Studie av ramverket Angular, React och Redux arkitektur

John Tran

Abstrak

Webbsidor har ändrat mycket på senaste åren. Webbsidor som ursprungligen är designat för att det dela information och bilder ihoplänkad via hyperlänkar har ändrat till webbapplikationer program som är jämförbara med skrivbordsprogram. Utveckling av webbapplikationer är möjligt med förbättring och tillkomst av nya teknologier såsom Javascript, DOM, AJAX och REST. Men Utveckling av webbapplikationer är inte utan sina problem. Omvandling from webbsidor till webbapplikationer kräver användning av nya programmeringsprinciper eftersom det inte längre handlar om statiska webbsidor med innehåll av text och bilder. Den här kandidaten kommer att gå igenom ramverket Angular, React och Redux som används för underlätta utveckling av webbapplikation.

1. Inledning

Webbrowser har på senaste åren blivit en av den mest använda mjukvaruprogram i många olika apparater alltifrån persondator, smarttelefoner, tablet, tv etc. Webbsidor som ursprungligen är enkla statisk text- och bildbaserade dokument anslutna via länkar har utvecklats till dynamiska webbsidor med användarinteraktion. En av de första ändringar i webbsidor var med introduktion av Flash, Realplayer och Java Applet, vilket möjliggör att webbsidor blev mera interaktiva. Istället för statisk text- och bildbaserade dokument liknar webbsidor multimedia presentationer.

Med introduktion av Ajax och Javascript har webbsidor undergått ytterligare ändringar. Numera används webbsidor ofta som applikationer så kallade webbapplikationer som påminner om skrivbordsprogram. Många program idag är skriven för att användas på Internet istället för specifik operativsystem.

Utveckling av webbapplikationer är dock inte utan sina problem. Webbapplikationer skrivna för att köras på webbrowser skiljer sig mycket från skrivbordsprogram. Den består av många olika lager. Klientsida som körs på webbrowser består av HTML, Javascript och CSS. Serversida fungerar som åtkomstkontroll till data som är lagrad i någon form av databaslagring. Problem som webbapplikationer stöter på är att det finns något tidigare programmeringsprinciper webbapplikationer, i likhet med skrivbordsprogram som också drabbad av spagettikod.

Många ramverk har utvecklats för att förenkla och skapa en standardisering för utveckling av webbapplikationer som följer programmeringsprinciper. Det är möjlig att skriva webbapplikationer utan att använda sig av dessa ramverk men är mycket tidskrävande.

1.1 Syfte

Denna kandidat kommer att undersöka två populära ramverk för utveckling av webbapplikationer Angularjs och React. Redux som inte egentligen tillhör i någon av de två nämnda ramverk tillkommer också eftersom React i sig själv är endast en view del och ramverk är baserad på MVC designmönster som en helhet.

Fästen båda ramverk är utvecklad för att bygga webbapplikationer följer de olika principer. Kandidaten kommer att undersöka skillnaden på dess arkitektur och implementationer.

1.2 Struktur

Kapitel 2 presenteras översikt av webbapplikationer uppbyggnad och MVC designmönster som de flesta ramverk är baserad på. Kapitel 3 kommer att delas i tre delar där Angularjs, React och Redux teknologier gås genom. Kapitel 4 Diskussion av de två ramverk fördelar och nackdelar. Kandidaten kommer endast att behandla klient sida och lite av server sida.

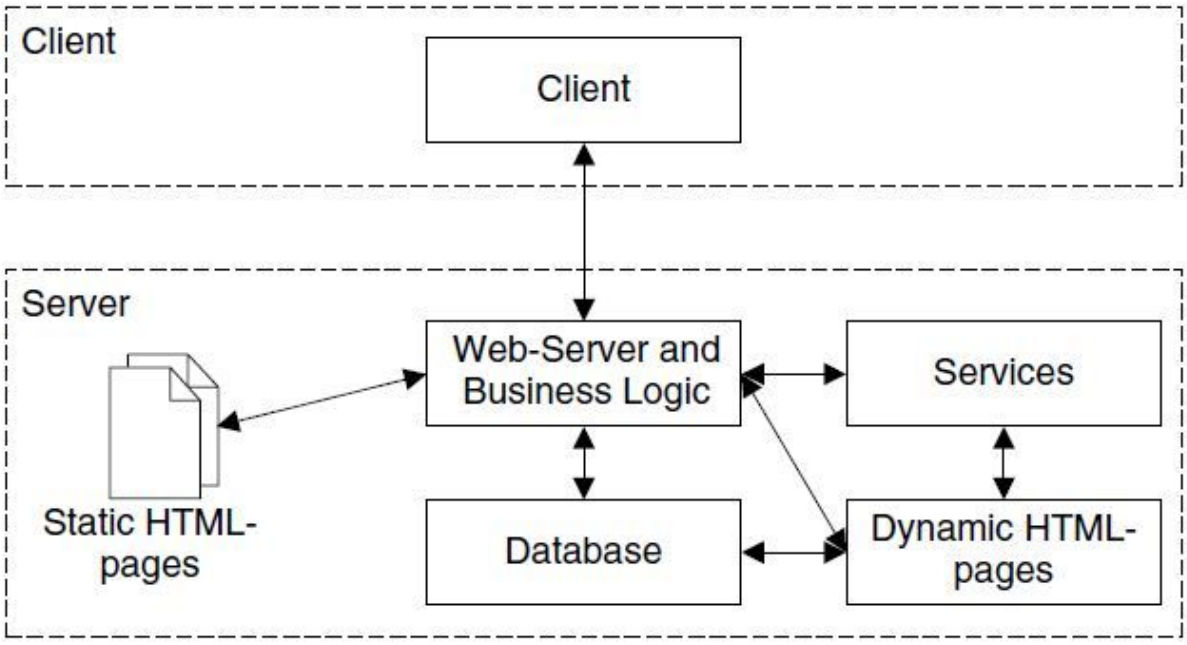
2. Webbsidor och Webbapplikationer översikt

I detta kapitel presenteras bakgrund och teori av de teknologier som ligger bakom Webbsidor och webbapplikationer. Kapitel 2.1 presenteras översikt av webbsidor, dess uppbyggnad och hur de fungerar i helhet. Kapitel 2.2 presenteras Webbapplikations bakgrund och teorier. 2.3 presenteras det teknologier som ligger bakom webbsidor och webbapplikationer.

2.1 Webbsidor

Webben är ursprungligen designat att fördela information, där varje webbsida består av text dokument och bilder. Dessa webbsidor är ihoplänkade via hyperlänkar. Dessa dokument är statiska.

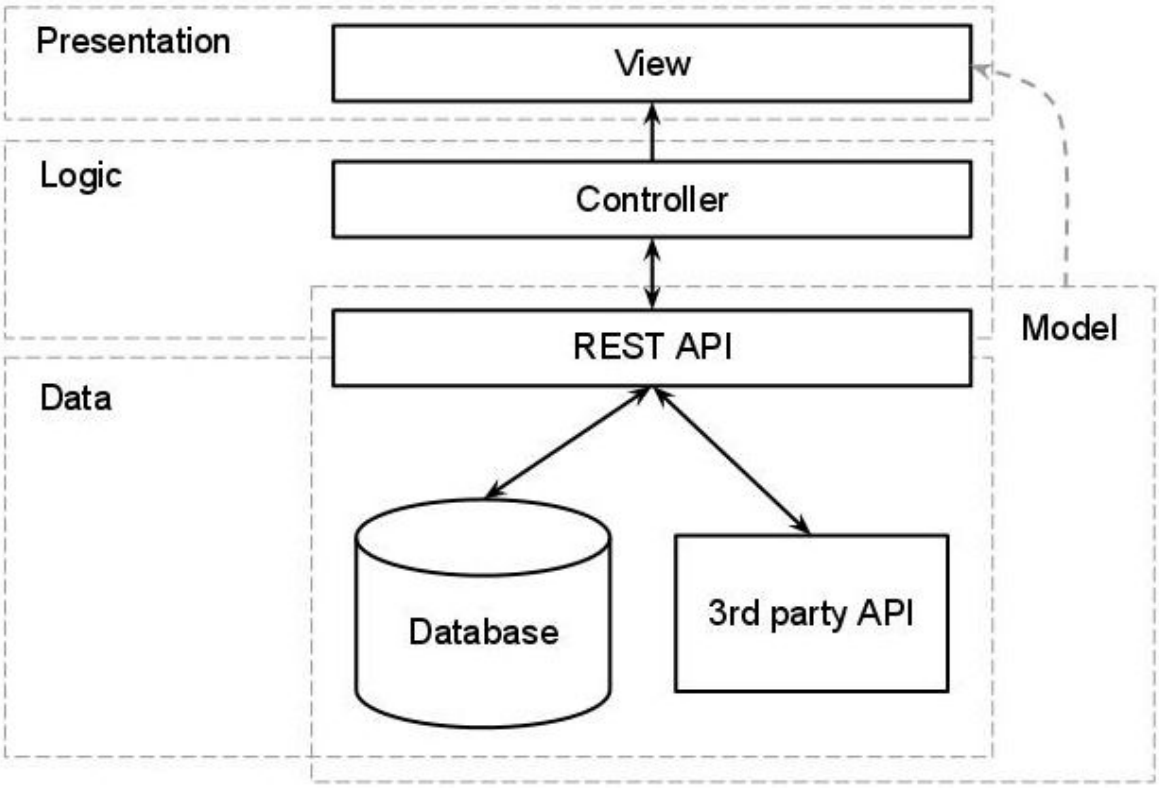
Klassiska webbsidor fungerar på så sätt att browsern processar användar input som sänds till server i form av HTTP begäran. Server hanterar begäran och skickar tillbaka den önskade webbsidan som svar. Bilden nedan 1.0 beskriver processen av klassiska webbsidor. Den orginala arkitektur för webbsidor är mycket begränsad på grund av att dessa webbsidor är statisk. Hela webbsidan måste renderas på nytt med varje begäran på ny information. Medan webbsida renderas är det inte möjligt att interagera med webbsidan. Även med introduktion av Javascript som tillåter små skript på klientsida är webbsidor fortfarande mycket begränsad på grund av den klassiska webbsida arkitektur [1].



Figur 1.0: Klient/Server arkitektur [2]

2.2 Webbapplikationer (ensidesapplikation)

Med tillägg av nya teknologier såsom AJAX och REST (se chapter) och snabbare browser motorer såsom V8 möjliggör utveckling av en annan typ av webbsidor kallad webbapplikationer som kan jämföras med applikationer på skrivbordsdator. Arkitektur för webbapplikationer fungerar på så sätt att vid första laddning laddas hela applikationen och server begäran därefter är endast för tillgång av data. Fördelen med denna teknik är att webbsidan måste inte laddas varje gång ny data hämtas och är möjligt att interagera med webbsidan medan hämtning av data pågår. Denna typ av webbapplikation kallas oftas som single page applikation eller med svensk översättning ensidesapplikation. Bilden 2.0 nedan vissa arkitektur för Ensidesapplikationer . Klientsidan innehåller ensidesapplikation presentation, logik och tillstånd data medan serversidan ansvar för beständig data [4].



Figur 2.0: webbapplikation arkitektur [3]

2.2.1 Presentations lager

Ensidesapplikation presentations lager fungerar som användargränssnitt och är uppbyggd med HTML, CSS och Javascript. Efter initiala laddning av sidan sker nya uppdatering via Javascript. Script skriven i Javascript modifier DOM-objekt som presenterar struktur och innehåll av sidan. HTML bestämmer layout av sidan och CSS hur sidan skall visas. Mera information om HTML, CSS, Javascript i (2.3.2) [4].

2.2.2 Logik lager

Ensidesapplikation Logik lager är skriven i Javascript ett dynamisk programmering språk som stöds i alla moderna browser. Logik lager hanterar separation av data och presentations lager. Med introduktion av AJAX är det möjligt att hämta data asynkront och bygga upp sida med DOM-objekt av varierande tillstånd. Eftersom sidan laddas endast en gång har varje modell tillstånd motsvarar URL dirigering [4].

2.2.3 Data lager

Webbapplikationer får sina data från REST-server där alla data lagras i någon form av databas såsom SQL eller NoSQL. Webbapplikationer hämtar data via HTTP begäran som sparas temporärt medan applikationen är igång. Därför existerar data lager för webbapplikationer både på klient- och serversida. Numera kan browser även spara data beständigt men för det mesta sparas data för en session [4].

2.3 Teknologier för utveckling av webbapplikationer

Detta kapitel presenteras det teknologier som används för utveckling av webbapplikationer.

2.3.1 HTML

HTML är ett påslag språk som används för att skapa hemsidor. Den möjliggör beskrivning av hur struktur på ett textbaserade dokument skall uppvisas på webbläsaren. Det inneslutar texten mellan HTML-taggar vilket definierar texten och beskriver dess utformning på sidan.Taggarna beskriver vilken del av texten är rubrik, vilken är paragraf och vilka delar är länkar osv. HTML-taggarna beskriver textens struktur till webbläsaren och webbläsaren använder det för att visa texten enligt det. Taggar är osynliga för användare när sidan visas upp i en webbläsaren.

HTML5 vilket är den nyaste version som introducerades år 2014. Med denna version möjliggör uppvisning av rörliga bilder och ljud utan tilläggsprogram som Flash. Andra teknologier som tillkommer med HTML5 är kanvas element som används för att rita grafer och animationer, urkopplad datalagring och dra och släpp funktioner [4].

2.3.2 CSS

CSS stilmall är ett språk som används för att skapa layout för webbsidor. De används för att definiera textstilar, tabell storlekar och andra beskrivande aspekter hos webbsidor som tidigare endast kan göras in i webbsidornas HTML-kod. CSS stilmall underlättar webbutvecklare att skapa en enhetlig stil över alla webbsidor i en webbplats. Istället för att definiera stil för alla textblock och tabeller i varje webbsida, kan stil som vanligen används definieras endast en gång

i ett CSS dokument. Stilmallen kan då användas till varje sidor som har en referens till CSS dokument filen [4].

2.3.3 Javascript

Javascript är ett dynamisk programmeringsspråk. Det är ett programmeringsspråk som vanligen används som den del av webbsidor, vars tillämpning möjliggör dynamisk interaktion mellan webbsidor och dess användare. Det är ett högnivå interpreterande språk som passar väl till objektorienterade och funktionell programmeringsstil. Javascript språket är dynamiskt typade, dynamiskt objekt med ett prototypbaserat objekt system. Språket stöder egenskaper såsom Inkapsling, polymorfism, mångvärding arv och komposition [5].

Fastän språket kompileras under körning i webbläsare är den mycket snabb. Javascript program är event driven baserad och är icke-blockerade. Javascript har på senaste åren även börja användas som ett programmeringsspråk på serversida. Detta är möjligt med introduktion av Node.js som är byggd på google V8 Javascript motor [6].

2.3.4 DOM

Document Object Model, oftast används förkortningen DOM är ett nödvändigt teknologi för att göra webbsidor interaktiv. DOM:en oftast kännetecknas som DOM-träd, vilket är ett träd av objekt
av knutar. Det är ett plattformsoberoende sätt att beskriva samling av objekt vilket utgör en webbsida i en webbläsare. DOM:en möjliggör skriptspråket Javascript att programmatiskt undersöka och ändra en webbsida. Med andra ord fungerar DOM:en som ett gränssnitt vilket med Javascript språket kan används för att manipulerar innehåll, struktur och stil av en webbsida.

Varje gång en webbsida utför en handling, såsom rotering av bilder i ett bildspel, uppvisa att ett fel har uppstått när en användare försöker sända in ett ogiltig form eller toggla mellan navigations meny, är det ett resultat av Javascript manipulering av DOM-trädet [4].

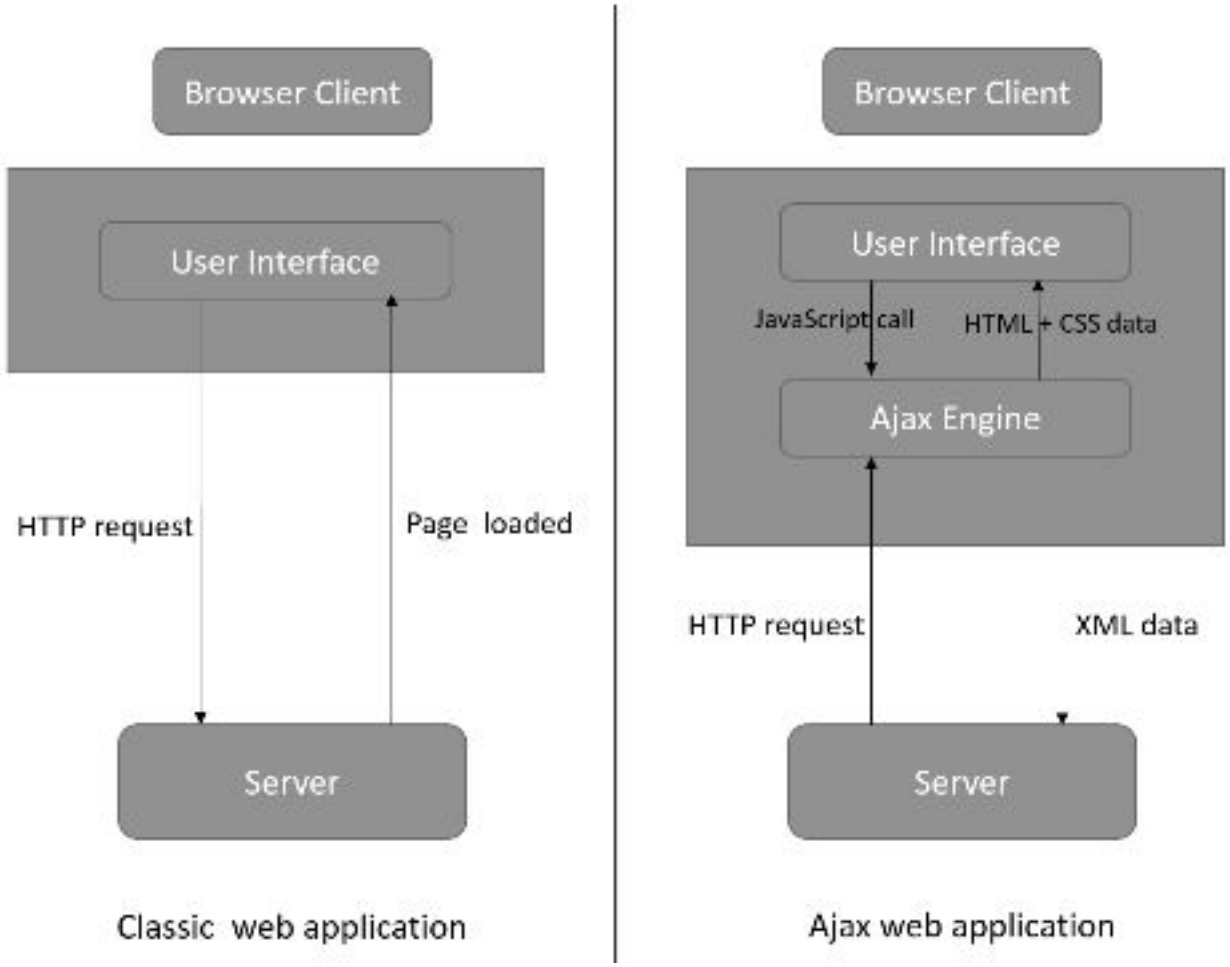
2.3.5 AJAX

Ajax är ett Javascript baserad teknologi som möjliggör att en webbsida kan hämta ny data och visa upp det utan hela sidan måste laddas om. Iden bakom AJAX är att göra webbsidor

snabbare och mera interaktiva för användare. Normalt måste hela webbsida laddas om för att se ny information. Till exempel när man fyller en form vid ett online köp med kreditkort, många webbsidor kräver att du måste trycka på nästa knappen förrän du kan verifiera att information du har fyllt i är rätt. Med AJAX dock är det möjligt för formen att informera dig ifall du har fyllt i information korrekt eller att du redan har gjort inköp redan.

Typisk exempel på användning av AJAX är att webbprogrammerare binder länkar, knappar eller andra typer av komponent som kräver användarinteraktion på en webbsida med Javascript kod. När användare aktiverar någon av det bundna komponenter kommunicerar Javascript koden med server och därefter tar emot data som kan varierar från HTML till andra format såsom JSON eller XML. Javascript koden använder då datan från responsen för att uppdatera webbsida utan att hela sidan måste laddas om.

AJAX tillåter webbutvecklare att skapa webbapplikationer som är enbart baserad på tillståndsändringar på klientsida istället för att det klassiska sättet att uppehålla session på server och renderar statiska webbsidor enligt dess tillstånd. Skillnaden kan se enligt bild 3.0. Fördelen med AJAX är den mängd av data som utbyts mellan server och klient drastiskt minskar och därmed förbättras webbsidor tillgänglighet.



Figur 3.0 klassisk webbapplikation och Ajax webbapplikation [7]

2.3.6 REST

REST definierar en uppsättning av arkitektonisk principer vilket möjliggör design av webbtjänster med fokus på data resurser och dess överföring över webben skall ske via HTTP. Varje system använder någon form resurs. Det kan varierar alltifrån bilder, webbsidor, video filer, information eller allt som kan representeras i ett datorbaserade system. REST webbtjänster huvuduppgift är att förse så att alla dess klienter har access till dessa resurser. REST passar väl till ensidesapplikationer för att det tillåter lös bindning mellan serversida och klient sida [4].

REST tjänst design är byggd med tanke på AJAX baserade webbapplikationer. Dess huvud kännetecken är att all dess tjänster använder enbart HTTP baserade metoder. HTTP GET, till exempel definieras som en metod för klientapplikation att hämta en resurs från webbserver. REST design etablerar en-till-en mappning av skapa, läsa, uppdatera och radera operationer med HTTP metoder [4].

- GET - hämta resurs från server
- READ - läsa resurs
- PUT - skapa ny resurs
- DELETE - raderar resurs

2.4 Webbapplikationer med programutveckling principer

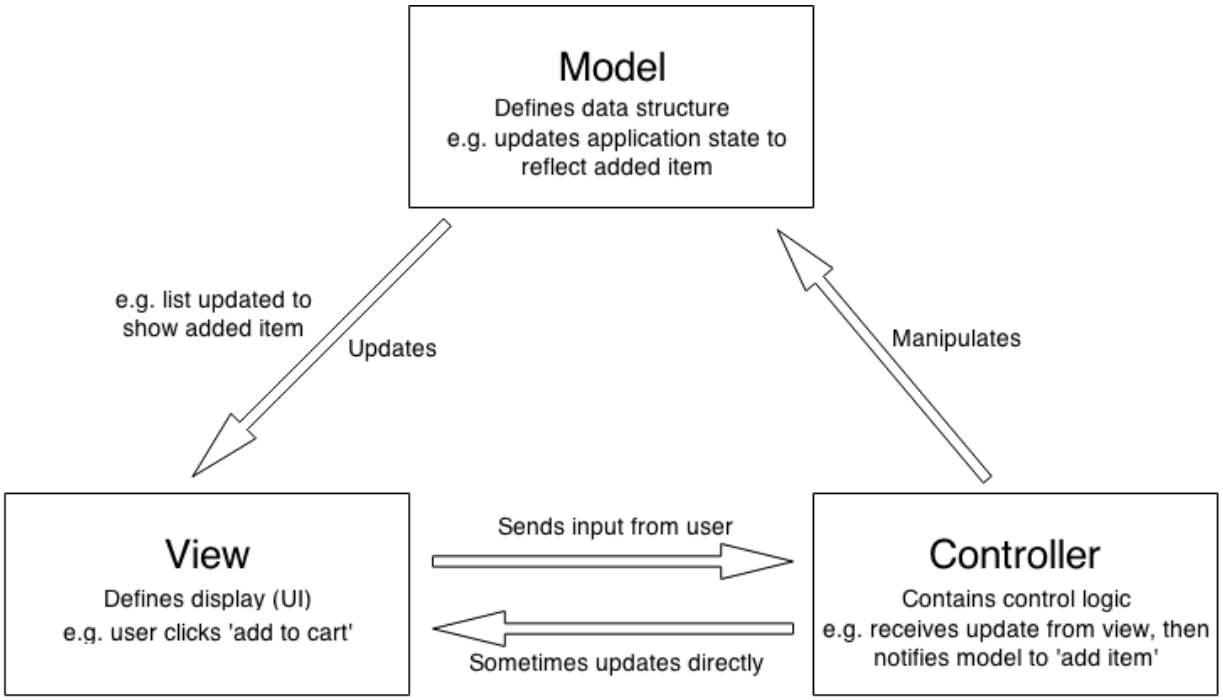
Webbutveckling har sedan dess ursprung att varit skapa webbsidor med dokument, bilder och form som innehåll. Behovet av programutvecklings principer har därmed inte behövs. Men med tillkomst av ny teknologi och behov av webbsidor med rikare användargränssnitt har denna inställning ändras. Då webbapplikationer växer, blivit bättre och snabbare har komplexitet också tillkommit. Stora webbapplikationer blir svårare att utveckla snabb, testa och kod underhållning blir mycket svårt i längden.

Uppdelning i problemområden (Separation of Concerns) är en programutveckling princip fokuserad på modularitet. Iden är att ett program uppdelas i många delar. Ändring av kod i en del skall kunna genomföras utan att det påverkar hela systemet. Detta gör att det blir lättare att identifiera fel och inför nya delar. (dijkstra)))))) MVC är en arkitektur design som främjar kod organisation med uppdelning i problemområde. Användning av MVC arkitektur har redan funnit i samband vanliga skrivbord applikationer men är ovanlig i samband med klientsida webbapplikationer skriven i Javascript för webbrowser.

2.4.1 MVC

MVC it arkitektur design mönster som delar upp dataprogram i tre olika distinkta modulära delar. Det tvingar separation av affärsdata (modell) från användargränssnitt(vy) och med en tredje komponent (kontroller) som sköter logiken och användar input enlig bild 4.0.

- Modell hanterar data för en applikation, vilket hämtat oftast från server. Data separated från användargränssnitt vy fastän det kan ses där och representerar det tillstånd applikationen är i.
- Vyn är användargränssnitt vilket användare ser och interagerar med. Det är dynamisk och genereras baserad på nuvarande modell av applikation.
- Kontroller är affärslogiken och presentationslager som sköter hämtning av data och hur data skall presenteras och vilka delar skall skall visas i vyn.



Figur 4.0 MVC modellen [4]

3 Webbramverk

Med tillkomst av webbapplikationer har också många webbramverk dykt upp under de senaste åren. Fastän det är möjlig att utveckla webbapplikationer utan dessa är det en kostsamt process.
Webbramverk i helhet en samling av komponenter designat för att underlätt utveckling av webbapplikationer. Alla är uppbyggda med programmeringsprinciper som bas och det flesta följer MVC arkitektur design mönstret. Iden är att tillåta webbutvecklare fokusera på viktiga detaljer och mål i ett projekt istället för att återuppfinna hjulet.

I detta kapitel kommer vi att gå igenom tre välkända webbramverk Angular, React och Redux.

3.1 AngularJS

Angularjs är ett webbramverk skapad och underhålls av Google. Det är ett mycket populär webbramverk som har gått igenom flera version iterationer sedan dess ursprung. Den här kandidat kommer att fokusera endast på version 1.x. Angularjs är ett klientsida fokus webbramverk för skapande av dynamiska webbapplikationer. Programmeringsspråk som används i Angularjs är Javascript och HTML. Angularjs är ett modulärt webbramverk där varje modul fungerar som en behållare för gruppering av olika delar av en applikation för lätt återanvändning. Ramverket använder sig av HTML som mall och utökar HTML ordförråd med egna vilket innehåller modelldata för dess applikation komponenter. Angularjs använder MVC arkitektur design mönster för alla dess webbapplikation design.

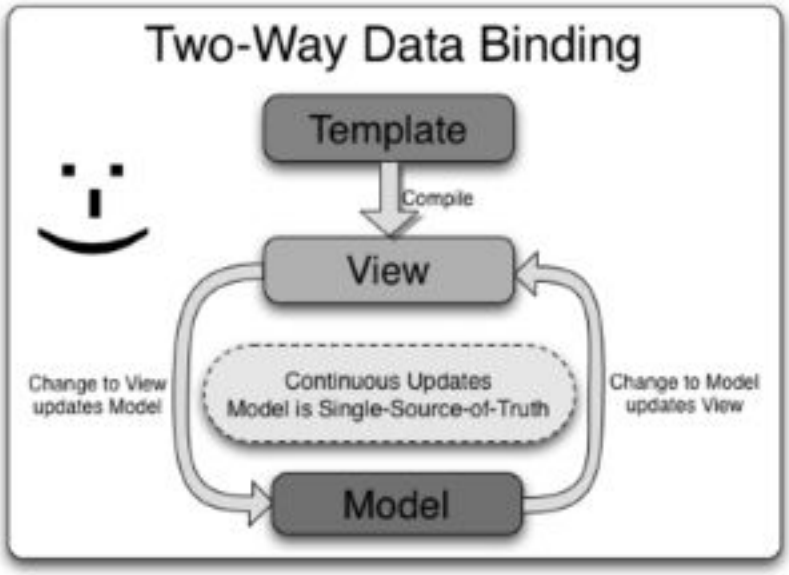
Detta kapitel skall gå igenom huvudkoncept för Angularjs och undersöka hur webbapplikationer byggd med Angularjs funkar.

3.1.1 AngularJS Kärnkoncept

Angularjs stöder för en deklarativ programmeringsparadigm där man deklarerar vad i HTML vad man vill åstadkomma. Det här görs med någonting AngularJS kallar för direktiv. Direktiv är i grunden en utökning av HTML ordförråd som fungerar som Angularjs:s egen typ av markör för DOM element. Före en webbapplikation byggd med AngularJS körs, kompileras den med HTML compiler där ersätts direktiv till DOM element med det funktioner som beskriv av direktivet. Direktiv används i mal som är vyn för en datamodell.

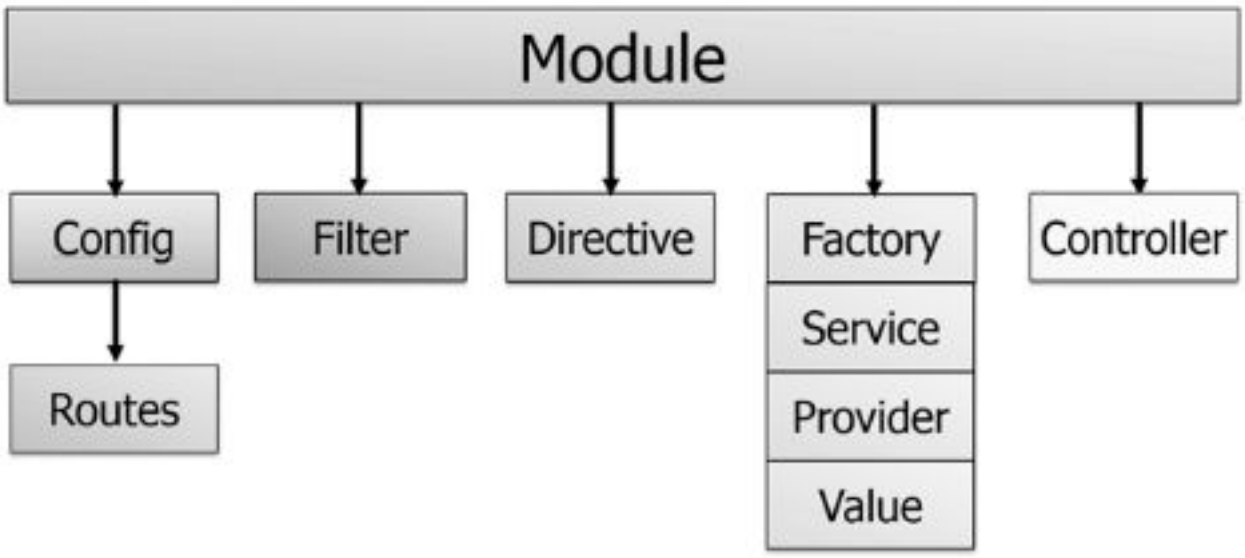
Webbapplikationer skriven med Angularjs är datadriven via databinding. Angularjs använder sig av en tvåvägsdatabindning enligt bild nedan 6.0. Där datamodellen och vyn är direkt

ihopkopplad. En ändring i vyn innebär att datamodellen uppdateras och ändring i datamodellen uppdateras vyn. Det här görs med Angularjs *$digest* loop som vi skall se på närmare på senare. Fastän Angularjs är ett MVC baserad designmönster är det i själva verket en annan variation av designmönstret oftast kallad för MVVM [8].



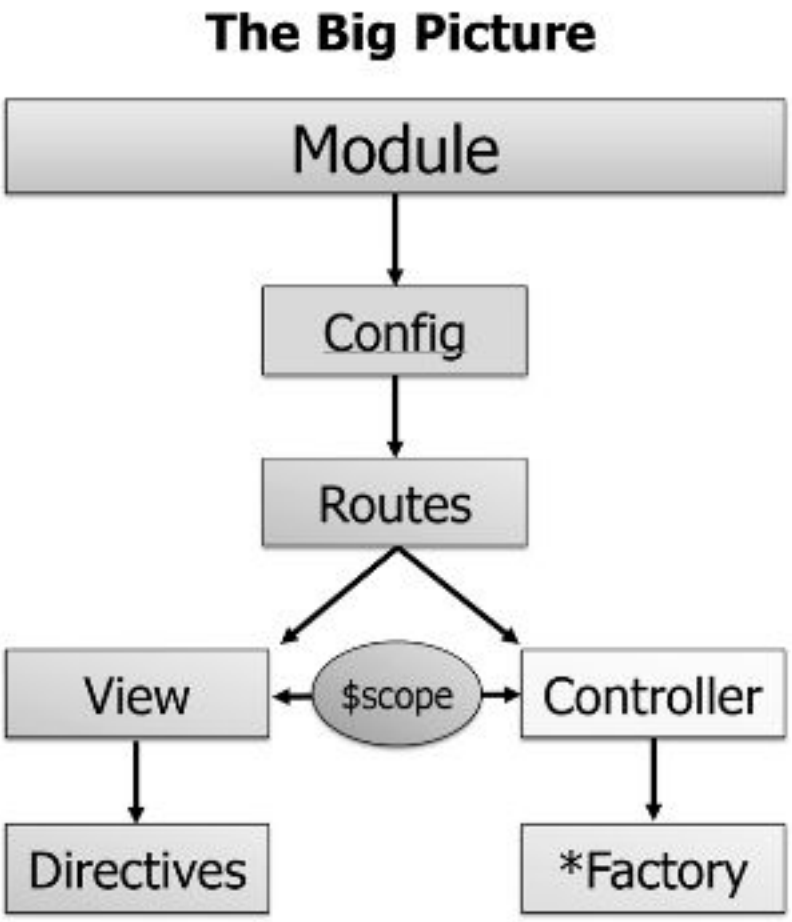
Figur 6.0 AngularJS tvåvägsdatabindning [8]

Som nämnts tidigare är Angularjs ett modulärt ramverk där alla inbyggd service komponenter som tillkommer med Angularjs är skapad för att användas med Dependency Injection. Dependency Injection är ett programmering mönster där istället för att skapa en ny instans in i en klass injiceras det utifrån. Iden är att det minimerar att kod limmas ihop och skapar komponenter på ett enhetligt och kontrollerat sätt. Detta gäller även de komponenter som är skapad av webbutvecklaren. Med Angularjs bryts webbapplikationen koden upp till många komponenter. Till exempel om vi skapar vår egen direktiv, kontroller eller mal, måste de separeras i sina egna filer. Vid användning måste det injiceras till det behövde komponent [8].



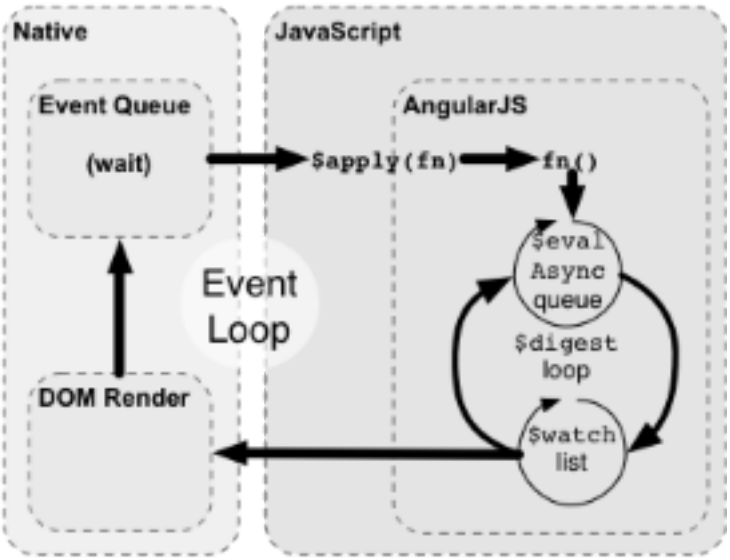
Figur 7.0 Angular module arkitekt [9]

Scope är ett objekt i Angularjs där varje applikation använder för att hänför till dess datamodell. Det förse två viktiga funktioner *$watch* och *$apply* som är nödvändiga för att databinding skall fungera. *$watch* observer datamodell för ändringar och *$apply* möjliggör att ändringar av datamodell utanför Angularjs område kan registreras. Scope binder ihop applikations vyn och kontroller. När Angularjs applikation körs registrerar dess Scope med *$watch*, då en ändring i datamodell sker meddelas direktiv om det av *$watch*. Både kontroller och direktiv har en referens till applikationens Scope men inte till varandra [8].



Figur 8.0 arkitektur för Angular Applikation [9]

Angularjs är ett mycket strikt webbramverk, allt måste göras på Angularjs sätt för att undvika problem. Exempel på detta är Javascript *setTimeout* funktion som inte kommer att fungera rätt ifall datamodell inte är bunden med *$apply* in i funktionen. Det är på grund av att ändring på datamodellen kommer att hända utanför Angularjs område. Bilden 9.0 beskriver arkitektur av Angularjs under huven. Förutom javascript egen event loop har Angularjs också sin egen. Varje datamodell måste registreras med till *$digest* loop med *$apply* för att Angularjs skall övervaka för ändringar. När en ändring sker kommer Angular att gå igenom $*watch* listan och uppdatera varje värde som ändras. Det görs flera gånger tills *$watch* inte längre upptäcker ändringar [8].



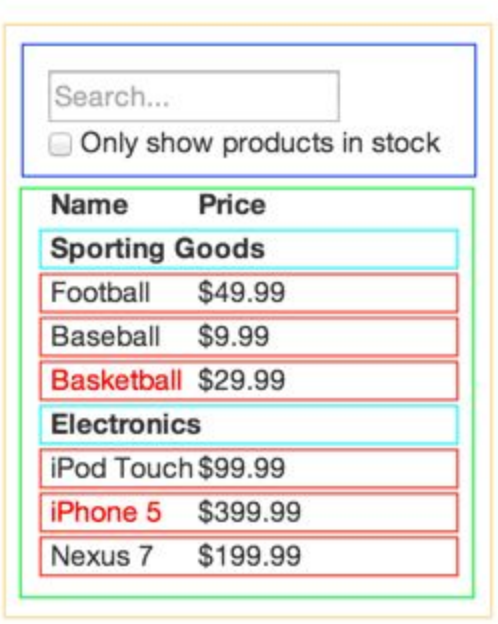
Figur 9.0 Angulars $digest loop [8]

3.2 React

React ett webbramverk skapat av Facebook, men använder en annan typ av arkitektur skillnad från AngularJS och andra MVC webbramverk. React är ett webbramverk med fokus på att bygga vy, webbramverket saknar stöd för datamodell och kontroll lager. För tillstånd och dataflöde hantering har Facebook lanserat Flux, men det bibliotek är inte lika populärt som React. Istället är det biblioteket Redux kapitel 3.3 baserad på Flux som har huvudval för många när det gäller dataflöde hantering för React. React är har en komponent baserat design men endast uppdateras när dess data uppdateras [10].

3.1.1 React Kärnkoncept

React av komponent design, där en stor komponent är uppbyggd av många små komponent. Exempel på detta är enligt bild nedan 10.0 på en produkttabell med sökfält. Varje färgade box representerar en komponent, men det är möjligt också att ha hela boxen som en stor komponent. I det här fallet är produkttabell en komponent som har två barn komponent, en sökfält komponent och en produktlist komponent. Produktlist komponent har produktkategori och produkt komponent. Det är möjligt att bryta upp de mindre komponenter vidare, iden är att utvecklaren skall själv bestämma komponentens storlek och ansvar [10].



Figur 10.0 produkttabell [10]

React separerar inte olika typer av teknologi (HTML, Javascript, CSS) såsom andra MVC webbramverk. Istället implementeras separation av problemområde (Separation of concern) och lös bindning (Loosely coupled) via Komponenter. Iden är att varje komponent representerar ett användargränssnitt och kan jämföras med en funktion som har en input och output. Komponent output är vad användare borde se på skärmen.

Komponent i React är en kombination av Javascript och HTML. React inför JSX en representation av HTML vilket möjliggörs användning av HTML i Javascript miljö. JSX i sig själv är endast syntaktisk socker som kompileras med Babeljs-kompilerare till vanlig Javascript kod. De två kodavsnittet nedan demonstrerar hur det funkar .

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Kodavsnitt 1.0 enkel hello world component

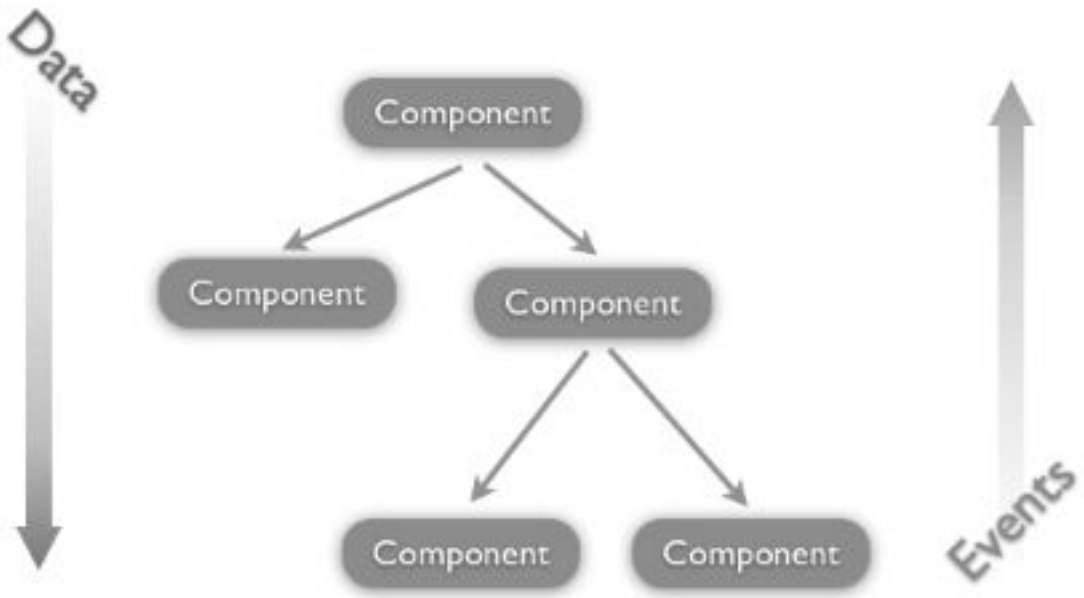
```
"use strict";
function Welcome(props) {
  return React.createElement(
      "h1",
      null,
      "Hello, ",
      props.name
```

```
  );
}
```
Kodavsnitt 2.0 enkel hello world after kompilation med Babeljs

Welcome() från kodavsnitt 1.0 är en enkel React komponent som tar en input och return en output av HTML header med en text “Hello” med namn från inputen props. När denna kod kompileras med Babeljs-kompilerare får vi som output kodavsnitt 2.0 där JSX delen har konverteras till en Javascript funktionsanrop för skapa en React element delar av JSX koden som parametrar.

React virtuell DOM är ett träd av noder såsom DOM på en hemsida. I React för varje DOM element som en komponent är uppbyggd av finns det en motsvarande representation i virtuell DOM. Där virtuell DOM:en har samma egenskaper som den riktiga. Vi kan se från tidigare kodavsnitt 2.0 exempel att outputen är en funktionsanrop *React.createElement*, detta för att bygga en virtuell DOM element. DOM manipulering i webbläsare är mycket långsamt på grund av att vid en ändring av nån element i DOM trädet kan utlösa omräkning av hela trädet. Virtuell DOM används för att omräkna trädet och sedan implementerar endast det ändrade elementet i den riktiga DOM:en.

Dataflöde i react är enkelriktad React komponenter kommunicerar med varandra via props input, från föräldrar komponent till deras barns komponent enligt bild 11.0. Iden är att alla tillstånd i en React applikation skall existera i på plats. Vilket gör att det är enklare att förstå och förutspå logiken för applikationen. Interaktion från en användare (till exempel från ett text input fält) kan inte direkt ändra en komponents tillstånd. Istället notifieras tillstånd källare om ändring via händelse, som sedan uppdaterar tillstånd ifråga. Därefter uppdateras komponenten med det nya tillståndet från källan [10].



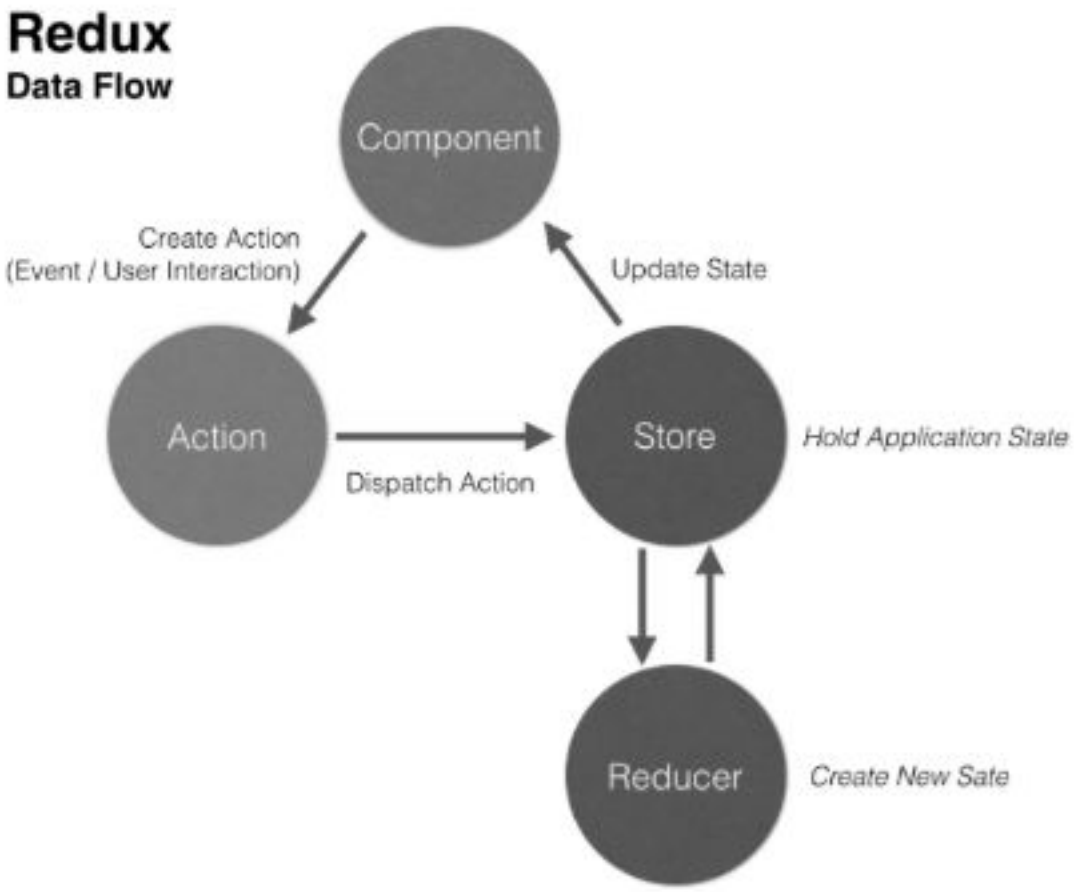
Figur 11.0 React dataflöde [11]

3.3 Redux

Redux bibliotek för för hantering av applikations tillstånd. Iden är att hela applikations tillstånd skall samlas i ett enda objekt. Redux skapades av Dan Abramov och är baserad på Facebooks Flux. Redux användning är inte begränsad till React men förknippas oftast med React.

Redux arkitektur är baserad på envägs dataflöde, där alla komponenter får sina data eller tillstånd från en enda källa. Det här är för att undvika att tillstånd är inte synkroniserad ifall det sparas i många olika ställe och blir svårt att avlusa. Redux implementation är baserade på följande koncept enlig bild 12.0 [13]:

- En källare: Redux förråd, är ett objekt som håller hela applikationens tillstånd. Olika tillstånd sparas som noder i objekt trädet.
- Redux förråd är skrivskyddad: Tillstånd i förrådet kan inte ändras direkt av en vy. För ändring krävs att det vy emittera en handling (action) som är en objekt som beskriver vad som skall ändras och vilken typ av ändring det handlar om. Denna ändring loggas av Redux som senare kan användas för avlusning och test.
- Redux reducer: har som uppgift att räkna ut vilken tillstånd som måste ändras och hur baserade på handling den tar emot.



Figur 12.0 Redux dataflödes arkitektur [12]

4 Diskussion

I kapitel 3 har vi gått igenom två webbramverk för att bygga webbapplikationer. Fastän dom är inte nödvändiga erbjuder dom många fördelar. Dessa ramverk kommer med standardfunktioner som är nödvändiga i webbapplikationer. Syftet man skall undvika att återuppfinna hjulet samt öka utvecklarens produktivitet.

Båda ramverk AngularJS och React har som fokus att underlätt konstruktion av webbapplikationer deras tillvägagångssätt varierar stort. AngularJS kan beskrivas som att det försöker sätta Javascript i HTML med direktiv medan React sätta HTML in i Javascript med JSX.
AngularJS kräver att man följer dess ide för vad som är en bra webbapplikation skall se ut och kräver att man följer dess vägledning annars uppstår lätt problem. React däremot försöker inte påtvinga hur lika mycket hur en ideal webbapplikation arkitektur skall se ut.

AngularJS tvåvägsdatabinding är mycket kraftig och förenklar byggandet av webbapplikationer men den är mycket krävande vad gäller prestanda i stora webbapplikationer, eftersom varje element måste ha en $watch för att upptäcka ändringar. React däremot fastän det är enklare och snabbare kräver kunskap av Redux.

Webbramverk är dock inte utan deras baksida. Fastän man spara tid på att använda webbramverk krävs det omfattande studie av ramverk förrän man kan påbörja att skapa ramverk. En annan är att utveckling av ramverk är också mycket snabbt. Exempel på detta är AngularJS som har gått från version 1.x till 2.x och nu 4.x på ett par år och har ändrat konceptuellt mycket mellan versionerna.

5 Referenser

[1] (2018.2.4) History of the Web hämtad från:
https://webfoundation.org/about/vision/history-of-the-web/

[2] (2018.2.4) Web Application Architechture – Client / Server Architecture hämtad från:
https://webingineer.wordpress.com/2010/07/24/web-application-architechture-client-server-architecture/

[3] (2018.2.4) PHP web application architecture/design hämtad från:
https://softwareengineering.stackexchange.com/questions/132275/php-web-application-architecture-design

[4] (2018.2.20) Web technology for developers hämtad från:
https://developer.mozilla.org/en-US/docs/Web

[5] (2008)Douglas Crockford: JavaScript: The Good Parts

[6] (2018.2.20) Node doc hämtad från:
https://nodejs.org/en/docs/

[7] (2018.2.21) ASP.NET MVC web application : Dynamic Content Load, using jQuery AJAX hämtad från:http://oniwebblog.blogspot.fi/2016/01/aspnet-mvc-web-application-dynamic.html

[8] (2018.3.11) Guide to AngularJS Documentation hämtad från:
https://docs.angularjs.org/guide

[9] (2018.3.11) Using an AngularJS Factory to Interact with a RESTful Service hämtad från:
https://weblogs.asp.net/dwahlin/using-an-angularjs-factory-to-interact-with-a-restful-service

[10] (2018.3.13) React doc hämtad från:
https://reactjs.org/docs/hello-world.html
[11] (2018.3.13) hämtad från:
https://medium.embengineering.com/in-react-js-data-flows-in-one-direction-from-parent-to-child-841103ed3aed

[12] (2018.3.13) Redux and React Native, simple login example flow hämtad från:
https://medium.com/@aurelie.lebec/redux-and-react-native-simple-login-example-flow-c4874cf91dde

[13] (2018.3.13) Redux doc hämtad från:
https://redux.js.org/