cphbusiness

COPENHAGEN BUSINESS ACADEMY

Event Handling and the DOM

Lars Mortensen

References/recommended reading:
DOM: https://www.w3schools.com/js/js_htmldom.asp
EVENTS: https://www.w3schools.com/js/js_htmldom_events.asp

JavaScript Functions

cphbusiness

Function declaration.

```
function square(number)
{
  return number * number;
}
```

Function expression.

```
var square = function(number) {
 return number * number;
}
```

Useful when passing a function as an argument to another function.

Named function expression.

```
var factorial = function fac(n) {
  return n<2 ? 1 : n*fac(n-1)
};
Call: console.log(factorial(4));
```

Name is only visible to the function itself and in Exception Stack Traces – Prefer this over Anonymous functions

Registering HTML DOM Events

```
function clickMe(){
  alert("I was clicked");
};
```

In the HTML:

```
<button id="b1" onclick="clickMe()">Click me</button>
```

The W3C standard event-registration model:

```
var button = document.getElementById("b1");
button.addEventListener("click",clickMe, false);
```

Event **EventHandler** **Bubling/tunneling**

Same as above but via an anonymous eventhandler:

```
var button = document.getElementById(”b1");
button.addEventListener("click", function() {
        alert("I was clicked");
});
```

HTML DOM Events

Obviously there are many other events than **click**

MDN Event Reference:

https://developer.mozilla.org/en-US/docs/Web/Events

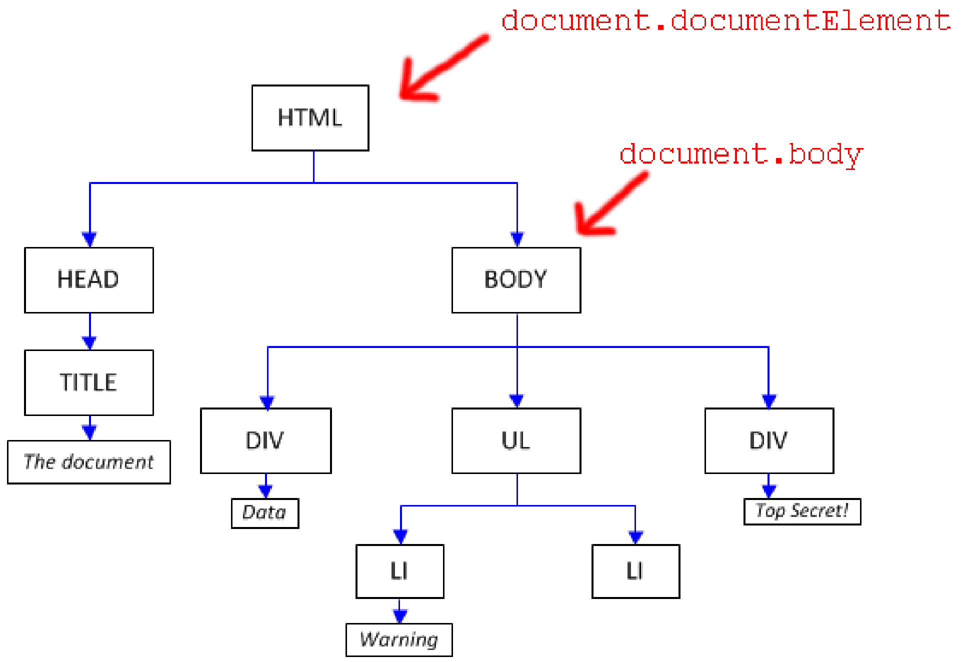
W3schools, list of events:

https://www.w3schools.com/jsref/dom_obj_event.asp

The Document Object Model (DOM)

cphbusiness

- The Document Object Model is an API for HTML and XML documents.
- It provides a structural representation of the document, enabling you to modify its content and visual presentation.
- Essentially, it connects web pages to scripts or programming languages.



https://developer.mozilla.org/en-US/docs/DOM/About_the_Document_Object_Model

JavaScript and the DOM

- JavaScript understands HTML and can directly access it.
- JavaScript uses the HTML Document Object Model to manipulate HTML.
- The DOM is a hierarchy of HTML things.
- Use the DOM to build an “address” to refer to HTML elements in a web page.
- Levels of the DOM are dot-separated in the syntax.

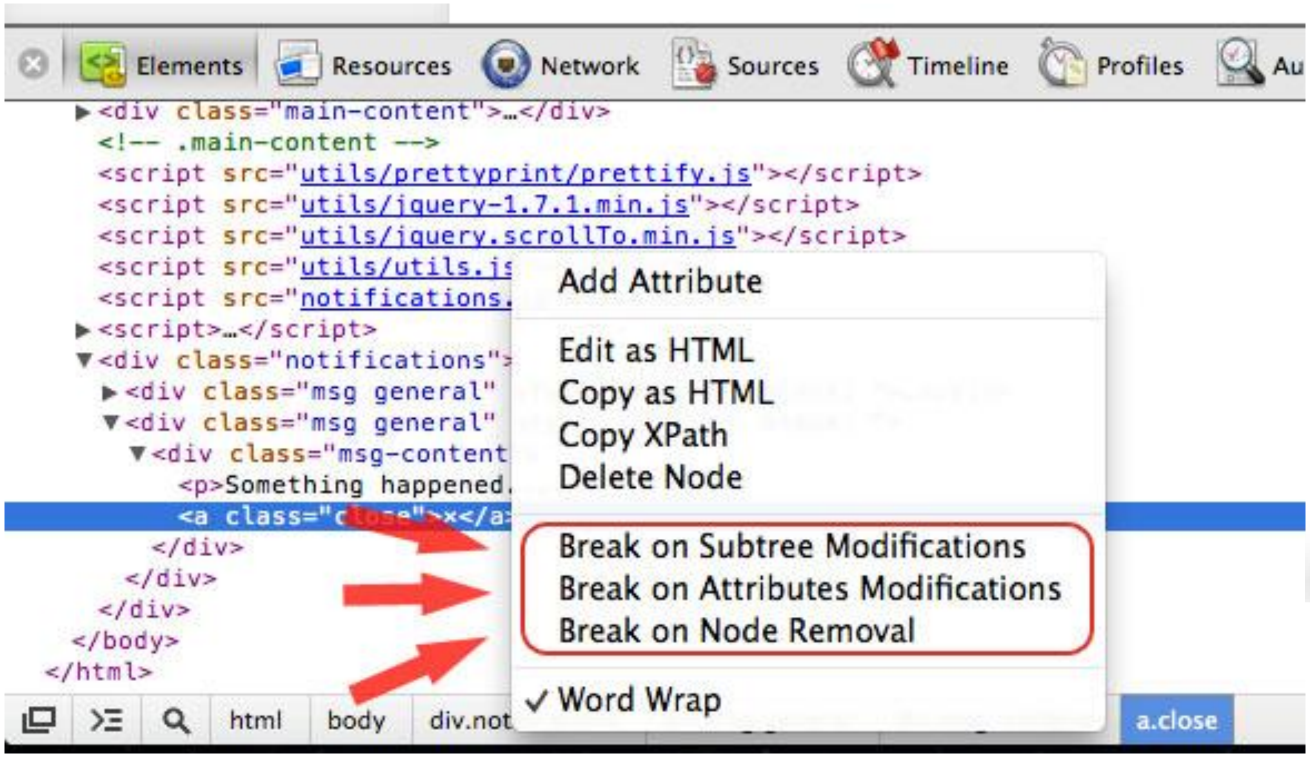
Manipulating the DOM

Manipulating the DOM can be done with plain JavaScript or using JavaScript libraries like JQuery, Angular React and many others.

In this module we will use plain JavaScript because you NEED to learn JavaScript.

In the next module we will use the REACT library.

Inspecting the DOM



```
▶<div class="main-content">…</div>
 <!-- .main-content -->
 <script src="utils/prettyprint/prettify.js"></script>
 <script src="utils/jquery-1.7.1.min.js"></script>
 <script src="utils/jquery.scrollTo.min.js"></script>
 <script src="utils/utils.js
 <script src="notifications.
▶<script>…</script>
▼<div class="notifications">
  ▶<div class="msg general"
  ▼<div class="msg general"
    ▼<div class="msg-content
       <p>Something happened.
       <a class="close">×</a>
     </div>
   </div>
 </div>
</body>
</html>
```

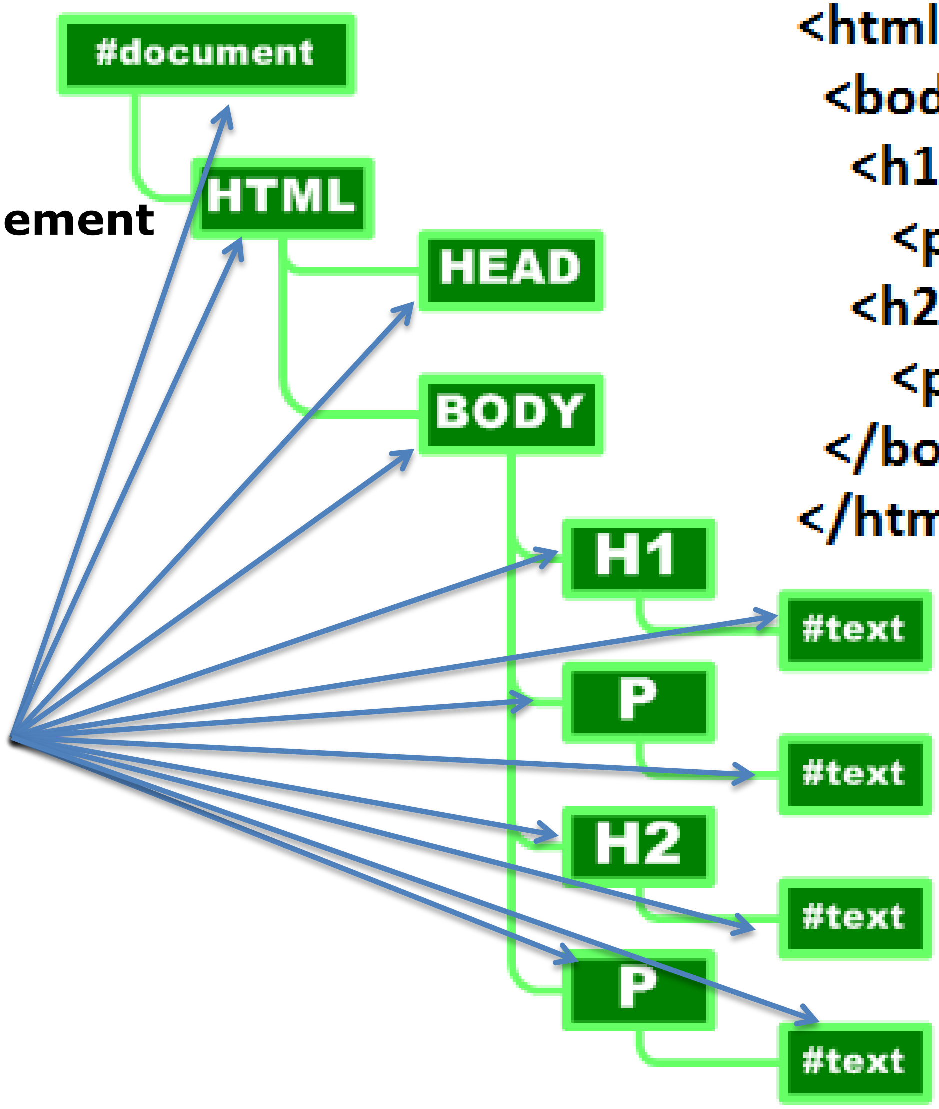
Document Tree Structure

document

document.documentElement

document.body

Dom nodes



```
<html>
  <body>
    <h1>Heading 1</h1>
      <p>Some text</p>
    <h2>Heading 1</h2>
      <p>Some text</p>
  </body>
</html>
```

DOM nodes

In a DOM tree, almost everything you'll come across is a node.

- Every element is at its most basic level a node in the DOM tree.
- Every attribute is a node.
- Every piece of text is a node.
- Comments
- Special characters (like &copy; a copyright symbol)
- DOCTYPE declaration

All are nodes

The document node

The *document node* is actually not an element in an HTML (or XML) page, but the page itself.
So in an HTML Web page, the document node is the **entire DOM tree**

Document methods for manipulating the DOM

```
document.getElementById(..);
document.getElementsByName(..)
document.getElementsByTagName(..)
document.querySelector();
```

Can be used for complex queries. If a query returns more than one element, only the first is returned

https://www.w3schools.com/jsref/met_document_queryselector.asp

Useful DOM methods

The Style Object

```
var style= document.getElementById("d1").style;

style.color = "red";
style.width = "100px";
style.backgroundColor = "yellow";
```

<u>A small oddity</u>

HTML is not case sensitive and uses dashes for readability.

CSS properties are converted to JavaScript by making them camelCase without any dashes.

```
#d1 {
    background-color: yellow;
}
```

Useful DOM methods

innerHTML

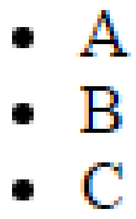
```
<div id="d1"></div>
```

The `Element.innerHTML` property *sets* or *gets* the HTML syntax describing the element's descendants.

Use it, to dynamically change the content of the DOM

```
var e = document.getElementById("d1");
e.innerHTML =
  "<ul><li>A</li><li>B</li><li>C</li></ul>"
```

$\longrightarrow$

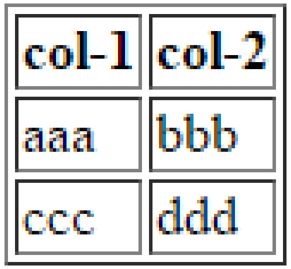


- A
- B
- C

Or (line-breaks inserted for readability)

```
e.innerHTML = "<table border='1'>
   <tr><th>col-1</th><th>col-2</th></tr>
   <tr><td>aaa</td><td>bbb</td></tr><tr>
   <td>ccc</td><td>ddd</td></tr>
   </table>";
```

$\longrightarrow$

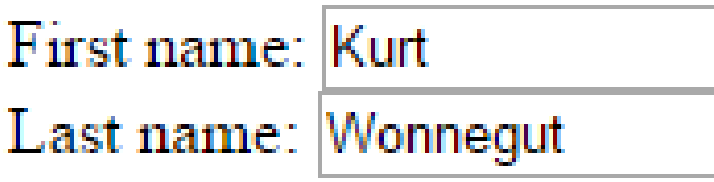


col-1	col-2
aaa	bbb
ccc	ddd

Useful DOM methods

Set get values from input fields

```
<form>
  First name: <input type="text" id="fname"> </br>
  Last name: <input type="text" id="lname">
</form>
```


First name: Kurt
Last name: Wonnegut

We can use the Elements ***value*** property to *get* or *set* values

```
document.getElementById("fname").value="Kurt";
document.getElementById("lname").value="Wonnegut";
```

Because the input fields is in a form, they can be accessed via the forms Element:

```
var form = document.getElementsByTagName("form")[0];
console.log(form.fname.value);
console.log(form.lname.value);
```

Useful DOM methods

What else can you do

Set or *get* attributes

```
<a id="fck" href="????">Where should I go? </a>
```

JavaScript:

```
var link = document.getElementById("fck");
link.setAttribute("href","http://fck.dk");
```

There is (almost) no limit to the "things" you can do with a DOM element.

https://developer.mozilla.org/en-US/docs/Web/API/Element

Event Handlers - continued

Event handlers and ***this***

When the event handler is invoked, the *this* keyword inside the handler is set to the DOM element on which the handler is registered.

The Event handlers *event argument*:

The handler takes an optional **event** parameter.

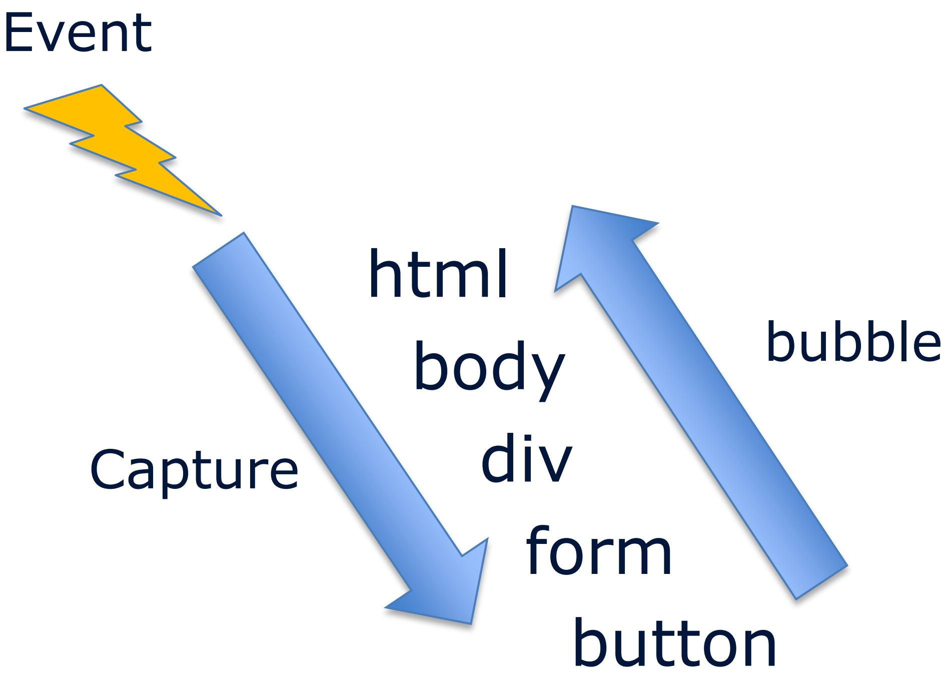
Returns the element that triggered the event

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

When will these two be different?

Event Phases

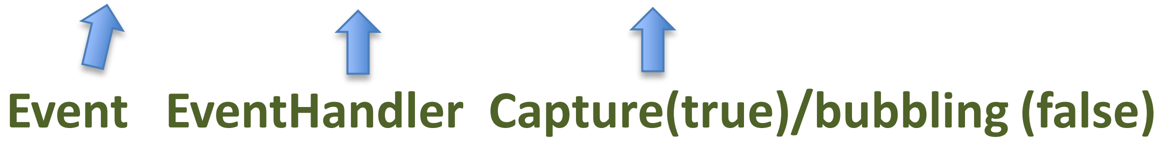
```
<html>
  <body id="body">
    <div id="div">
      <form id="form">
        <input id= "btn"
          type="button" value="test">
      </form>
    </div>
  </body>
</html>
```



When we click the button, the click event actually "drills down" the document through the button's parents to reach it, and then makes its way back up. This initial "drilling down" is called the "Capture Phase," and when it circles around and goes back up to the document element, it is known as the "Bubble Phase."

This should explain the third argument in the addEventlListener method

```
button.addEventListener("click",clickMe, false);
```



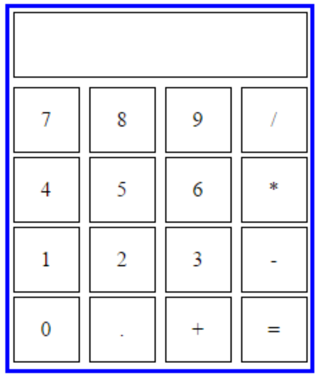
Event Phases - Why should I care?

Suppose you have x DOM Elements (Buttons, anchors or what ever)

We could attach x sets of event handlers to those objects.

Or we could attach one single event handler to the container of the x objects

This weeks exercises will demonstrate this:



Cancel Bubbling

cphbusiness

Cancel bubbling to keep the parent nodes from seeing the event.

```
function myHandler(evt){
  e.stopPropagation();
  …
}
```

Exercises will provide an example

Prevent Default Action

An event handler can prevent a browser action associated with the event

```
function myHandler(evt){
   e.preventDefault();
   …
}
```

Examples:

- We could prevent a form from submitting data (when would that make sense?)
- We could prevent a link from navigation "out"

Exercises will provide an example