# Web Application Development

Events

## WHAT IS AN EVENT?

Events are the browser’s way of saying, “Hey, this just happened.”

When an event fires, your script can then react by running code (e.g. a function).

By running code when an event fires, your website responds to the user’s actions.

It becomes interactive.

## DIFFERENT EVENT TYPES

## USER INTERFACE (UI) EVENTS

These occur when a user interacts with the browsers user interface (UI) rather than the web page.
The following slides show a selection of these events.

## USER INTERFACE (UI) EVENTS

| Event | Description |
|---|---|
| load | Web page has finished loading |
| Unload | Web page is unloading (usually because a new page was requested) |
| error | Browser encounters a JavaScript error or an asset doesn’t exist |
| resize | Browser window has been resized |
| scroll | User has scrolled u or down the page |

## KEYBOARD EVENTS

These occur when a user interacts with the keyboard.

## KEYBOARD EVENTS

| Event | Description |
|---|---|
| keydown | User first presses a key |
| keyup | User releases a key |
| keypress | Character is being inserted |

## MOUSE EVENTS

These occur when a user interacts with a mouse.

## MOUSE EVENTS

| Event | Description |
|---|---|
| click | User presses and releases a button over the same element |
| dblclick | User presses and releases a button twice over the same element |
| mousedown | User presses a mouse button while over an element |
| mouseup | User releases a mouse button while over an element |
| mousemove | User moves the moue |
| mouseover | User moves the mouse over an element |
| mouseout | User moves the mouse off an element |

## FOCUS EVENTS

These occur when an element gains or loses focus.

## FOCUS EVENTS

| Event | Description |
|---|---|
| focus / focusin | Element gains focus |
| blur / foucusout | Element loses focus |

## FORM EVENTS

These occur when a user interacts with a form element.

## FORM EVENTS

| Event | Description |
|---|---|
| input | Value in any <input> or <textarea> element has changed(IE 9+) or any element with the contenteditable attribute set |
| change | Value in select box, checkbox or radio button changes (IE 9+) |
| submit | User submits a form using a button or a key |
| reset | User clicks on a forms reset button |
| cut | User cuts content from the form field |
| copy | User cuts copies from the form field |
| paste | User pastes content into the form field |
| select | User selects some text in a form field |

## HOW EVENTS TRIGGER JAVASCRIPT CODE.
## - Event Handlers

# 1

1

Select the element node(s) the script should respond to

1

Select the element node(s) the script should respond to

2

1

Select the element node(s) the script should respond to

2

Indicate the event on the selected node(s) that will trigger a response

1

Select the element node(s) the script should respond to

2

Indicate the event on the selected node(s) that will trigger a response

3

1

Select the element node(s) the script should respond to

2

Indicate the event on the selected node(s) that will trigger a response

3

State the code you want to run when the event occurs

## BINDING AN EVENT TO AN ELEMENT

There are three ways to bind an event to an element:

HTML event handler attributes
Traditional DOM event handlers
DOM Level 2 event listeners

The following examples show a blur event on an element stored in a variable called `el` that triggers a function called `checkUsername()`.

These examples show the three different ways for handling events:
- HTML event handler attributes
- Traditional DOM event handlers
- DOM Level 2 event listeners

## HTML EVENT HANDLER ATTRIBUTES
## (DO NOT USE)

```
<input type=“text” id=“username”
       onblur=“checkUsername()”>
```

## HTML EVENT HANDLER ATTRIBUTES (DO NOT USE)

ELEMENT

```
<input type=“text” id=“username”
       onblur=“checkUsername()”>
```

Do not use this way to handle events but just be aware if reviewing older code.

## HTML EVENT HANDLER ATTRIBUTES
## (DO NOT USE)

```
<input type=“text” id=“username”
       onblur=“checkUsername()”>
```

EVENT

## HTML EVENT HANDLER ATTRIBUTES (DO NOT USE)

```
<input type=“text” id=“username”
       onblur=“checkUsername()”>
```

$\underbrace{\texttt{checkUsername()}}_{\text{FUNCTION}}$

## TRADITIONAL DOM EVENT HANDLERS

```
el.onblur = checkUsername();
```

This way allows you separate the JavaScript from the HTML. The main drawback of this approach is that you can only attach a single function to a any event.

## TRADITIONAL DOM EVENT HANDLERS

```
el.onblur = checkUsername();
```

ELEMENT

## TRADITIONAL DOM EVENT HANDLERS

$$\texttt{el.}\underbrace{\texttt{onblur}}_{\text{EVENT}}\texttt{ = checkUsername();}$$

## TRADITIONAL DOM EVENT HANDLERS

$$\texttt{el.onblur = }\underbrace{\texttt{checkUsername();}}_{\text{FUNCTION}}$$

## EVENT LISTENERS

```
el.addEventListener(‘blur’, checkUsername, false);
```

This is now the favoured way of handling events.

## EVENT LISTENERS

```
el.addEventListener('blur', checkUsername, false);
```

$\underbrace{\texttt{el}}_{\text{ELEMENT}}$

## EVENT LISTENERS

```
el.addEventListener(‘blur’, checkUsername, false);
```

$\underbrace{\texttt{‘blur’}}_{\text{EVENT}}$

## EVENT LISTENERS

```
el.addEventListener('blur', checkUsername, false);
```

$\underbrace{\texttt{checkUsername,}}_{\text{FUNCTION}}$

## EVENT LISTENERS

```
el.addEventListener('blur', checkUsername, false);
```

$\underbrace{\texttt{false}}$

BOOLEAN (OPTIONAL), Determines Event Flow

Because you cannot have parentheses after the function names in event handlers or listeners, passing arguments requires a workaround.

## PARAMETERS WITH EVENT LISTENERS

```
el.addEventListener(‘blur’, function() {
   checkUsername(5);
}, false);
```

## PARAMETERS WITH EVENT LISTENERS

```
el.addEventListener('blur', function() {
    checkUsername(5);
}, false);
```

An anonymous function is used as the second argument.

## PARAMETERS WITH EVENT LISTENERS

```
el.addEventListener('blur', function() {
    checkUsername(5);
}, false);
```

Inside the anonymous function, a named function is called.

IE5 - 8 had a different event model and did not support `addEventListener()` but you can provide fallback code to make event listeners work with older versions of IE.

## SUPPORTING OLDER VERSIONS OF IE

```
if (el.addEventListener) {

  el.addEventListener(‘blur’, function() {
    checkUsername(5);
  }, false);

} else {

  el.attachEvent(‘onblur’, function() {
    checkUsername(5);
  });

}
```

## SUPPORTING OLDER VERSIONS OF IE

```
if (el.addEventListener) {

  el.addEventListener(‘blur’, function() {
    checkUsername(5);
  }, false);

} else {

  el.attachEvent(‘onblur’, function() {
    checkUsername(5);
  });

}
```

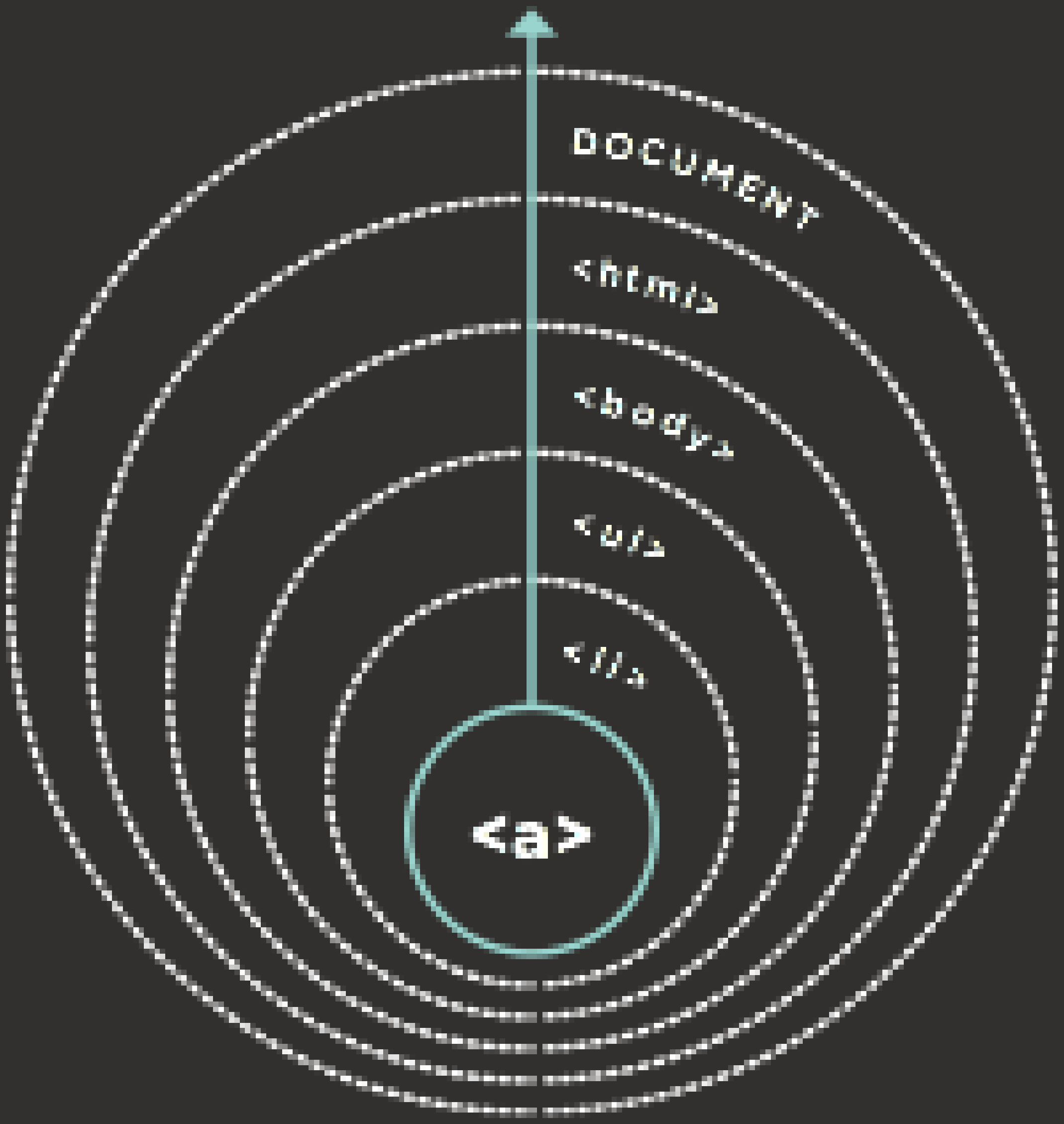
## Exercise

- On Moodle go to Javascript Events Lab 1 under the Labs section.

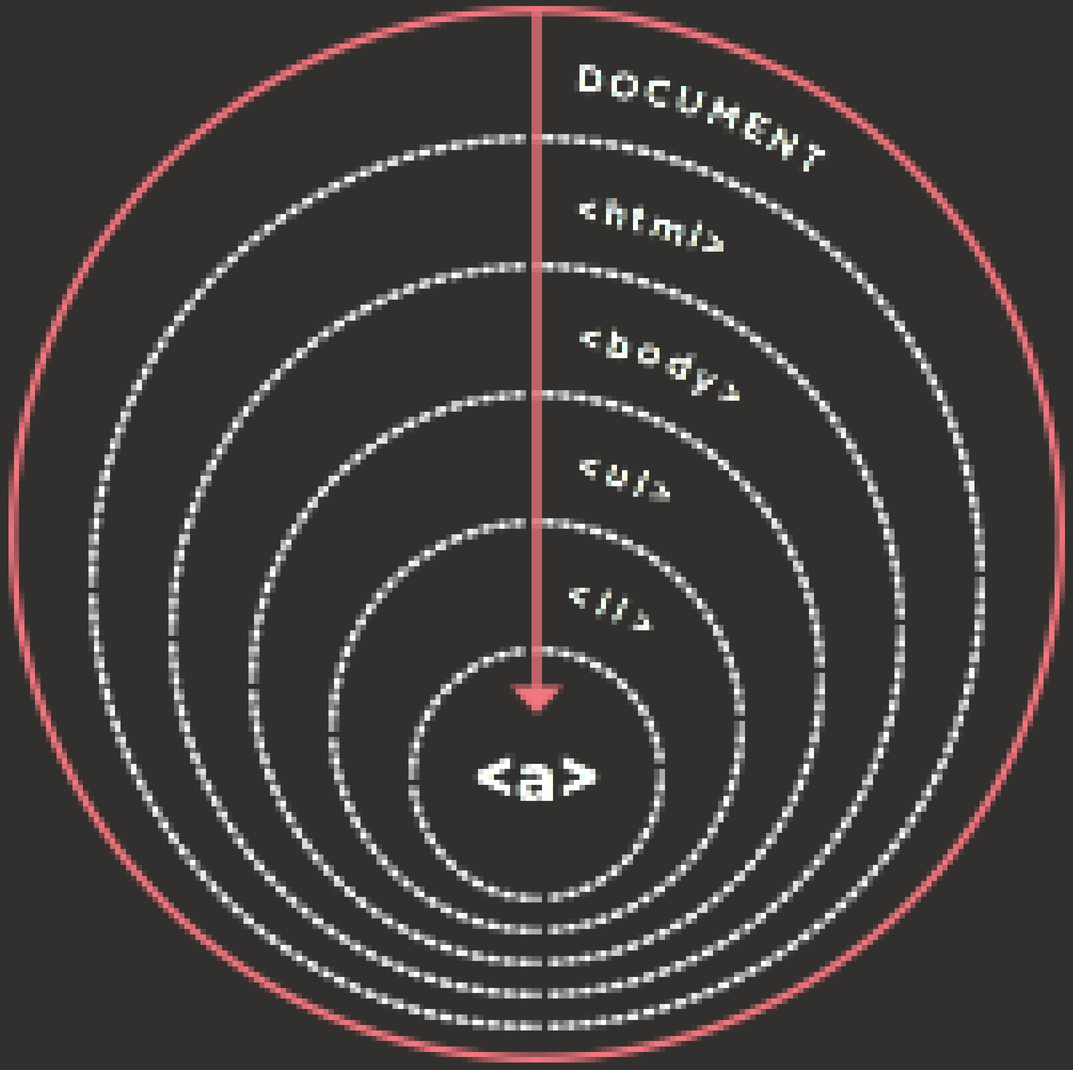
## EVENT FLOW

HTML elements nest inside other elements. If you hover or click on a link, you will also be hovering or clicking on its parent elements.

## EVENT BUBBLING



# EVENT CAPTURING



## Event Flow

- Event Bubbling: Flows outwards
- Event Capture: Flows inwards.
- Event flow matters only really when your code has event handlers on an element and one of its ancestors or descendent elements.

## Event Listener

- El.addEventListener(‘click’, myFunction(), false);
- See final argument. When set to false this directs event flow as bubble. When set to true this directs event flow as capture.

## Exercise

- On Moodle go to Javascript Events Lab 2 under the Labs section.

## THE EVENT OBJECT

When an event occurs,the `event` object can tell you information about it and which element it happened upon.

## EVENT OBJECT Properties

| Property | Purpose |
|---|---|
| target | The target of the event (most specific element interacted with). |
| type | Type of event that was fired. |
| cancelable | Whether you cancel the default behaviour of an element. For example, a buttons default behaviour is submit (type attribute defaults to submit). You can cancel this behaviour using this property. |

## EVENT OBJECT Methods

| Method | Purpose |
|---|---|
| preventDefault() | Cancel default behaviour of the event (if it can be cancelled). For example, clicking on a link, preventing it from following the specified url. |
| stopPropagation() | Stops the event from bubbling or capturing any further. |

## ELEMENT AN EVENT OCCURRED ON

### 1: EVENT LISTENER CALLS FUNCTION

```
function checkUsername(e) {
  var target = e.target;
}

var el = document.getElementById(‘username’);
el.addEventListener(‘blur’, checkUsername, false);
```

# ELEMENT AN EVENT OCCURRED ON

## 2: EVENT OBJECT PASSED TO FUNCTION

```
function checkUsername(e) {
  var target = e.target;
}

var el = document.getElementById(‘username’);
el.addEventListener(‘blur’, checkUsername, false);
```

## ELEMENT AN EVENT OCCURRED ON

3: ELEMENT THAT EVENT HAPPENED ON

```
function checkUsername(e) {
  var target = e.target;
}

var el = document.getElementById('username');
el.addEventListener('blur', checkUsername, false);
```

## ELEMENT AN EVENT OCCURRED ON

### 2: EVENT OBJECT PASSED TO FUNCTION WITH PARAMETER

```
function checkUsername(e, minLength) {
  var target = e.target;
}

var el = document.getElementById('username');
el.addEventListener('blur', function(e){
  checkUsername(e, 5);
}, false);
```

## Exercise

- On Moodle go to Javascript Events Lab 3 under the Labs section.
- Just download the zip file, unzip it, and review the html and js files syntax. Make sure to read the comments.
- Run some tests on the web page, to get a feel for the event object.
- Feel free to update the html, and js to run some other tests to ensure you understand the event object.

## EVENT DELEGATION

Creating event listeners for a lot of elements can slow down a page, but event flow allows you to listen for an event on a parent element.

# Event Delegation Benefits

- Works with new elements
  - If you add new elements to the DOM tree, you do not have to add event handlers to the new elements because the job has been delegated to an ancestor.
- Solves Limitation with *this* keyword
  - this Keyword not supported for events in some browsers. By using event delegation you don’t need to worry about that.
- Simplifies your code
  - It requires fewer functions to be written, and there are fewer ties between the DOM and your code, which helps maintainability

## Exercise

- On Moodle go to Javascript Events Lab 4 under the Labs section.