AngularJS

Varför AngularJS?

Javascript används för att göra sida mer interaktiva. Dock kan det lätt bli rörig kod om man ska skriva i ren Javascript-kod och tar tid.

AngularJS är ett ramverk för att lättare göra sidor mer interaktiva, att något händer på sidan.

Det har metoder för att manipulera DOM, istället för att du ska göra det själv.

Med AngularJS kan man göra en s.k. Single Page Aplikation. Det innebär att inte hela sidan laddas om utan bara delar av den.

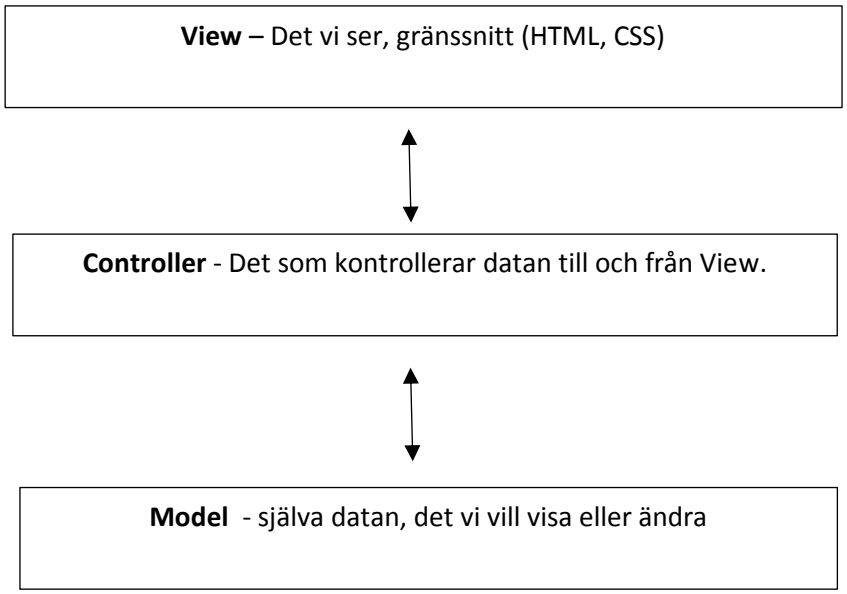
När man skriver en Webbapplikation har man dels det som användaren ser (HTML m.m.), dels det som händer bakom kulisserna (Javascript). Att själv hantera detta kan lätt bli jobbigt och rörigt.

För att lösa detta problem använder man sig delvis av MVC-modellen.

Vad är då MVC?

MVC står för modell – view – controller och är ett sätt att dela upp en applikation så att de olika delarna är separerade.

Det vi ser – View – ska inte blandas ihop med vår data. Datan är det som är det intressanta på sidan. View visar bara datan på ett bra och tydligt sätt.



AngularJS har dock något som kallas Modell – View – Whatever eller MV*

Det innebär att man har **Model** som är den data som vi är intresserade av och en **View** som visar det vi är intresserade av.

Data-binding

När vi har någon data vi vill visa måste vi göra en del manuellt arbete för att få det att hända. När vi hade Javascript fick vi skriva kod så att vi kunde ändra i vår data (t.ex. texten vi ville visa).

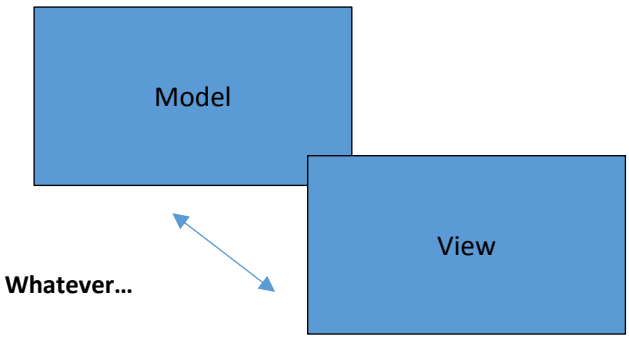
Ju mer komplexa saker vi ska hantera, desto mer kod.

AngularJS har ett koncept för att hantera detta som kallas **data-binding**.

Det som sker i **model** (datan) visas automatiskt i vår **view** och det som sker i vår tvärtom.

Vi behöver inte sköta detta utan det sker automatisk genom någonting.

var aName = 'Lars';



** Namn: Lars **

När vi ändrar något i **model**, vår data, ändras det som visas i vår view som är HTML. (Egentligen DOM som är trädet som byggs upp av HTML)

När vi ändrar något i **view**, ändras också vår data, vår **model**.

De är sammanbundna.

Det som binder dom kan vara olika saker och är det man i Angular kallar **whatever**.

Man har ändå något slags kontroller.

Vi tittar på lite exempelkod, en första sida.

```
<!doctype html>
<html ng-app> <!-- Talar om att detta är en Angular-applikation -->
  <head>
<!—Infogar Angular-ramverket -->
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.9/angular.min.js">
    </script>
  </head>
  <body>
    <div>
      <label>Name:</label>
      <input type="text" ng-model="name" /> <!— ng-model är vår data, det vi vill visa, det intressanta. Textrutan binds till vår model som binds till vår HTML-->

      <hr>
      <h1>Hello {{name}}!</h1> <!— ng-model skrivs ut här. Kallas för directive -->
    </div>
  </body>
</html>
```

Nu Ska vi testa vår kod

Börja med att som innan skapa ett repository som innan på gitHub. Klona sedan ner ditt repo till projektmappen. Se till att du är i rätt mapp (projektmappen).

Öppna din editor och skriv in eller klipp och klistra koden. Om du klipp- och klistrar så tänk på att vissa tecken kan bli fel som du då får ändra manuellt.

Spara i din nya mapp.

Se till att du har dina fönster sida vid sida som innan. Editorn på den ena sidan och Chrome på den andra.

Se till att Chrome är aktiv och tryck CTRL + O. Då får du välja en fil att öppna. Välj den fil du precis sparade.

Uppdatering direkt

När du klickar på en knapp t.ex. uppdateras normalt hela sidan, d.v.s. hela sidan hämtas igen.

AJAX är en teknik som funnits tidigare för att hämta bara delar av sidan. Det fungerar på liknande sätt med Angular.

Testa nu att skriva något i rutan. Om allt fungerar ska det synas direkt på sidan. Även om du suddar så ska det försvinna direkt.

Det beror på s.k. data-binding. Att view (det du ser, gränssnitt) och model (data) är sammanbundna.

Lite matematik

Nu ska vi testa hur det kan fungerar med lite matematik. Testa på samma vis. Så fort du ändrar siffrorna ändras resultatet. Spara som html-fil.

Märk att vi inte behöver använda variabler utan kopplar bara vår model, vår data, till textrutan.

```
<html ng-app>
  <head>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js">
    </script>
  </head>
  <body>
    <div>
      <label>First number:</label>
      <input type="number" ng-model="num1" /><br />
      <label>Second number:</label>
      <input type="number" ng-model="num2" />
      <hr>
      <h1>Sum: {{num1 + num2}}</h1>
      <h1>Product: {{num1 * num2}}</h1>
    </div>
  </body>
</html>
```

Uppgifter (Lättare)

Använd exempelsidan som vi hade i Bootstrap. Kommentera bort knappen.

Koppla det du skriver till <h1>-taggen där det står Hello Word som vi gjorde nu. Testa att skriva och sudda ut olika saker.

Uppgifter (Lite svårare)

Använd två eller fler textrutor där du skriver in siffror.

När du skriver in siffrorna ska det någonstans visas summan (plus), differens (minus), produkt (gånger) och kvot (delat med)

Uppgifter (Svår)

Koppla en sökfunktion till rutan så du söker på sidan när du skriver in.

Lägga till en controller

Nu ska vi prova att lägga till en s.k. controller som hanterar vad som visas i vyn (view). Vi använder oss av en klass i JavaScript. Skapa en ny mapp som du kallar kanske matte2. Lägg både HTML-filen där och Javascript-filen. En controller ligger i en modul som är en slags behållare för där vi kan lägga kod.

Javascript-filen

```
angular.module('app', []).controller("MainController", function() {
    this.num1 = 0;
    this.num2 = 0;
  });
```

HTML-filen

```
<!doctype html>
<html>
  <head>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js">
    </script>
    <script src="app.js"></script>
  </head>
  <body>
      <div ng-app="app" ng-controller="MainController as main" >
          <label>First number:</label>
          <input type="number" ng-model="main.num1" /><br />
          <label>Second number:</label>
          <input type="number" ng-model="main.num2" />
          <hr>
          <h1>Sum: {{main.num1 + main.num2}}</h1>
          <h1>Product: {{main.num1 * main.num2}}</h1>
    </div>
  </body>
</html>
```

Förklaring

1. `<script src="app.js"></script>` – **Denna raden gör att vi får kontakt med vår externa Javascript-fil.**

2.Här har vi flyttat ng-app ner till en div. Det innebär att det i denna div:en är angivit att det är en Angular-applikation. Vi anger också en s.k. modul som är den som vi anger i vår klass.

Modulen ’app’ är alltså den som vi anger i vår klass som vi ser med pilen.

Vi anger också en s.k. controller som är den som kontrollerar vad som visas. Det är också den vi anger i klassen. (se pil).

Vi anger också en alias för MainController som kan se lite som ett objekt. (På liknande sätt som man kan göra i SQL)

Med hjälp av vårt alias och s.k. punktnotation använder vi de uttryck (variabler) som finns i vår klass.

Övning

Skriv in koden och testa att det fungerar.

Vi har egentligen inte gjort så mycket skillnad utan övningen är mer till för att se hur man kan separera ut kod för att kunna skapa vårt MV*-mönster.

Vi kan dock som i ett objekt redan sätta ett värde. Prova att ändra värdet för variabeln i vår Javascriptfil och se hur det påverkar resultatet.

Övning 2

Lätt

Skriv in denna koden för att få lite snyggare presentation.

Lite svårare

Presentera resultatet på en snyggt ännu snyggare sätt.

HTML-filen

```
<!doctype html>
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js">
    </script>
    <script src="app.js"></script>
  </head>
  <body>
       <div ng-app="app" ng-controller="MainController as main" >
          <label>First number:</label>
          <input type="number" ng-model="main.num1" /><br />
          <label>Second number:</label>
          <input type="number" ng-model="main.num2" />
          <hr>
          <h1>Sum: {{main.num1}} + {{main.num2}} = {{main.sum + main.num2}}</h1>
          <h1>{{main.num1}} * {{main.num2}} = {{main.num1 * main.num2}}</h1>
    </div>
  </body>
</html>
```