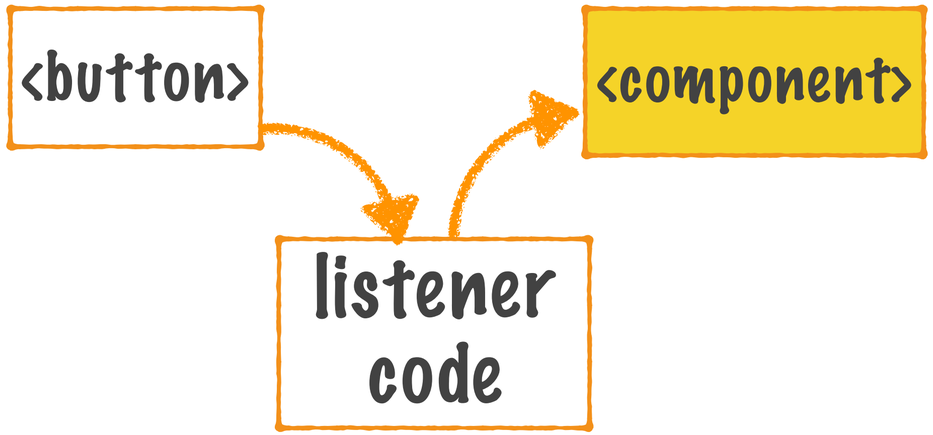
What if there was a way for the component to announce that it wants its color changed?

Custom Events



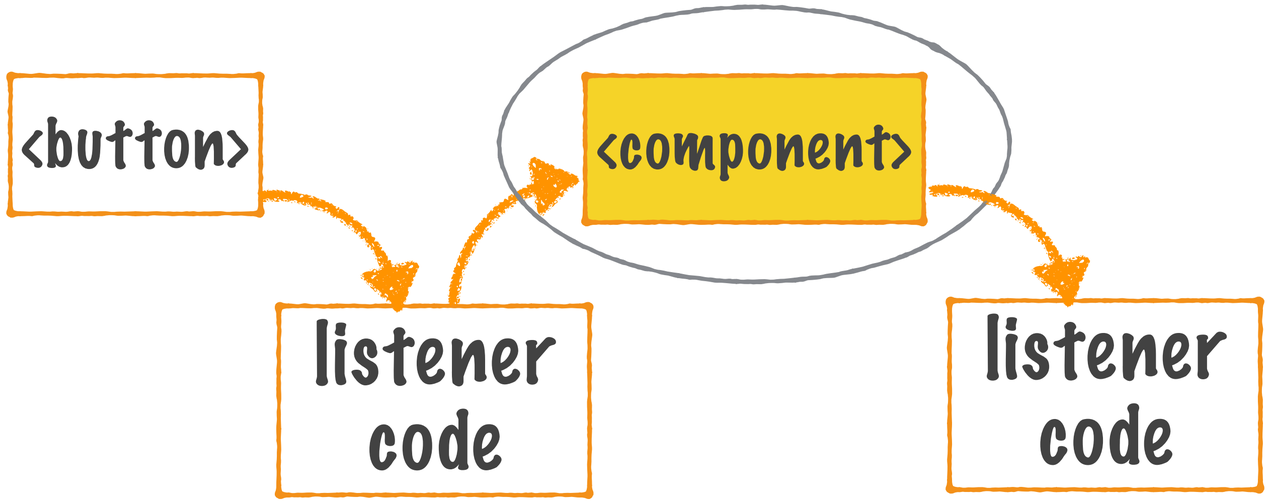
This puts the spotlight on the component being acted upon

Custom Events



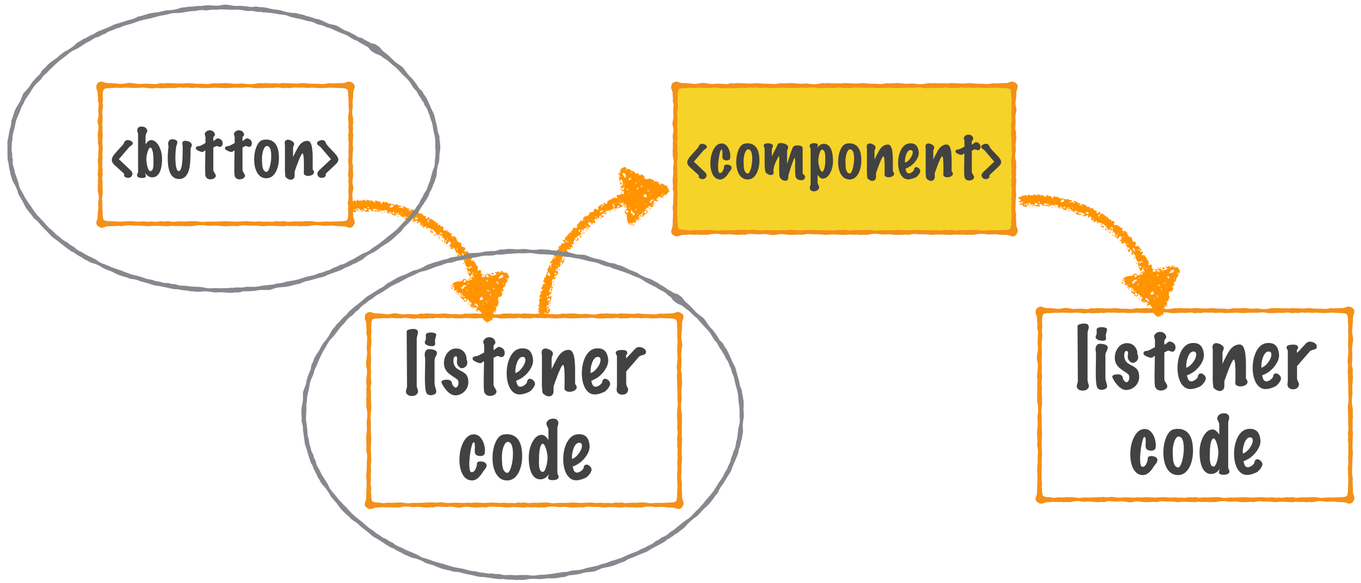
Behaviors are more clearly associated with the target element

Custom Events



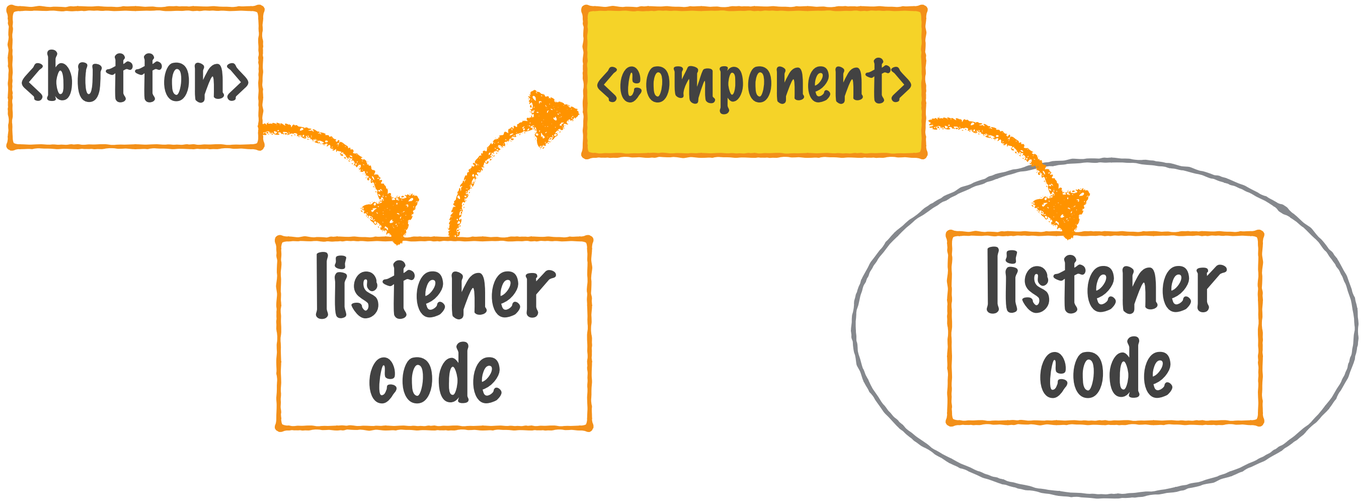
Listen to color change events on the <component>

Custom Events



Trigger the color change event on the component when the button is clicked

Custom Events



The color change event listener for colors will be triggered

EXAMPLE 19

CustomEvents.html

CustomEvents.html

Example19Demo

CustomEvents.html

```
<body>
<div class="box">
    <button id="changeColor1">Change</button>
</div>
<br>
<div class="box">
    <button id="changeColor2">Change</button>
</div>
<br>
<div class="box">
    <button id="changeColor3">Change</button>
</div>
<br>
<div class="box">
    <button id="changeColor4">Change</button>
</div>
<br>
</body>
```

CustomEvents.html

```
$(document).ready(function () {
    $('.box').on('color:change', function() {
        var colors = ['red', 'blue', 'green', 'yellow', 'grey', 'pink', 'brown', 'fuchsia'];
        var box = $(this);
        box.css('background-color', colors[Math.floor(Math.random() * colors.length)]);
    });

    $('body').on('click', 'button', function() {
        $(this).closest('.box').trigger('color:change');
    });
});
```

Listen to events on every box

CustomEvents.html

```
$(document).ready(function () {
    $('.box').on('color:change', function() {
        var colors = ['red', 'blue', 'green', 'yellow', 'grey', 'pink', 'brown', 'fuchsia'];
        var box = $(this);
        box.css('background-color', colors[Math.floor(Math.random() * colors.length)]);
    });

    $('body').on('click', 'button', function() {
        $(this).closest('.box').trigger('color:change');
    });
});
```

Notice how we name the events - 'color:change'

CustomEvents.html

```
$(document).ready(function () {
    $('.box').on('color:change', function() {
        var colors = ['red', 'blue', 'green', 'yellow', 'grey', 'pink', 'brown', 'fuchsia'];
        var box = $(this);
        box.css('background-color', colors[Math.floor(Math.random() * colors.length)]);
    });

    $('body').on('click', 'button', function() {
        $(this).closest('.box').trigger('color:change');
    });
});
```

This makes it unlikely that it will clash with any of the browser events

CustomEvents.html

```
$(document).ready(function () {
    $('.box').on('color:change', function() {
        var colors = ['red', 'blue', 'green', 'yellow', 'grey', 'pink', 'brown', 'fuchsia'];
        var box = $(this);
        box.css('background-color', colors[Math.floor(Math.random() * colors.length)]);
    });

    $('body').on('click', 'button', function() {
        $(this).closest('.box').trigger('color:change');
    });
});
```

These are the colors that the box cycles through

CustomEvents.html

```
$(document).ready(function () {
    $('.box').on('color:change', function() {
        var colors = ['red', 'blue', 'green', 'yellow', 'grey', 'pink', 'brown', 'fuchsia'];
        var box = $(this);
        box.css('background-color', colors[Math.floor(Math.random() * colors.length)]);
    });

    $('body').on('click', 'button', function() {
        $(this).closest('.box').trigger('color:change');
    });
});
```

Set the box element to a new color at random

CustomEvents.html

```
$(document).ready(function () {
    $('.box').on('color:change', function() {
        var colors = ['red', 'blue', 'green', 'yellow', 'grey', 'pink', 'brown', 'fuchsia'];
        var box = $(this);
        box.css('background-color', colors[Math.floor(Math.random() * colors.length)]);
    });

    $('body').on('click', 'button', function() {
        $(this).closest('.box').trigger('color:change');
    });
});
```

the .css() function allows us to set CSS styles on the element

CustomEvents.html

```
$(document).ready(function () {
    $('.box').on('color:change', function() {
        var colors = ['red', 'blue', 'green', 'yellow', 'grey', 'pink', 'brown', 'fuchsia'];
        var box = $(this);
        box.css('background-color', colors[Math.floor(Math.random() * colors.length)]);
    });

    $('body').on('click', 'button', function() {
        $(this).closest('.box').trigger('color:change');
    });
});
```

A click on the button will trigger the 'color:change' event

CustomEvents.html

```
$(document).ready(function () {
    $('.box').on('color:change', function() {
        var colors = ['red', 'blue', 'green', 'yellow', 'grey', 'pink', 'brown', 'fuchsia'];
        var box = $(this);
        box.css('background-color', colors[Math.floor(Math.random() * colors.length)]);
    });

    $('body').on('click', 'button', function() {
        $(this).closest('.box').trigger('color:change');
    });
});
```

The element with the "box" class which is **closest** to the button fires the event

CustomEvents.html

```
$(document).ready(function () {
    $('.box').on('color:change', function() {
        var colors = ['red', 'blue', 'green', 'yellow', 'grey', 'pink', 'brown', 'fuchsia'];
        var box = $(this);
        box.css('background-color', colors[Math.floor(Math.random() * colors.length)]);
    });

    $('body').on('click', 'button', function() {
        $(this).closest('.box').trigger('color:change');
    });
});
```

And is the one whose color is updated