JavaScript och DOM

Det här har vi gjort förut

Johan Holmberg 2020-04-01

MALMÖ UNIVERSITET

Dagens föreläsning

- Förra veckans bravader
- DOM
- Händelser i JavaScript

MALMÖ UNIVERSITET

Förra veckan

- Vi tittade på vad JavaScript är och dess bakgrund
- Vi tittade på syntaxen i JavaScript
- Vi lärde oss att arbeta med variabler, operationer och flödeskontroll (if, switch, loopar)
- Vi tittade på funktioner

Språket Javascript

- Ursprungligen avsett för att göra webbsidor roligare
  - Bra stöd för att manipulera DOM
  - Bra stöd i de flesta webbläsare
- Körs numera lite över allt
- Har numera ett bra stöd av tredjepartsbibliotek

Språket Javascript

- Syntaxen liknar C, Java, etc
- Löst typat – inga explicita variabeltyper
- Multiparadigm:
  - Händelsedrivet
  - Funktionellt
  - Prototypbaserat (ungefär som objektorienterat)
  - Imperativt

MALMÖ UNIVERSITET

DOM

MALMÖ UNIVERSITET

VanillaJS – JavaScript utan ramverk

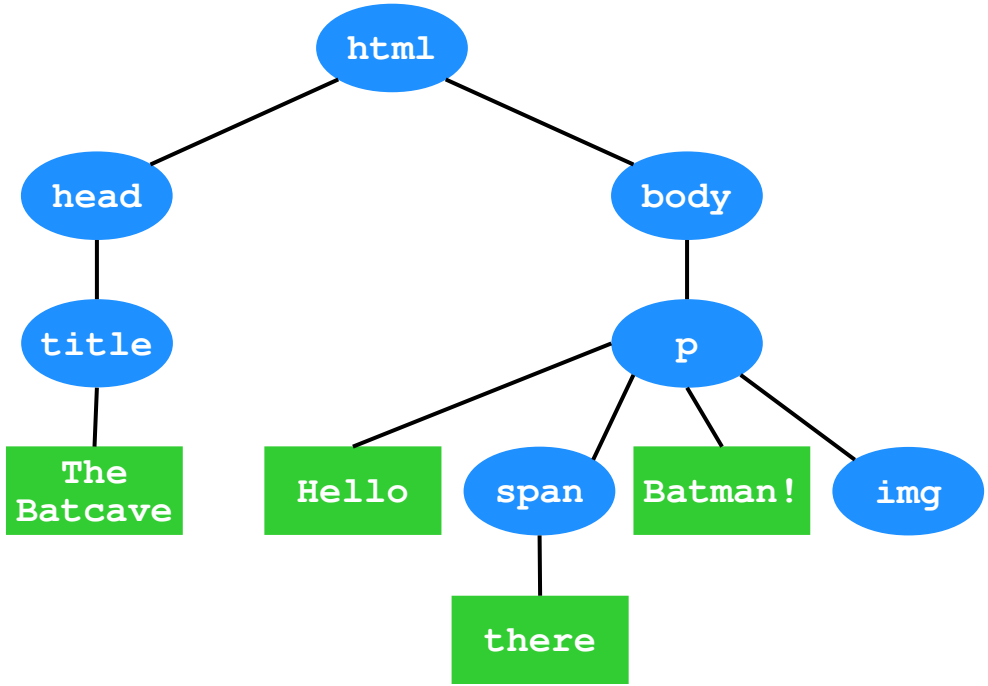
Vad är DOM?

HTML DOM (Document Object Model):

- En representation av HTML-dokumentet som ett träd av element – noder.
- HTML-dokumentet kan ses som ett stort objekt

Vad är DOM?

```
<!DOCTYPE html>
<html>
<head>
  <title>The Batcave</title>
</head>

<body>
  <p>
    Hello <span>there</span> Batman!
    <img src="batman_logo.png">
  </p>
</body>

</html>
```



MALMÖ UNIVERSITET

Kom åt element/noder

Det finns fyra sätt att komma åt element i DOM-trädet:

- `document.getElementById(id)` → ett element
- `document.getElementByTagName(tagName)` → en samling av element
- `document.getElementByClassName(className)` → en samling av element
- `document.querySelector(query)` → ett element

MALMÖ UNIVERSITET

Hitta alla element av en specifik typ

```
var elements = document.getElementByTagName("p");
```

Hittar alla p-element och returnerar ett objekt av typen `HTMLCollection`, som liknar ett array-objekt:

```
var el = elements[0]; // Första elementet i samlingen

var size = elements.length; // Antal element
```

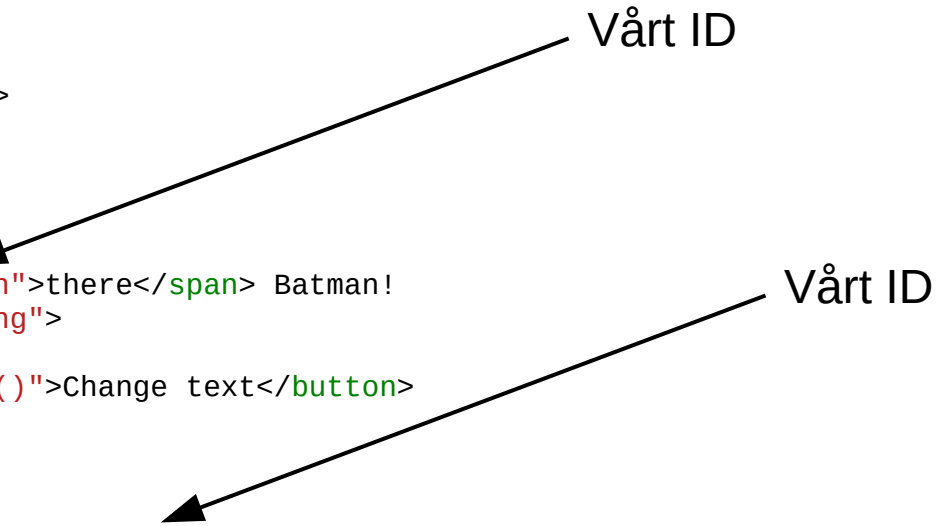
MALMÖ UNIVERSITET

ID kontra klass i HTML

- Enskilda HTML-element (kan) identifieras med ett *ID*:
  - Ett ID består av en sträng
  - ID:n är unika och kan bara finnas en gång per dokument
  - Refereras till som `#mitt_id`
- Ett element kan tillhöra en *klass*:
  - Ett klassnamn består av en sträng
  - Flera element kan tillhöra samma klass (och ett element kan tillhöra flera klasser).
  - Refereras till som `.min_klass`

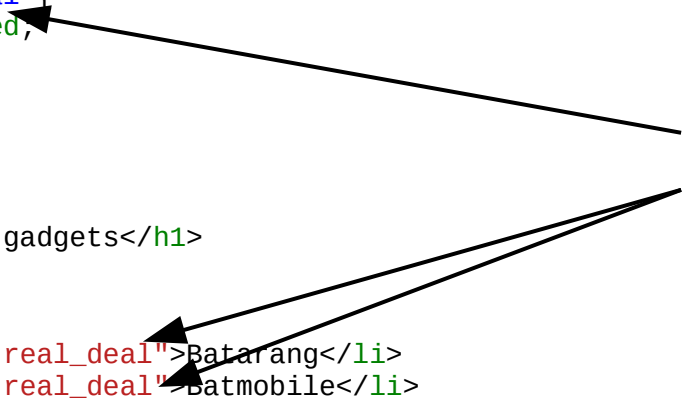
Exempel på användande av ID

```
<!DOCTYPE html>
<html>
<head>
  <title>Sample Site</title>
</head>


<body>
  <p>
    Hello <span id="location">there</span> Batman!
    <img src="batman_logo.png">
  </p>
  <button onclick="relocate()">Change text</button>


  <script>
    function relocate() {
      document.getElementById("location").innerHTML = "here";
    }
  </script>
</body>


</html>
```



Exempel på användande av klass

```
<!DOCTYPE html>
<html>
<head>
  <title>Sample Site</title>
  <style>
    li.real_deal {
      color: red;
    }
  </style>
</head>


<body>
  <h1>Batman's gadgets</h1>


  <ul>
    <li class="real_deal">Batarang</li>
    <li class="real_deal">Batmobile</li>
    <li class="bad_pun">Batteries</li>
  </ul>
</body>


</html>
```



Den här regeln
matchar de här elementen

MALMÖ UNIVERSITET

Exempel på användande av båda

```
<!DOCTYPE html>
<html>
<head>
  <title>Sample Site</title>
</head>

<body>
  <h1>Batman's gadgets</h1>

  <ul id="gadget_list">
    <li class="real_deal">Batarang</li>
    <li class="real_deal">Batmobile</li>
    <li class="bad_pun">Batteries</li>
  </ul>
</body>

</html>
```

MALMÖ UNIVERSITET

Query Selector

```
var element = document.querySelector(".someClass");
```

Returnerar den första noden av klassen `someClass`.

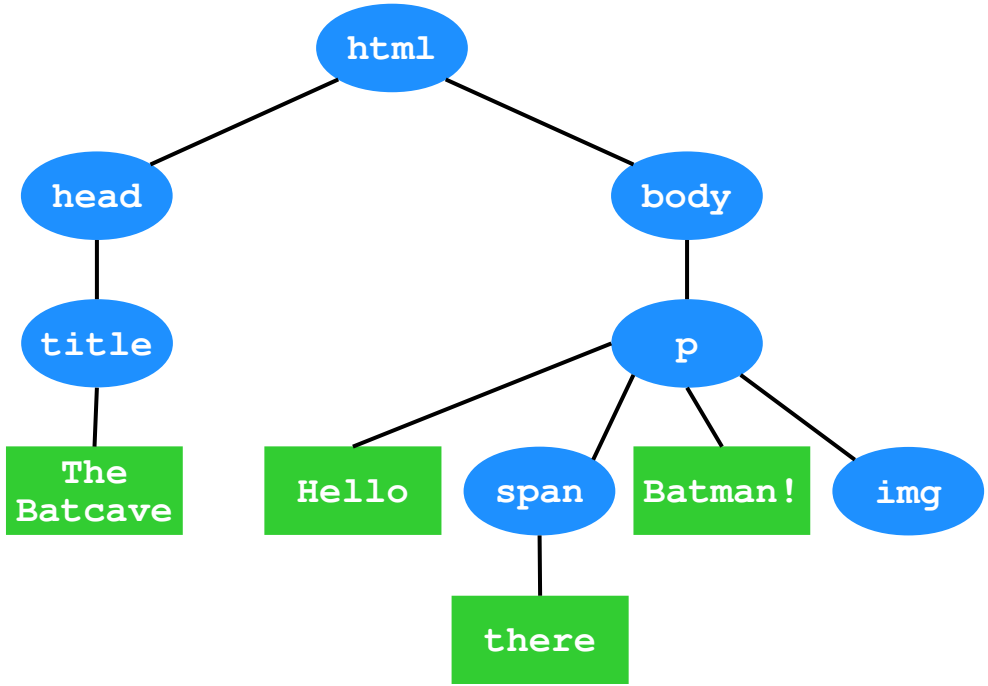
Kan användas med:

- Taggnamn: `document.querySelector("tag");`
- Klassnamn: `document.querySelector(".someClass");`
- ID: `document.querySelector("#someID");`

MALMÖ UNIVERSITET

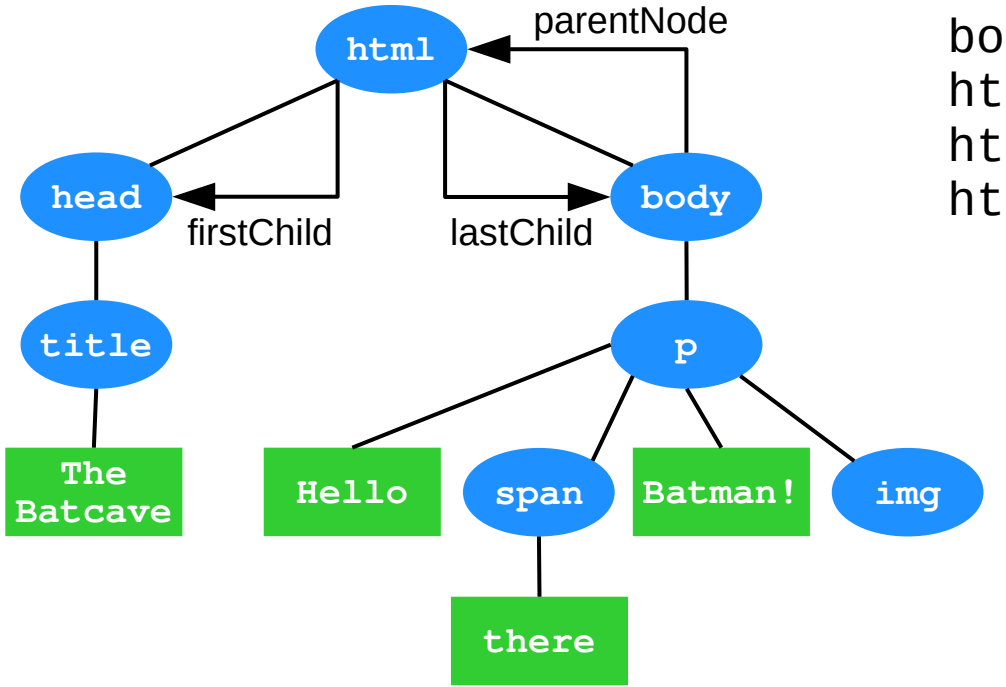
Navigera i DOM

```
<!DOCTYPE html>
<html>
<head>
  <title>The Batcave</title>
</head>

<body>
  <p>
    Hello <span>there</span> Batman!
    <img src="batman_logo.png">
  </p>
</body>

</html>
```



MALMÖ UNIVERSITET

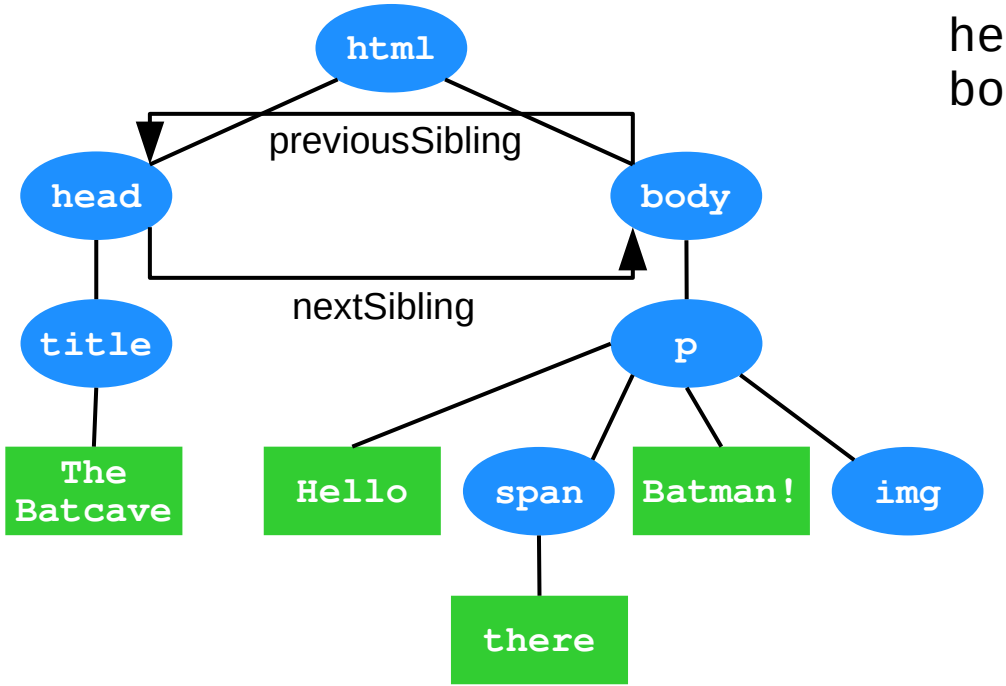
Navigera i DOM – föräldrar och barn




```
body.parentNode → html
html.firstChild → head
html.lastChild → body
html.childNodes[1] → body
```

MALMÖ UNIVERSITET

Navigera i DOM – syskon




```
head.nextSibling → body
body.previousSibling → head
```

MALMÖ UNIVERSITET

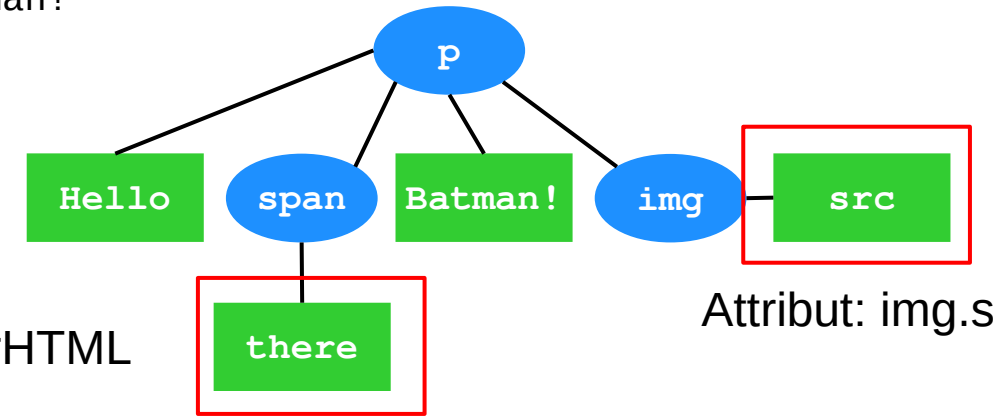
Modifiera en nod

Vi behöver känna till två typer av egenskaper hos en nod:

- InnerHTML
  - Den text som “innesluts” av ett HTML-element
  - *Kan* vara HTML-kod
- Attribut
  - Förändrar ett elements beteende. Exempel:
    - src
    - style

Vad är vad?

```
...
<p>
  Hello <span>there</span> Batman!
  <img src="batman_logo.png">
</p>
...
```



span.innerHTML

Attribut: img.src

MALMÖ UNIVERSITET

Arbeta med innerHTML

```
<!DOCTYPE html>
<html>
<head>
  <title>Sample Site</title>
</head>

<body>
  <h1>Welcome to the Batcave!</h1>

  <p>
    Alfred is <span id="status">out</span>.
  </p>

  <script>
    var status = document.getElementById("status");
    status.innerHTML = "in";
  </script>
</body>

</html>
```

MALMÖ UNIVERSITET

Arbeta med attribut

```
<!DOCTYPE html>
<html>
<head>
  <title>Sample Site</title>
</head>

<body>
  <h1>Welcome to Arkham Asylum!</h1>

  <h2>Inmate of the month:</h2>
  <img id="inmate" src="penguin.jpg">

  <script>
    var imnate = document.getElementById("inmate");
    inmate.setAttribute("src", "joker.jpg");
  </script>
</body>

</html>
```

MALMÖ UNIVERSITET

Special: attribut-noder och CSS

- Attributet `style` (inline CSS) är en attribut-nod och hanteras så här:

```
...
<body>
  <h1>Welcome to Wayne Mansion!</h1>

  <p>
    Alfred is <span id="status" style="color: red">out</span>.
  </p>

  <script>
    var element = document.getElementById("status");
    element.innerHTML = "in";
    element.style.color = "green";
  </script>
</body>
...
```

MALMÖ UNIVERSITET

Lägg till nya element till trädet

Vi skapar nya element genom att använda oss av

```
document.createElement("tag_name")
```

Nya element behöver ofta text. Det skapas så här:

```
document.createTextNode("My text")
```

Dessa sammanfogas sedan:

```
var p = document.createElement("p");
var textNode = document.createTextNode("Batman");
p.appendChild(textNode);
```

MALMÖ UNIVERSITET

Lägg in nya element till trädet

Vi har två sätt att lägga till nya element till trädet

- `element.appendChild(node)`

  lägg `node` sist bland barnen till `element`

- `element.insertBefore(node, child)`

  lägg till före elementet `child`

MALMÖ UNIVERSITET

Exempel på appendChild()

```
<!DOCTYPE html>
<html>
...

<body>
  <h1>Batman's gadgets</h1>

  <ul id="gadget_list">
    <li id="batarang" class="real_deal">Batarang</li>
    <li id="batmobile" class="real_deal">Batmobile</li>
    <li id="batteries" class="bad_pun">Batteries</li>
  </ul>

  <script>
    var batcave = document.createElement("li");
    var label = document.createTextNode("Bat cave");
    batcave.appendChild(label);

    var parent = document.getElementById("gadget_list");
    parent.appendChild(batcave);
  </script>
</body>

</html>
```

MALMÖ UNIVERSITET

Exempel på insertBefore()

```
<!DOCTYPE html>
<html>
…

<body>
  <h1>Batman's gadgets</h1>

  <ul id="gadget_list">
    <li id="batarang" class="real_deal">Batarang</li>
    <li id="batmobile" class="real_deal">Batmobile</li>
    <li id="batteries" class="bad_pun">Batteries</li>
  </ul>

  <script>
    var batcave = document.createElement("li");
    var label = document.createTextNode("Bat cave");
    batcave.appendChild(label);

    var parent = document.getElementById("gadget_list");
    var batteries = document.getElementById("batteries");
    parent.insertBefore(batcave, batteries);
  </script>
</body>

</html>
```

MALMÖ UNIVERSITET

Exempel på insertBefore()

```
<!DOCTYPE html>
<html>
…

<body>
  <h1>Batman's gadgets</h1>

  <ul id="gadget_list">
    <li id="batarang" class="real_deal">Batarang</li>
    <li id="batmobile" class="real_deal">Batmobile</li>
    <li id="batteries" class="bad_pun">Batteries</li>
  </ul>

  <script>
    var batcave = document.createElement("li");
    var label = document.createTextNode("Bat cave");
    batcave.appendChild(label);

    var batteries = document.getElementById("batteries");
    batteries.parentNode.insertBefore(batcave, batteries);
  </script>
</body>

</html>
```

MALMÖ UNIVERSITET

Ta bort element från trädet

Vi har två sätt att ta bort element från DOM-trädet:

- `parent.removeChild(child)`

  tar bort ett element

- `parent.replaceChild(element1, element2)`

  ersätter ett element

MALMÖ UNIVERSITET

Ta bort ett element

```
<!DOCTYPE html>
<html>
...
<body>
  <h1>Batman's gadgets</h1>

  <ul id="gadget_list">
    <li id="batarang" class="real_deal">Batarang</li>
    <li id="batmobile" class="real_deal">Batmobile</li>
    <li id="batteries" class="bad_pun">Batteries</li>
  </ul>

  <script>
    var parent = document.getElementById("gadget_list");
    var child = document.getElementById("batmobile");
    parent.removeChild(child);
  </script>
</body>

</html>
```

MALMÖ UNIVERSITET

Ta bort ett element

```
<!DOCTYPE html>
<html>
...
<body>
  <h1>Batman's gadgets</h1>

  <ul id="gadget_list">
    <li id="batarang" class="real_deal">Batarang</li>
    <li id="batmobile" class="real_deal">Batmobile</li>
    <li id="batteries" class="bad_pun">Batteries</li>
  </ul>

  <script>
    var child = document.getElementById("batmobile");
    child.parentNode.removeChild(child);
  </script>
</body>

</html>
```

MALMÖ UNIVERSITET

Ersätt ett element

```
<!DOCTYPE html>
<html>
…

<body>
  <h1>Batman's gadgets</h1>

  <ul id="gadget_list">
    <li id="batarang" class="real_deal">Batarang</li>
    <li id="batmobile" class="real_deal">Batmobile</li>
    <li id="batteries" class="bad_pun">Batteries</li>
  </ul>

  <script>
    var batcave = document.createElement("li");
    var label = document.createTextNode("Bat cave");
    batcave.appendChild(label);

    var parent = document.getElementById("gadget_list");
    var batteries = document.getElementById("batteries");
    parent.replaceChild(batcave, batteries);
  </script>
</body>

</html>
```

MALMÖ UNIVERSITET

Ersätt ett element

```
<!DOCTYPE html>
<html>
…

<body>
  <h1>Batman's gadgets</h1>

  <ul id="gadget_list">
    <li id="batarang" class="real_deal">Batarang</li>
    <li id="batmobile" class="real_deal">Batmobile</li>
    <li id="batteries" class="bad_pun">Batteries</li>
  </ul>

  <script>
    var batcave = document.createElement("li");
    var label = document.createTextNode("Bat cave");
    batcave.appendChild(label);

    var batteries = document.getElementById("batteries");
    batteries.parentNode.replaceChild(batcave, batteries);
  </script>
</body>

</html>
```

MALMÖ UNIVERSITET

Händelser i JavaScript

MALMÖ UNIVERSITET

Vad är händelser?

En händelse är ett meddelande som skickas till JavaScript-motorn, och som sedan påverkar körningen av programmet. Exempel:

- Användaren klickar på en knapp
- Ett AJAX-anrop (dvs HTTP) slutförs
- En timeout tar slut
- Ett formulär skickas

Vi kan lyssna efter dessa händelser och reagera på dem!

Vanliga händelser i JavaScript/DOM

- Blur – när ett objekt förlorar fokus
- Click – när användaren klickar på ett objekt
- Focus – när ett objekt får fokus
- Load – när objektet är färdigladdat
- Mouseover – när muspekaren svävar över objektet
- Select – när ett val i en drop down-meny väljs
- Submit – när ett formulär skickas

Reagera på händelser

Körning av JavaScript i webbläsaren är *asynkron*. JS-motorn har en lista av kod som den kan köra.

Motorn kör kod så länge den kan, en funktion i taget, och delegerar allt hårt arbete till webbläsaren.

När webbläsaren är färdig med ett uppdrag (eller får en input från användaren) läggs ett uppdrag i JavaScript-motorns att göra-lista.

Metoden .addEventListener()

```
element.addEventListener(event, function, useCapture);
```

- Parametern `event` är en sträng som anger vilken händelse vi vill lyssna efter.
- Parametern `function` är en funktionspekare och berättar vilken funktion som ska köras när händelsen inträffar. Den kan vara anonym.
- Parametern `useCapture` berättar om vi vill använda event bubbling eller event capturing. Överkurs, och helt frivilligt att använda.
- Ett element kan ha flera event listeners, även per händelse!

MALMÖ UNIVERSITET

Exempel med addEventListener()

```
<!DOCTYPE html>
<html>
...

<body>
  <h1>Welcome to the Wayne Mansion intercom system!</h1>

  <span id="caller">Click to summon Alfred</span>

  <script>
    var caller = document.getElementById("caller");
    caller.addEventListener("click", callAlfred);

    function callAlfred() {
      alert("You called, Master Wayne");
    }
  </script>
</body>

</html>
```

MALMÖ UNIVERSITET