JavaScript och DOM

...och info om peer reviewing

Johan Holmberg 2022-04-04

MALMÖ UNIVERSITET

Dagens föreläsning

- DOM
- Händelser i JavaScript
- Peer reviewing

MALMÖ UNIVERSITET

DOM

VanillaJS – JavaScript utan ramverk

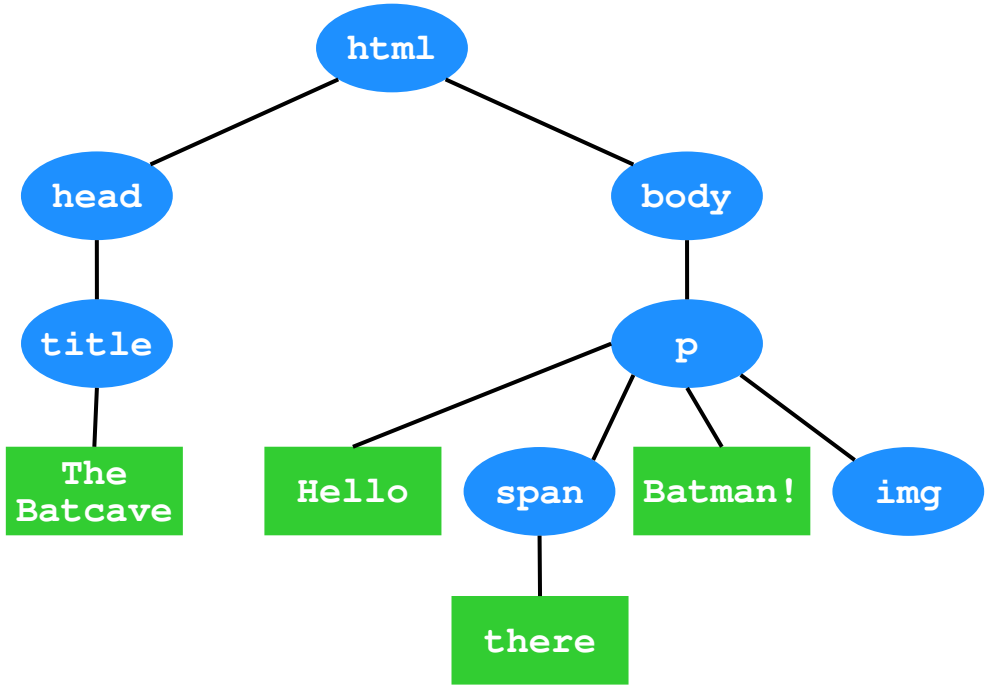
Vad är DOM?

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.getElementsByTagName(tagName)` → en samling av element
- `document.getElementsByClassName(className)` → en samling av element
- `document.querySelector(query)` → ett element

MALMÖ UNIVERSITET

Hitta alla element av en specifik typ

```
let elements = document.getElementsByTagName("p");
```

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

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

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

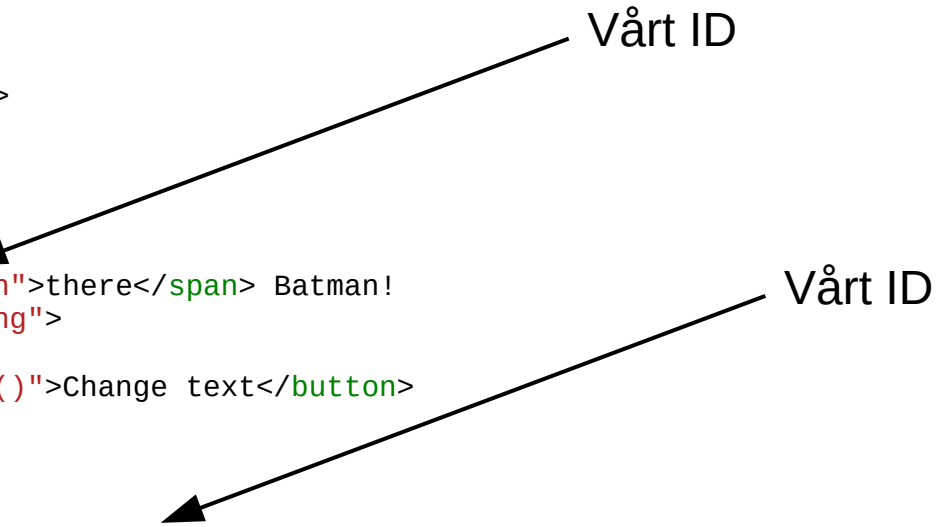
MALMÖ UNIVERSITET

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>The Batcave</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>
```



MALMÖ UNIVERSITET

Exempel på användande av klass

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

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

Returnerar den första noden av klassen `someClass`.

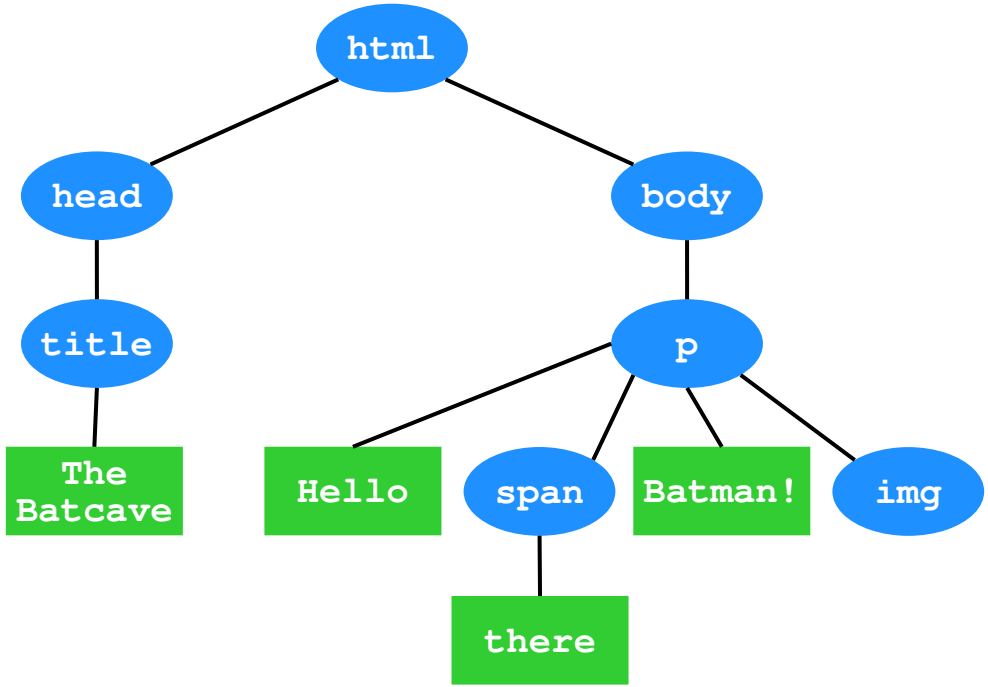
Kan användas med:

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

MALMÖ UNIVERSITET

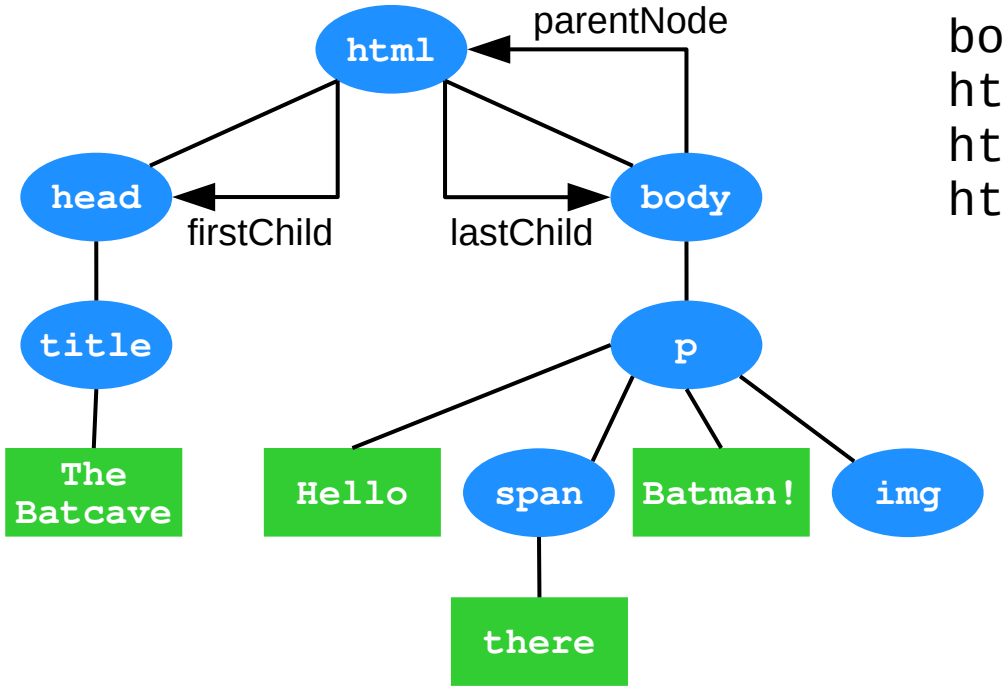
Navigera i DOM

```
<!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

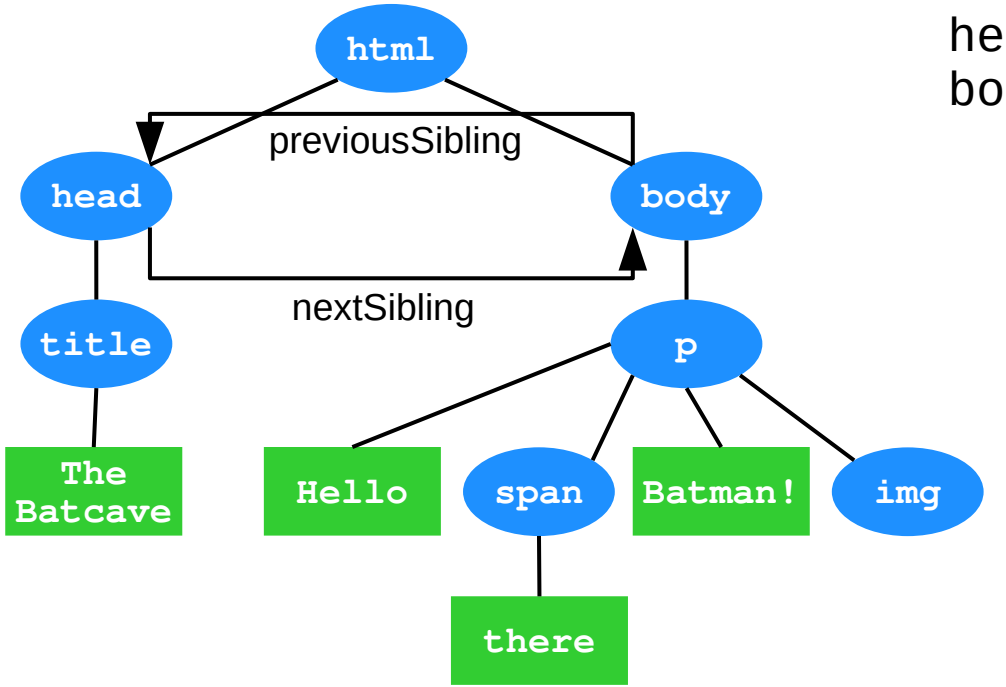
Navigera i DOM – föräldrar och barn




```
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

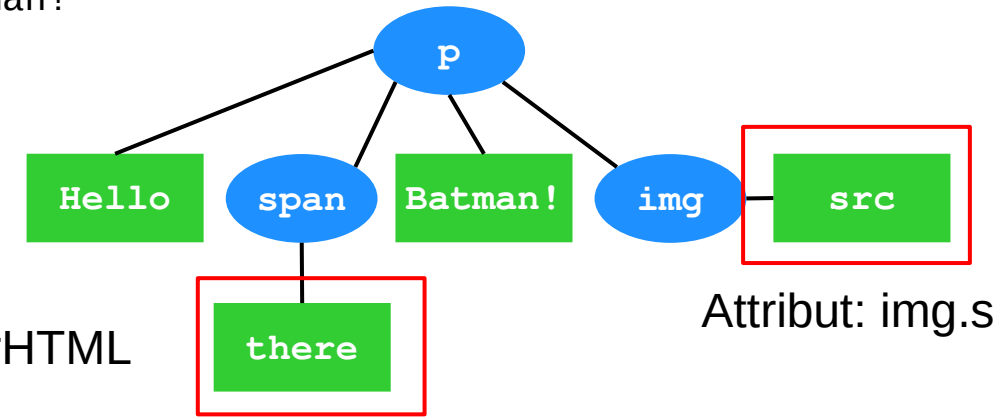
Modifiera en nod

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>The Batcave</title>
</head>

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

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

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

</html>
```

MALMÖ UNIVERSITET

Arbeta med attribut

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

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

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

  <script>
    let 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>
    let 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`

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

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

    let parent = document.getElementById("gadget_list");
    let 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>
    let batcave = document.createElement("li");
    let label = document.createTextNode("Bat cave");
    batcave.appendChild(label);

    let 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>
    let parent = document.getElementById("gadget_list");
    let 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>
    let 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>
    let batcave = document.createElement("li");
    let label = document.createTextNode("Bat cave");
    batcave.appendChild(label);

    let parent = document.getElementById("gadget_list");
    let 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>
    let batcave = document.createElement("li");
    let label = document.createTextNode("Bat cave");
    batcave.appendChild(label);

    let 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>
    let caller = document.getElementById("caller");
    caller.addEventListener("click", callAlfred);

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

</html>
```

MALMÖ UNIVERSITET

Peer Reviewing

Vilken återkopping ger du, när, och varför?

MALMÖ UNIVERSITET

Vad är peer reviewing?

Peer reviewing är en teknik som används för att upptäcka brister i en framställning, exempelvis en vetenskaplig text. Idén är att någon annan än författaren själv går igenom framställningen och hittar fel och brister i logik, språklig framställning och metod, vilka måste åtgärdas innan arbetet slutligen läggs fram.

Vi kommer i kursen att använda peer reviewing, eller snarare *kamratgranskning*, för att säkerställa kvaliteten i era inlämningar.

MALMÖ UNIVERSITET

Kamratgranskning != bedömning

MALMÖ UNIVERSITET

Varför kamratgranskning?

Kamratgranskning ger dig som granskar en möjlighet att förbättra din förmåga att läsa och förstå kod, samtidigt som du övar upp din förmåga till att ge konstruktiv återkoppling till dina kolleger/kursare.

I din roll som granskad kommer du å andra sidan att få en möjlighet att lära dig att ta konstruktiv kritik och implementera de förändringsförslag du får, samt att öva upp din förmåga att bedöma relevansen i den återkoppling du får.

MALMÖ UNIVERSITET

Vad är konstruktiv kritik?

Vid granskning är det viktigt att ge *konstruktiv kritik*, vilket innebär att du som granskare inte bara kan säga att ”det här är dåligt/fult/etc”, utan även måste peka på ett lösningsförslag. I begreppet innefattas även tanken om att inte ge kritik för kritikens skull, utan för att just förbättra. Detta innebär att du även kan ge *positiv kritik*, genom att peka ut något som är särskilt bra, nyskapande eller liknande. Du kan även ge konstruktiv kritik genom att peka ut ett förbättringsförslag som höjer kvaliteten på något som redan är bra, men som har förbättringspotential.

MALMÖ UNIVERSITET

Vad ger jag återkoppling på?

Du ger återkoppling på fyra saker:

- Strukturella aspekter av inlämningen (uppdelning av struktur, logik, utseende, etc)
- Stilistiska aspekter av inlämningen (kodstil, etc)
- Logiska brister hos inlämningen (eventuella fel i koden, buggar, bad code smell, etc)
- Positiva aspekter av arbetet, exempelvis något som du själv lärt dig av det du just läst.

Hur skriver jag återkopplingen?

Tänk på din tonalitet – var inte onödigt negativ eller anklagande. Det som händer i den granskandes hjärna är då att den går i försvarsposition och har svårare att ta till sig av din kritik. Tänk också på att kritisera *arbetet*, inte *personen*. Ett exempel:

- Mindre bra: “Du har skrivt en metod som inte fungerar”
- Bättre: “Metoden xx fungerar inte”

MALMÖ UNIVERSITET

Hur skriver jag återkopplingen?

Använd CodeGrade via Canvas för att skiva återkopplingen. Du behöver inte – och ska inte – använda externa program för detta.

MALMÖ UNIVERSITET