



Ett intro till JavaScript

# Agenda

- Vad
- Varför
- Infoga JS
- DOM
- Objekt
- Events/eventhandlers
- Öva lite...

- <http://www.codecademy.com/tracks/javascript>
- <http://www.w3schools.com/js/default.asp>

# Vad?

- JavaScript är ett *scriptspråk*;
  - ett s.k "lightweight" språk;
  - interpretarande (behöver inte kompileras) – exekveras i klienten (läsaren)
- Kan bäddas in med htmlkod, läggas i extern fil (infogas som ramverk).
- Tolkas av (alla) webbläsare Chrome, Safari, IE, Firefox, Opera, etc
- Väldigt enkel syntax – enkelt att skriva och förstå
- JavaScript != Java  
(även om det i vissa fall liknar Java)

# Varför?

- För interaktivitet, dynamik och logik till webbsidor.
- Med JavaScript kan man t ex få följande funktionalitet;
  - Reagera på händelser beroende på t ex vad användaren gör, data/innehåll förändras.
  - Få tillgång till innehållet i elementen.
  - Validera data.
  - Skapa effekter.
  - Skapa webbapplikationer.

# JS är stort...

- JavaScript är idag STORT.
- Förutom interaktivitet och dynamik finns en mängd olika ramverk/toolkits och bibliotek skrivna i JavaScript.

De innehåller färdig funktionalitet för att åstadkomma allt möjligt

- allt från att skapa färdiga webbapplikationer till att manipulera innehåll i ett element på en webbsida.

- *Man måste kunna programmera för att till fullo kunna använda dem.*



Webbsockets

*Knockout.*



# Infoga JS

- Ett antal sätt att infoga JS (ren JS, inga ramverk);
  - Externt script
  - Script-element i html-filen
  - Direkt i vissa taggar via t ex onclick-attributet.
  - En kombination av ovanstående.

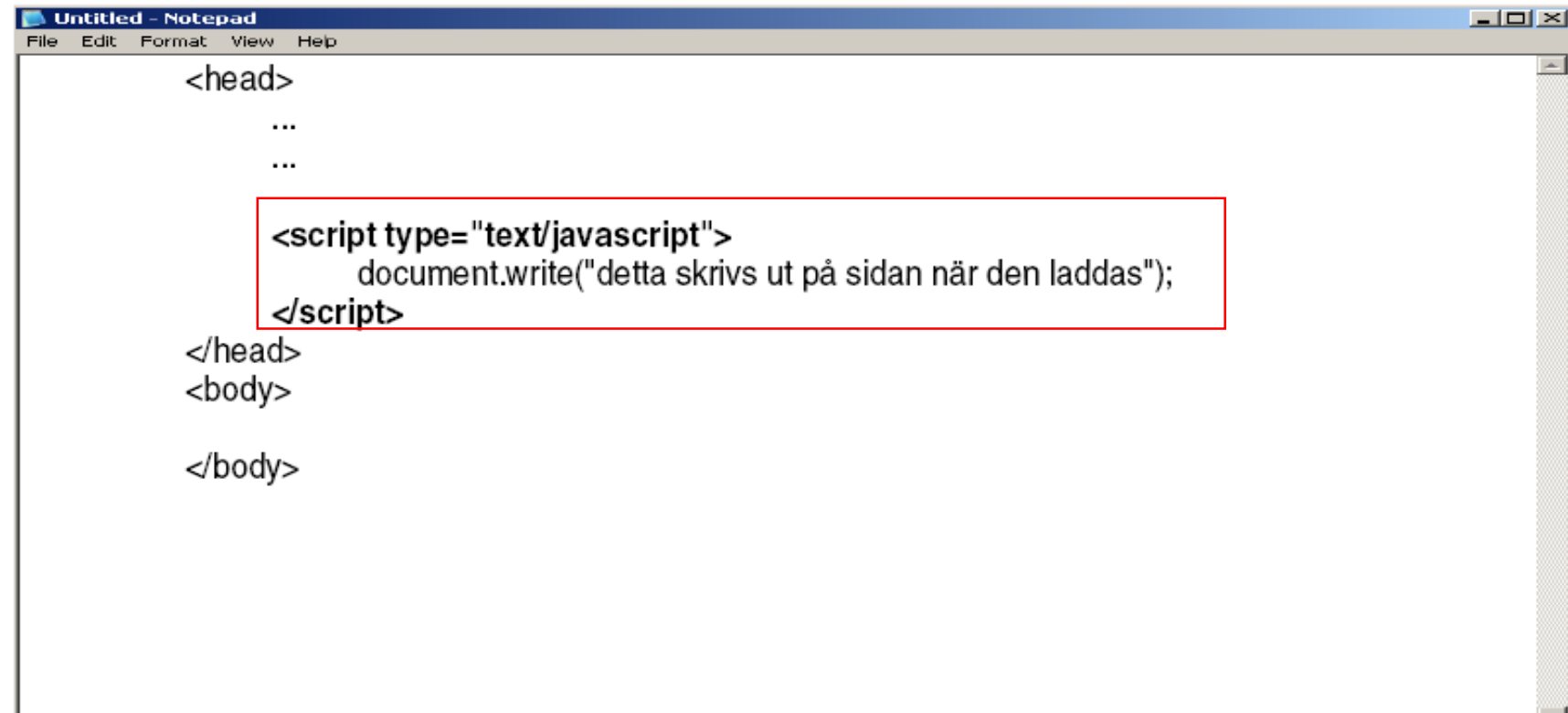


# JS i html-taggar

- Vi kan skriva JS i vissa html-taggars attribut. Attributen måste vara kopplade till en s.k händelse (klicka, hovra etc).
- Exempel:  
`<button onclick="alert(' Oh noes! ');">Clickz me</button>`
- I vissa fall nödvändigt, t ex om en knapp ska reagera på klick.
- Osmidigt om det blir mycket JS.

# Infoga script i html-filen

- Använd `<script></script>` - elementet.  
All JS placeras där emellan.  
Vi kan ha flera script-element (var som helst i body/head).
- Script som placeras i head kommer att läsas in innan body läses in.



```
Untitled - Notepad
File Edit Format View Help

<head>
  ...
  ...
  <script type="text/javascript">
    document.write("detta skrivs ut på sidan när den laddas");
  </script>
</head>
<body>

</body>
```

```
Untitled - Notepad
File Edit Format View Help

<head
    ....
    <script type="text/javascript">
        document.write("Nu infogar vi scriptet i head.");
    </script>
    <script type="text/javascript">
        document.write("Nu infogar vi scriptet i head igen.");
    </script>
</head>
<body>
    <script type="text/javascript">
        document.write("Nu infogar vi scriptet i body.");
    </script>

    <script type="text/javascript">
        document.write("Hoppsan! Ytterligare ett script i body.");
    </script>

</body>
</html>
```

# Infoga externt script

- Jmfr med externt stylesheet.
- Extension på JavaScriptfiler: **.js**
- Fördelar:
  - Uppdateringar
  - Mindre komplexa filer
  - Återanvändning
  - ...
- Nackdelar:
  - Fler filer

# Infoga externt script 2

- `<script type="text/javascript" src="sc.js"></script>`

*eller*

`<script src="sc.js"></script>`

- Länka in scriptet i head (oftast).
- Vi kan sedan anropa funktionalitet i externt script från andra script-avsnitt i vår html-fil.

# Summering infoga JS

- Vi kan skriva JS inom `<script>...</script>`, som läggs i head/body.
- Vi kan skriva externa JS-script i .js-filer som länkas in med `<script src="myScript.js"></script>`.
- Script som läggs i head läses in före innehållet i body.
- Vi kan infoga flera script, både i head och body.

# DOM

- Document Object Model

# DOM

- JavaScript som språk för webbsidor har byggts upp mha av en modell - Document Object Model.
- DOM byggs upp av s.k **objekt**.  
Dessa objekt kan vi använda för att manipulera innehållet på en webbsida.
- Ett objekt representerar 'någonting', t ex själva webbläsarfönstret eller ett element.  
Dessa objekt har i sin tur en massa funktionalitet.
- DOM är en W3c standard och består av tre delar/levels;
  - Core DOM
  - XML DOM
  - HTML DOM

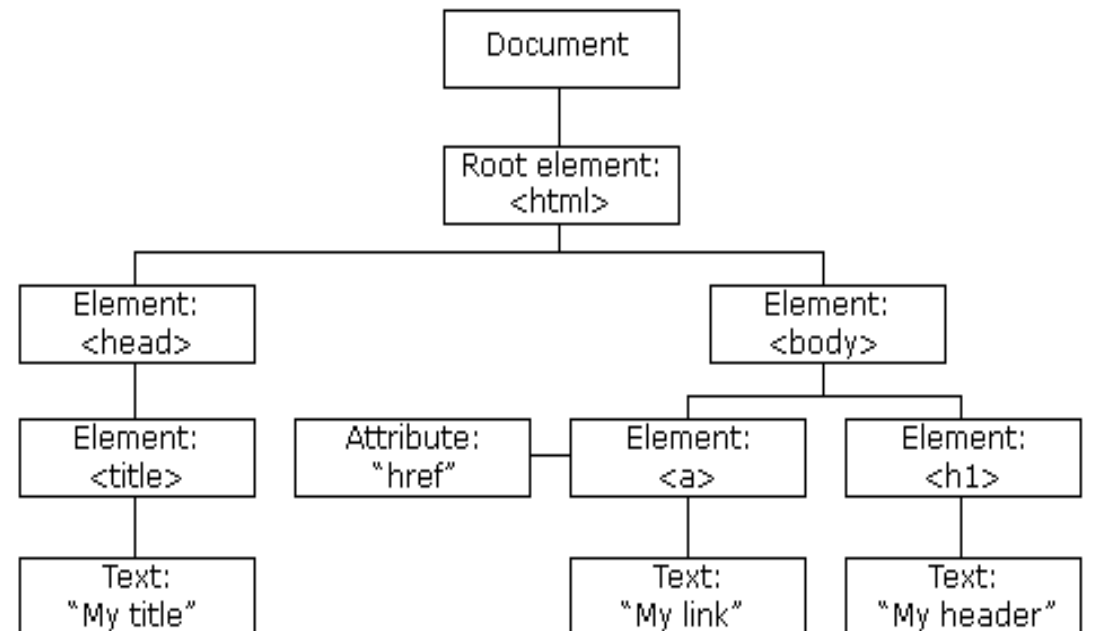


# Objekt i DOM

- *Exempel på objekt;*

<u>Namn</u>	<u>Beskrivning</u>
<b>window</b>	webbläsaren ('fönstret')
<b>body</b>	body elementet
<b>document</b>	hela html-dokumentet/filen med dess innehåll
<b>style</b>	en stylesheet för ett element
<b>navigator</b>	information om webbläsaren
...	...
...	...

- Med DOM kan vi alltså 'komma åt' *alla* element på sidan.



# Funktionalitet för objekten

- Varje objekt i sig har en massa funktionalitet.  
T ex **document** objektet.
- Funktionaliteten finns i tre olika former;
  - Propertys (jmf css)
  - Methods/Metoder
  - Collections
- Vi kommer åt en property, en metod eller en collection genom att skriva objektets namn följt av en punkt, därefter specifik property/metod/collection.

**document.title**  
**document.write()**  
**document.images**

# Prop's/Meth's/Coll's

- **Property**

information/värden vi kan påverka direkt

*objektetsNamn.property*

**document.title** = "Ny titel på sidan";

---

- **Metoder ()**

en funktion som kan användas för objektet.

*objektetsNamn.metod()*

**document.write**("Hejsan");

---

- **Collections []**

samlingar med en massa värden i. Värdena är indexerade (finns på numrerade platser i samlingen).

*objektetsNamn.collection[]*

**document.images**[3]

# Ett exempel – getElementById()

- Ett exempel på en metod i document är getElementById(), som används för att komma åt ett specifikt element i dokumentet. Elementet måste ha ett värde för attributet id.
- Metoden ger oss tillgång till ett element (som ett objekt). Vi kan sedan arbeta med elementet, te x förändra css, ändra text etc.

- **Html:** `<p id="myText">Bananer och bacon.</p>`

**js:**

```
document.getElementById("myText").innerHTML = "Apelsiner";
```

# Forts getElementById()

```
<p id="myText">Bananer och bacon.</p>
```

```
document.getElementById("myText").innerHTML = "Apelsiner";
```

- **Annat sätt;**

```
var txt = document.getElementById("myText");
```

```
txt.innerHTML = "Apelsiner och kalopps";
```

```
txt.style.color = "blue";
```

# Avslutningsvis

- Det finns alltså en mängd objekt (som i sin tur har en mängd propriety, metoder och collections).
- Se på W3's referens.
- Det är DOM vi använder för att manipulera saker och en ting på en sida. Tillsammans med events och eventhandlers så kan vi göra detta för att t ex svara på operationer från användaren.
  - *när bilden x klickas så byter vi ut texten i elementet z.*
  - *När länken a får mouseover så döljer vi div-boxen b.*
  - *Etc etc.*

# Events och eventhandlers

- Händelser och händelsehanterare

# Events/eventhandlers

- Med **js** kan vi 'fånga' händelser i webbläsaren och element; när ngt händer kan vi använda js för att svara på just den händelsen.
- För dessa händelser, finns färdiga hanterare vilka vi kan använda; **eventhandlers**.
- Exempel på händelser;

Event	Inträffar när
<b>load</b>	ett element laddas, t ex body
<b>click</b>	ett element klickas på
<b>mouseover</b>	muspekaren förs över ett element
<b>mouseout</b>	muspekaren lämnar ett element
<b>focus</b>	när ett element 'får fokus'
...	...
...	...



# Eventhandlers

- För varje event finns alltså **s.k event handlers**.  
De 'aktiveras' vid respektive event och vi kan exekvera specifik js.
- Vi infogar dessa handlers i html via attribut i taggarna.
- Exempel;  
`<p onclick="gör ngt roligt med js">`  
``

- Exempel event handlers:

Handler	Aktiveras av event
<b>onclick</b>	click, element klickas på
<b>onmouseout</b>	mouseout, muspekare lämnar elementet
<b>onmouseover</b>	mouseover, muspekare över elementet
<b>onchange</b>	change, innehåll i elementet ändras
<b>onload</b>	load, elementets laddas
...	...
...	...

- Låt oss öva lite....

# Övning

- Skapa en sida med ett text-element (<p>) med text i.
  - Lägg till JavaScript som gör att användaren kan klicka, på något på sidan, som i sin tur döljer och/eller visar texten.
  - Lägg till ytterligare funktionalitet som gör att användare kan byta textfärg, storlek på texten etc etc.

# Övning

- Undersök elementet **`<input type="text">`** (textfält)
- Lägg till javascript som tar innehållet från ett textfält och skriver ut det i en s.k alert-prompt.
- Lägg till javascript som hämtar innehållet från ett textfält och lägger till detta till en s.k textarea. Innehållet i textarean ska alltså kunna utökas flera gånger.

# Övning

- Lägg till två bilder till en html-sida.
- Lägg till javascript så att bilderna byter position med varandra. En användare ska kunna klicka på en knapp (button) och då byter bilderna plats med varandra.

# Övning

- Undersök olika eventhandlers.
- Titta på onload;
  - Lägg till en onload för body som exempelvis skriver en text till ett p-element

# Övning

- Lägg till många bilder (5-6 st) till en html-sida.
- Förändra, med javascript, bildernas storlek genom att förminska eller förstora dem.  
Användaren ska kunna göra det med knappar för varje bild, och/eller för alla bilder på samma gång.
- Tips – kolla upp `images[]`