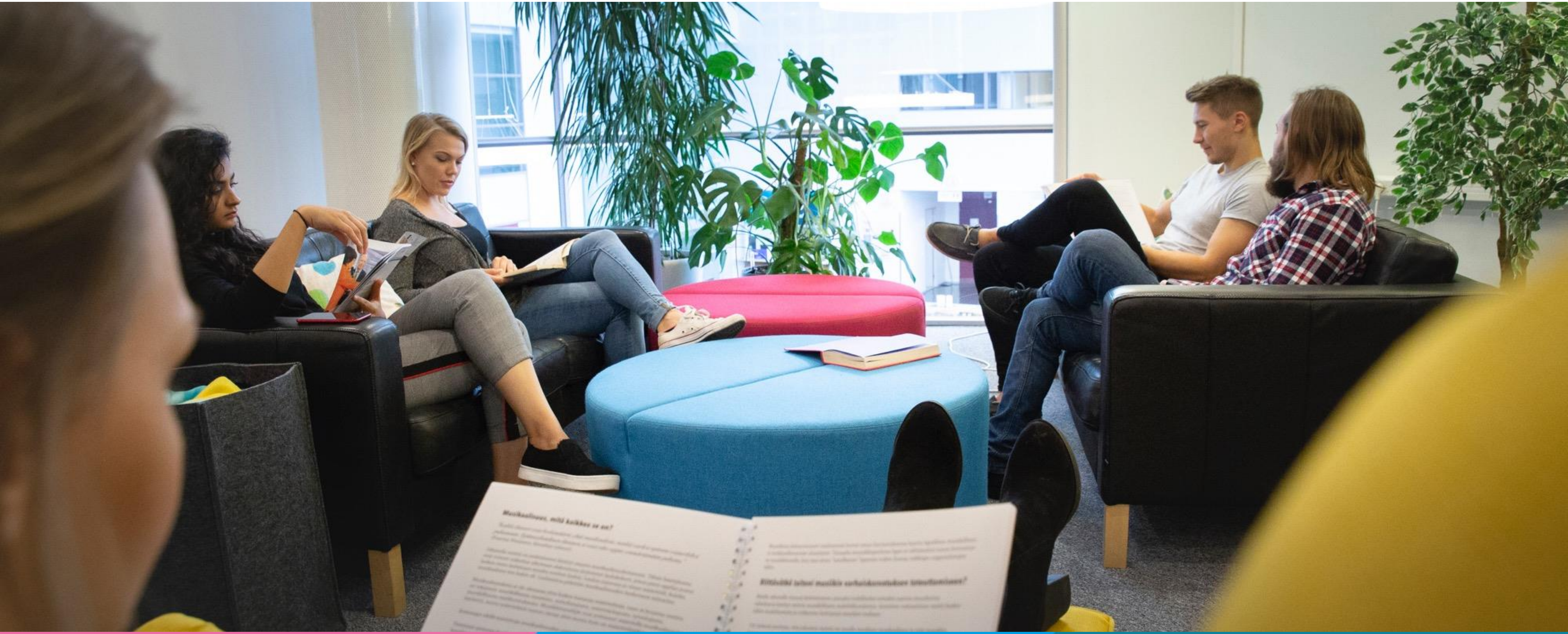
Laurea-ammattikorkeakoulu



LAU REA Digital Living Lab
Turvallisuus
LAU REA Digital Living Lab
LAU REA Digital Living Lab

JavaScript Events

Web-sovellusten kehittäminen Javascriptillä
TO00BL10

Jari Kovalainen

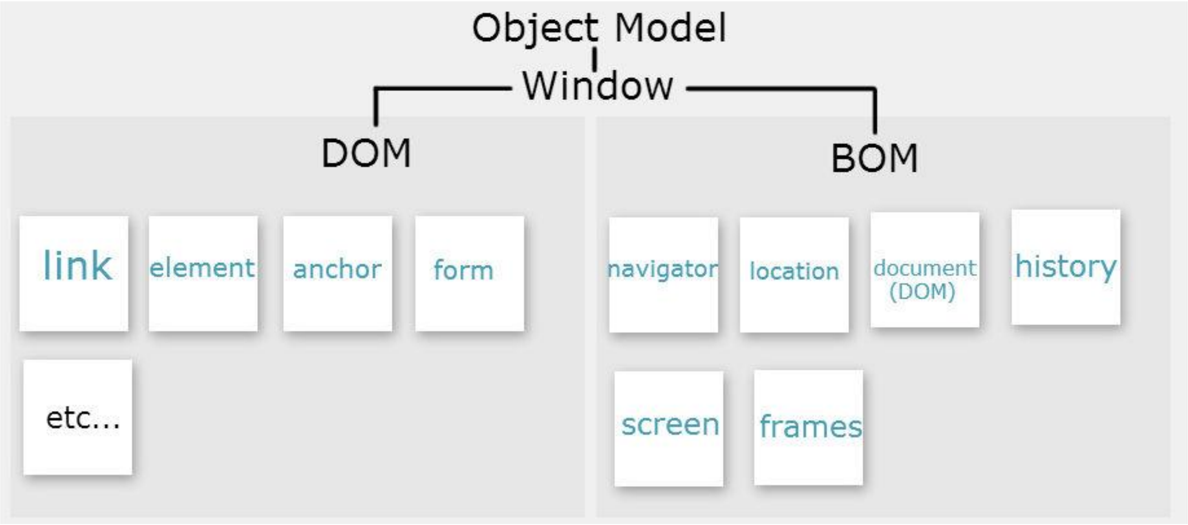
Laurea Tikkurila

What will be discussed

- Talking to the browser – The BOM
- Functions
- JavaScript Events

1. The BOM– Browser Object Model

- Since modern browsers have implemented (almost) the same methods and properties for JavaScript interactivity, it is often referred to, as methods and properties of the BOM.

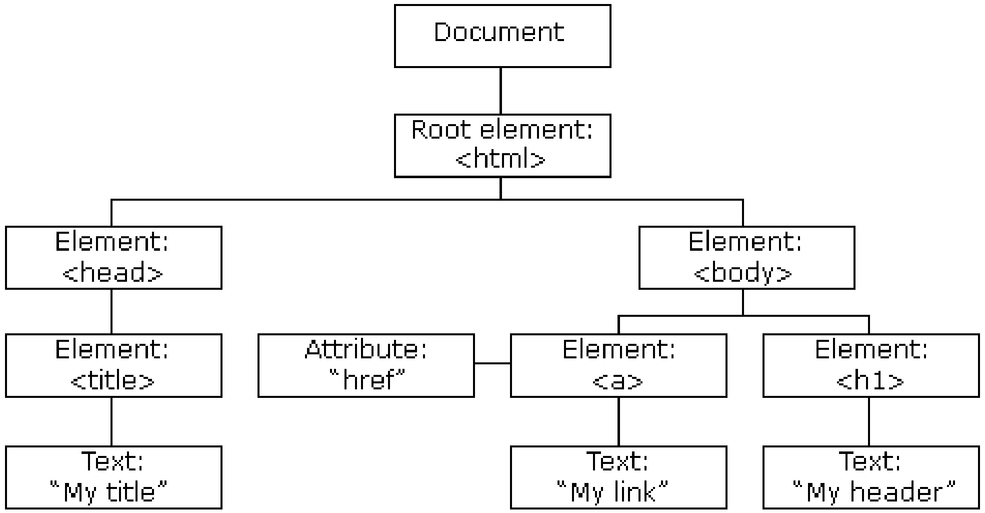


1. Using The BOM

- Allows JS to talk to the browser and get information about:
  - Browser Window contents (DOM)
  - Frames shown in page
  - Screen size, orientation, color depth
  - Navigator: browser specific information
  - History: web site history
  - Location: current web page information

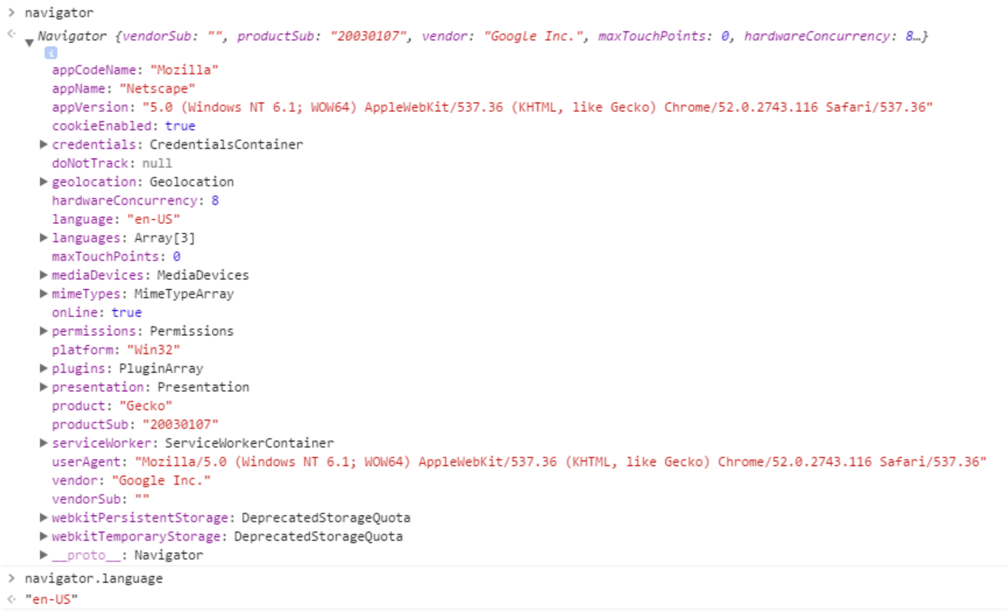
2. THE DOM – Document Object Model

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**:



BOM: Navigator object

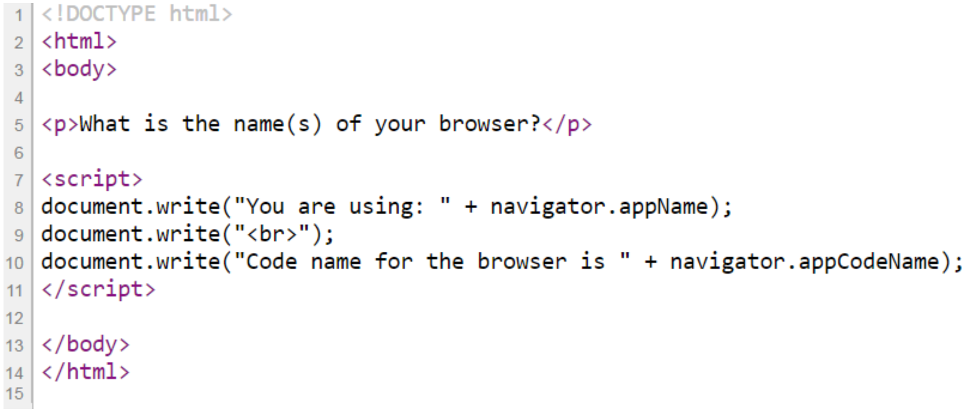
- For example, we can access information about the browser



```
> navigator
<  ▼ Navigator {vendorSub: "", productSub: "20030107", vendor: "Google Inc.", maxTouchPoints: 0, hardwareConcurrency: 8…}
      appCodeName: "Mozilla"
      appName: "Netscape"
      appVersion: "5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36"
      cookieEnabled: true
    ▶ credentials: CredentialsContainer
      doNotTrack: null
    ▶ geolocation: Geolocation
      hardwareConcurrency: 8
      language: "en-US"
    ▶ languages: Array[3]
      maxTouchPoints: 0
    ▶ mediaDevices: MediaDevices
    ▶ mimeTypes: MimeTypeArray
      onLine: true
    ▶ permissions: Permissions
      platform: "Win32"
    ▶ plugins: PluginArray
    ▶ presentation: Presentation
      product: "Gecko"
      productSub: "20030107"
    ▶ serviceWorker: ServiceWorkerContainer
      userAgent: "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36"
      vendor: "Google Inc."
      vendorSub: ""
    ▶ webkitPersistentStorage: DeprecatedStorageQuota
    ▶ webkitTemporaryStorage: DeprecatedStorageQuota
    ▶ __proto__: Navigator
> navigator.language
< "en-US"
```

BOM: Navigator object

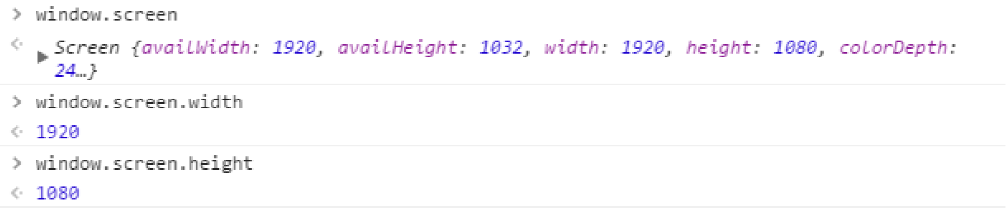
- We can add this to the <script>-tag in the HTML page.
- Code is run when the page is loaded



```
<!DOCTYPE html>
<html>
<body>

<p>What is the name(s) of your browser?</p>

<script>
document.write("You are using: " + navigator.appName);
document.write("<br>");
document.write("Code name for the browser is " + navigator.appCodeName);
</script>

</body>
</html>
```

BOM: Window object

- Window-object lets us query the information about screen properties, such as width and height



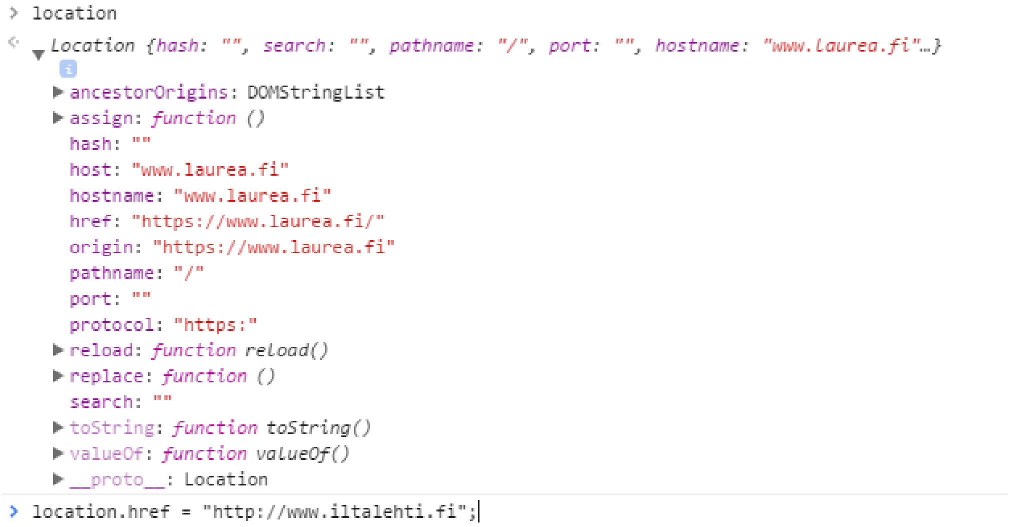
```
> window.screen
< ▶ Screen {availWidth: 1920, availHeight: 1032, width: 1920, height: 1080, colorDepth:
    24…}
> window.screen.width
< 1920
> window.screen.height
< 1080
```

BOM: History object

- History-object lets us query the information about browser history
- We can also control the browser by telling it to go back or forward in history
- NOTE: History is protected by the browser; Javascript is not allowed to read the contents of it.
  - <script>
  -   history.back();
  - </script>

BOM: Location object

- Location-object lets us query the information about current location
- We can also set the location which causes the browser to load it



```
> location
< ▼Location {hash: "", search: "", pathname: "/", port: "", hostname: "www.laurea.fi"…}
    ▶ancestorOrigins: DOMStringList
    ▶assign: function ()
     hash: ""
     host: "www.laurea.fi"
     hostname: "www.laurea.fi"
     href: "https://www.laurea.fi/"
     origin: "https://www.laurea.fi"
     pathname: "/"
     port: ""
     protocol: "https:"
    ▶reload: function reload()
    ▶replace: function ()
     search: ""
    ▶toString: function toString()
    ▶valueOf: function valueOf()
    ▶__proto__: Location
> location.href = "http://www.iltalehti.fi";
```

JavaScript Events

Introduction

- So far we’ve learned that the browser will run any JavaScript code whenever it encounters one while loading the page
- With an exception: code within functions will only be run when the function gets called
- This Chapter introduces a way to call those functions, other than from a block of code itself: events

HTML Events

- HTML events are **"things"** that happen to a page or its elements
- These can be something like:
  - a web page has finished loading
  - an input field was changed
  - a button was clicked
  - a form was submitted
- JavaScript can **"react"** (execute code) on these events
- Common tasks are checking or validating the input

Handling Events in JS

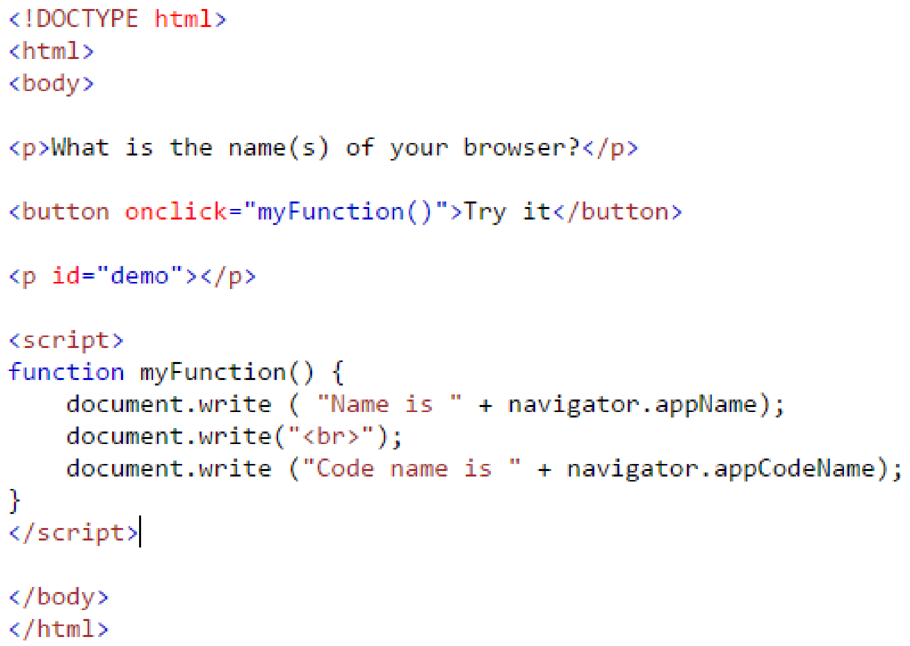
- HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements

- Some examples could be:

```
<button onclick="alert('Click')">Click me</button>

<input onfocus="myFunction()"></input>

<form onsubmit="validateForm()"></form>

<button onmouseover="alert('On me!')"   onmouseout="alert('Off
me')">Nada</button>
```

Example

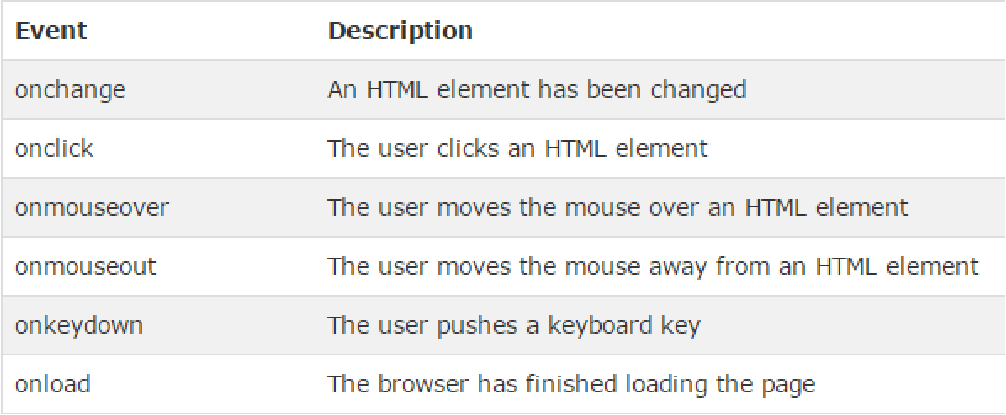
- While we can write the code directly into the event, it is usually easier to call for a named function
- Then we declare the programming code later on as a function



```
<!DOCTYPE html>
<html>
<body>

<p>What is the name(s) of your browser?</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
    document.write ( "Name is " + navigator.appName);
    document.write("<br>");
    document.write ("Code name is " + navigator.appCodeName);
}
</script>

</body>
</html>
```

Common Events in JS

- Common events are listed below
- Full list of events can be found online



Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

Adding listeners dynamically

- In some cases, one might want to add event listeners dynamically through JavaScript
- This can be done using **addEventListener**-method
- Removing the listener is done using **removeEventListener** –method
- Why:
  - keeps the UI and logic on a separate files and leaves HTML files clean from JavaScript
  - Separation of Concerns

Adding listeners dynamically

- // Get reference to an element
- var element = document.getElementsByTagName(‘h1’)[0];
 // Add
- *element*.addEventListener("click", function(){ alert("Hello World!"); });

- // Add
- *element*.addEventListener(”mouseover", function myFunction(){   alert("Hello World!");
- });

- // Remove – works only on NON ANONYMOUS FUNCTIONS
- *element*.removeEventListener("mousemove", myFunction);

Adding listeners dynamically

- <html>
-
-   <body>
-     <h1>Eka nappi</h1>
-     <p id="info">Lorem ipsum dolorem</p>
-     <button id="button1">Lisää kuuntelija</button>
-     <button id="button2">Poista kuuntelija</button>
- **// Siirrä JS-koodi viimeiseksi ennen </body> tägiä**
  // Kaikki JS koodi, myös kuuntelijat tiedostoon
- **<script src="koodit.js"></script>**
-   </body>
- </html>

Adding listeners dynamically

- **// Etsitään viite ja lisätään kuuntelija + funktio**
- var x = document.getElementsByTagName('h1')[0];
- x.addEventListener("click", function(){ alert("You Clicked Me!"); });

  - **// Etsitään viite ja lisätään kuuntelija + funktio**
    - var p = document.getElementById('info');
  - p.addEventListener("mouseover", function myFunction(){ console.log("You Clicked Me!"); });

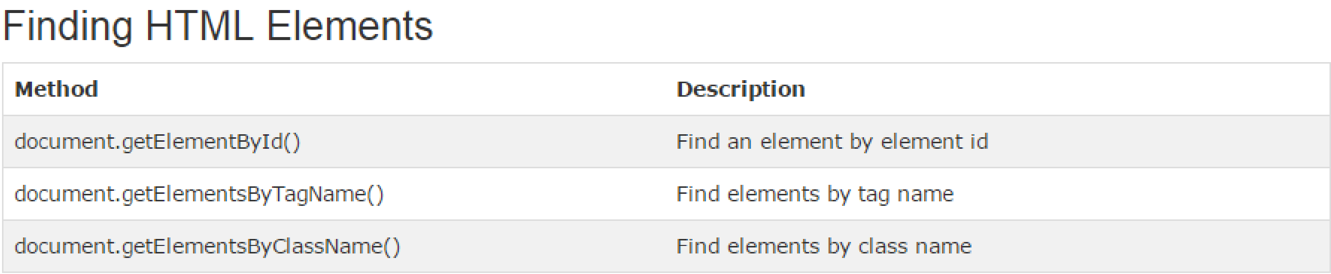
- var b1 = document.getElementById('button1');
- b1.addEventListener("click", function(){
-   console.log("Button clicked");
-   p.addEventListener("mouseover", function myFunction(){ alert("You hovered on Me!"); });
- });

  - **// Etsitään viite ja poistetaan kuuntelija**
  - var b2 = document.getElementById('button2');
  - b2.addEventListener("click", function(){
  -   console.log("Button clicked");
  -   p.removeEventListener("mouseover", myFunction, true);
  - });

- // https://codepen.io/mjstenbe/pen/ExYJpzO

Finding the elements using DOM

- DOM Scripting is about finding an element and changing its attributes



Finding HTML Elements

Method	Description
document.getElementById()	Find an element by element id
document.getElementsByTagName()	Find elements by tag name
document.getElementsByClassName()	Find elements by class name

- For example:
  var myElement = document.getElementById(‘main-title’);

Finding the elements using DOM

- Search within **the entire document** (HTML-page)

- **document**.getElementById('main-navi');

- // Placing the resultselt in a variable for later use
 var myResult = document.getElementById('main-navi');

- Search within a **previous resultset** – not the entire page

- // Search only within the myResult -variable
- myResult.getElementsByTagName('li');