# **React From The Ground Up**

lärare: <http://davidtkatz.com/#/about>

kod: <https://github.com/15Dkatz/ReactJS_VideoGuides>

Plattform: Udemy [1]

### Intro

React är toppar listan, och har det gjort ett tag nu, på GitHubs hemsida. React ligger just nu på sjätte plats och visar tydligt att det är väl ändvänt idag. React skapades… har idag växt till en av de populäraste sätt att bygga en webbapplikation. och därmed bra är biblioteket bra underhållet. [2]

React är skapat och underhålls utav Facebook och deras utvecklare. Reacts bibliotek har blivit mycket populärt som med stor sannolikhet beror på att det funkar så bra hos Facebook. Men även stora företag som Netflix, Airbnb och Walmart stödjs utav React. [1]

### React, ES6 & Redux

React [3], ett JavaScript-bibliotek som huvudsakligen hjälper oss att dela upp vår applikation i flera komponenter som är huvudämnet i denna rapport. React är ett stort bibliotek som löser hur projektets filer kommer struktureras, hur projektets kod vanligtvis ska skrivas och som styr hur allt kommer visas för användaren. Själva React-koden skrivs och körs som JavaScript och koden följer ES6, den sjätte versionen av JavaScript som kom först i Juni 2015 [4].

Tillsammans med React kör man ofta ett annat bibliotek vid namn Redux[5]**.** Redux tillåter oss att bygga applikationer som kan växa. Redux är en bra kombination med React som gör att sättet att jobba blir och kopplingar mellan olika data blir tightare. [1]

ES6 [6] är det sjätte utgåvan och den så kallade versionen utan språket JavaScript. ES& står för ECMAScript 6 och släpptes först 2015. Det används av applikationer för att möjliggöra skript för kundsidan. Språk som JavaScript, JScript och ActionScript regleras av denna specifikation. Denna handledning introducerar dig till ES6-implementering i JavaScript. Att använda detta sättet att skriva kod när man jobbar med React kan starkt rekommenderas. [1

#### Golden React Rules

* Never mutate state directly. In order to change your state data, you can’t manipulate it like you would with normal variables.

“For example, something like `x++` or `counter += 1` would be fine in any other case. But in order to update state, you must use the **setState()**function and have new instances of data.”[1]

### Verktyg – att installera React

Man kommer behöva att installera en del verktyg och program för att kunna köra React och Redux tillsammans på sin plattform. Man behöver helt enkelt förbereda för att så smidigt som möjligt kunna konfigurera och jobba med React-projekt. Tack och lov finns det många nya sätt som hjälper oss med detta.

#### Node.js – [7]

Node låter utvecklare använda JavaScript för att skriva kommandoradsverktyg och för serverns sida kunna köra scripts för att dynamiskt producera webbsidans innehåll innan sidan skickas till användarens webbläsare.

Behövs endast installeras på din dator. Använd den senaste rekommenderade versionen för ett stabilt resultat. Skriv kommandot node -v i konsolen för att se om det funkar och du har rätt version. Skriv npm -v i terminalen för att kolla vilken version. Det ska vara en version som är högre än 3 för att kunna köras med Node. Npm kommer med Node, vanligtvis. Detta program hanterar de olika modulerna och paketen i Node.

Att köra Node JS tillsammans med React i detta projekt är nödvändigt för att koden ska vara exekverbar. Detta används främst i React då man behöver sätta upp en miljö att ta med sin JavaScript till. Med andra ord så skapar Node en lokal server till React där man kan hantera data via JavaScript.

#### Atom – [8]

Något att skriva kod i. I detta projekt kommer skrivas i Atom.

Efter installationen av Atom, om den är på en Windows plattform, så kan man använda atom . i konsolen för att starta upp atom i den projekt mappen man står för nuvarande i.

#### create-react-app – [9]

För att installera och få tillgång till ett enkelt sätt att skapa ett React projekt. Detta sättet att strata upp ett React-projekt kommer användas vid utveckling av applikationer mer React.

För att konfigurera paketet behöver man Node. Via Nodes pakethanterare installerar man via kommandot: *npm install -g create-react-app* installerar paketet på sin maskin.

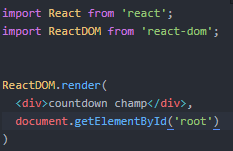
För sedan använda paketet initialitierar man ett React projekt via kommandot: *create-react-app ”namn-på-app”.*

Denna function skapar ett React-projekt. I projktet skapar installationen en viss struktur på projektet och tre olika tillhörande mappar. De olika mapparna heter och innehåller;

1. node\_modules – Med all React kod och alla andra packet.
2. public – index fil i html. Första som visas.
3. Src – Egna javascript och css filer. Where we HOST all the React code. det är I den man kommer jobba mest.

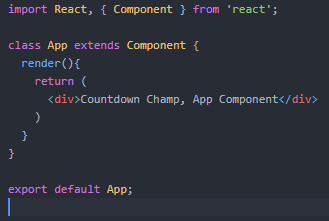
### Starta Projekt – Redigera din applikation

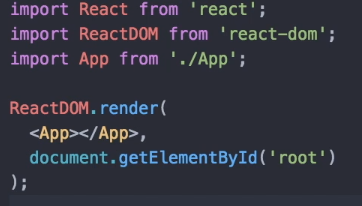
För att sedan starta upp miljön med Node. Alltså där man kommer köra själva servern och all JavaScript kod så använder man konsolen. Man går in i projektets mapp och skriver *npm start*.

En bra sak att börja efter du har skapat din React applikation med kommandot kan vara att ta bort all förinställd React kod. Detta genom att helt enkelt ta bort scr mappen. Detta för att enkelt kunna börja på ny kula genom att skapa en ny scr mapp, dom nu är tom på innehåll. I denna mappen skapar du sedan det fil som kommer köras först av all React kod, nämligen index.js. Den filen hämtar vart på index.html (den enda html-sidan) som React-koden ska skrivas och renderas.

#### Skapa en React komponent

I React jobbar man med att skapa sina egna komponenter som man sedan bygger upp hela sitt projekt med. Man försöker med detta skilja på olika delar av projektet, hålla reda på filer och skapa en bra struktur. Man får också lättare till att återanvända skriver kod.

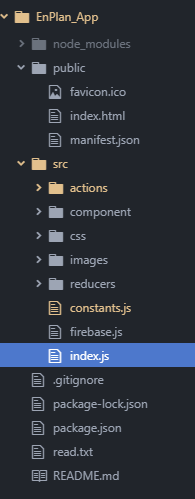
Komponenterna i detta projektet kommer skrivas i JavaScript förlängningen JSX. Som är en senare version av JavaScript och förenklar vissa delar av JavaScripts skrivandet. Men det är vanligt JavaScript med en förlängning. Från Reacts bibliotek importerar man *Component* som kommer användas för att bygga egna komponenter. För att bygga en egen så skapar du en klass av din komponent. Detta påminner objektorienterad programmering fast i JavaScript. Klassen ärver sedan egenskaper från React komponenten som kan bland annat rendera och returnera html till webbläsaren.

Viktigt att exportera komponenten sedan om det ska gå att använda den i andra filer av ditt projekt. Sedan behöver du bara importera komponenten i de filer där du ska använda den.

På bilden längst ner importeras alla nödvändiga paket sedan läggs den skapade applikationen ”App” till i DOM renderingen.

Här kan man även ta bort ”dubbel-taggen” från den skapade komponenten i DOM renderingen. Detta genom att endast skriva <App />.

#### Strukturen I en React applikation

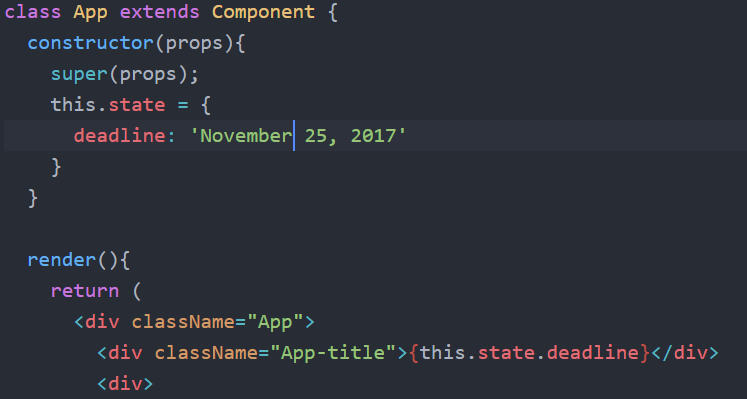
När man jobbar med React är det viktigt att veta vilka slags filer som ska ligga på vilka slags platser. För att React-koden ska kunna fungera rätt krävs det en viss struktur på mappstrukturen av projektet och det måste innehålla vissa filer. Som tidigare nämnt skapas det en viss struktur när kommandot create-react-app används. Detta resulterar i att projektet innehåller en viss mängd och typ av olika filer och mappar. Först kommer oftast mappen Node\_modules som man under projektet kan ta bort och sedan installera igen genom att skriva npm install. Mappen innehåller alla packet som projektet har installerat och innehåller delar som behövs för att det ska kunna köras. Mappen blir ofta rätt stor och kan därför vara bra att enkelt ta bort och lägga till för att projektet inte ska bli för stor plats.

I mappen public hittar man den index.html fil som visas när man kör applikationen. Den innehåller ingenting förutom id till det element där man anropar React.

I mappen src kommer alla filer med all JavaScript kod som kommer skrivas. Det är här React koden skrivs och allt som tillhör skapandet av själva mjukvaran i projektet. I denna mappen kommer man också lägga till CSS till projektet. CSS-filer importeras sedan i de olika JavaScript eller JSX filerna som ska ha egenskaperna. I React använder man className, istället för class som man gör vanligtvis,när man definierar vart i sin kod man ska använda klasser från CSS.

### State

State i React är något som kommer användas en del. I React syftar man oftast på användningen av återanvändningsbara, formbara och statiska komponenter. Man kan tänka att State är tillstånd på de uppgifter som är relevanta för applikationen. Där har varje komponent sitt eget lokala “State” med hänsyn till det globala tillståndet. Exempelvis en skola som har tillstånd av antalet lärare, antal elever, klassrum mm. Men varje elev har också en egen tillstånd, ett antal läroböcker och pennor mm.

I React beskriver vi användargränssnitt med komponenter. Du kan tänka på komponenter som enkla funktioner (i något programmeringsspråk). Vi kallar funktioner med lite inmatning och de ger oss lite resultat. Vi kan återanvända funktioner efter behov och komponera större funktioner från mindre.

Komponenterna är exakt samma; vi kallar deras inmatning "egenskaper" och "tillstånd", och en komponentutgång är en beskrivning av ett användargränssnitt (vilket liknar HTML för webbläsare). Vi kan återanvända en enskild komponent i flera användargränssnitt och komponenter kan innehålla andra komponenter.

Till skillnad från rena funktioner kan en fullständig React-komponent ha en privat stat för att hålla data som kan förändras över tiden. För att lägga till ett State till komponenten lägger man till constructor. I constructorna ska man deklarera vad för slags state det är till komponenten. I React är State alltid ett objekt som skickas. Sedan hämtar man sitt objekt i renderingen via {this.state.valt state.}

#### Uppdatera State

För att uppdatera vårat State eller tillstånd dynamiskt i våran applikation kan man ta fördel av en specifik metod i React. Denna metod kallas för setState. Man måste kalla på this.setState genom koden för att ändra eller uppdatera på lokala klassers data. Detta måste ske genom att man skapar en valfri metod eller funktion som ska uppdatera State åt komponenten.

* ”You must never mutate or change state directly”

Betyder egentligen att komponenten inte kommer att läsas rätt om vi muterar direkt. Istället för i en funktion eller metod. Måste ske genom setState() 🡪 En metod.

För att kalla på en skapad metod i vår knapp använder man onClick i renderingen. I onClicken skapar man en ES6 funktion där man lägger in this.metoden. Med ES6 så kommer den köras en gång och rätt.

Reacts namn är den enkla förklaringen till detta koncept. När tillståndet för en komponent (inmatningen) ändras ändras också användargränssnittet (utmatningen). Denna ändring i beskrivningen av användargränssnittet måste återspeglas i den enhet vi jobbar med.

I en webbläsare behöver vi regenerera HTML-vyerna i Document Object Model (DOM). Med React behöver vi inte oroa oss för hur vi ska reflektera dessa förändringar, eller till och med hantera när vi ska ändra ändringar i webbläsaren. React kommer helt enkelt att reagera på tillståndsändringar och uppdatera automatiskt DOM när det behövs.

### Props

Props refererar till vissa data inom applikationen. Med Props skickar du data eller tillstånd till barn-komponenter från moder-komponenter. Detta är betydande o kan ange en tidsfrist inom komponenten i renderingsmetoden.

### Lifecycle methods

1. componentWillMount() 🡪 Detta är en funktion i React som kan användas när man har en komponent. Denna speciella med denna funktionen är att den körs före komponenten har blivit färdigt renderad till komponenten som den renderas till. Bra för att hålla koll på hur State förändras. Detta ingår i React biblioteket.
2. componentDidMount() 🡪 Likndande den förra så är detta också en funktion i Reacts bibliotek. Till skillnad från den förra så körs denna funktionen direkt efter en React komponent har renderats.

### Adding bootstrap to our App – [10]

Att lägga till en CSS ramverk när man jobbar med design och gränssnitt på en applikation kan vara smart. Detta kan man installera direkt in i sitt React projekt och ta användning av designade komponenter speciellt utvecklat för React. Bootstrap installerar man till projektet via Nodes pakethanterare npm. I konsolen skriver man *npm install react-bootstrap –save* när man står i mappen till projektet.

1. <!-- Latest compiled and minified CSS to our html -->

**<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/cs**s/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

## Outro

Här är en översikt över de viktiga begreppen som du just lärt dig:

* Reaktorkomponenter - återanvändbara och oberoende delar av React-kod som innehåller användargränssnittet.
* ES6 import och export - en ny syntax för delning av kod mellan separata filer. Används i fall som "importreakt från" reagera "
* Ange - relevanta uppgifter till en ansökan. Varje komponent har sitt lokala tillstånd så länge du förklarar lägger konstruktorn till en komponent och deklarerar sitt statobjekt.
* Uppdateringsstatus - Vid uppdatering av tillstånd, se till att aldrig mutera staten direkt. Om du gör det leder det till dödliga fel i din ansökan. Istället, omklara nya instanser av statliga arrayer eller objekt och använd funktionen setState () för att uppdatera tillstånd.
* Rekvisita - liknande staten, förutom att dessa data ärar från föräldrakomponenter som anger statliga delar som egenskaper.
* LifeCycle Methods - händelser i React-komponenter som utlöser i fall som att göra på eller av skärmen eller under statliga uppdateringar. Ett exempel är komponenten `componentDidMount ()` lifeCycle.

## Källförteckning

[1] – Udemy Course

[2] – <https://github.com/trending/javascript>

[3] - React - <https://reactjs.org/>

[4] - React: Making faster, smoother UIs for data-driven Web apps, Paul Krill, 2014 -https://www.infoworld.com/article/2608181/javascript/react--making-faster--smoother-uis-for-data-driven-web-apps.html

[5] – Redux - <https://redux.js.org/>

[6] – ES6, <https://es6.io/>

[7] – Node JS, <https://nodejs.org/en/>

[8] – Atom, <https://atom.io/>

[9] – React create App

[10] - <https://react-bootstrap.github.io/>

[11] - <https://react-bootstrap.github.io/getting-started/introduction>