React, Angular, Vue.js - en jämförelse och analys mellan de tre främsta ramverken för webbapplikationer

Henrik Martola 42003

Kandidatavhandling i datateknik

Handledare: Dragos Truscan

Fakulteten för naturvetenskap och teknik

Åbo Akademi

2021

Abstrakt

Innehållsförteckning

1 Inledning

2 Allmänt om webbramverk

Ramverk, och mer specifikt webbramverk, används för att underlätta utvecklarnas arbete och för att göra det enklare att starta ett nytt projekt då ramverket sköter den underliggande konfigurationen. Att skapa ett nytt projekt med ett webbramverk kan vara så enkelt som att köra ett kommando via kommandoraden (eng. command line interface, CLI), som ger en färdig webbsida redo för vidare utveckling.

2.1 Definition

Ett ramverk i mjukvarukontext kan definieras som ett större programbibliotek med specifika regler och sätt för utveckling av mjukvara. Det är ett mångsidigt verktyg som erbjuder skalabilitet och stabilitet genom att endast tillåta ett visst sätt för utvecklingen av programvara. Varje ramverk har sina egna styrkor och svagheter med specifika designmönster som bör tas i beaktande då man väljer det lämpliga verktyget för projektet. [1]

Vid sidan om ramverk finns programbibliotek, vars syfte är att stöda utvecklare lösa specifika problem som andra utvecklare råkat ut för. Programbibliotek är verktyg och dikterar inte hur ett program uppbyggs, utan används för att spara tid och resurser. Om det finns ett programbibliotek som löser ett problem är det ofta onödigt att själv försöka implementera en lösning. [1]

2.2 Webbramverkens betydelse

Med hjälp av de moderna webbramverken har det blivit mycket enklare att skapa komplexa, interaktiva webbsidor. Applikationer som tidigare endast kunde användas som nativa program på datorn kan nu även köras på webbläsare, vilket har gett upphov till termen webbapplikation. Majoriteten av dagens webbapplikationer baserar sig på något webbramverk, såsom Ember, Angular och React. [1]

Webbramverkens stora inverkan inom webbutveckling kan förklaras med hur arbetsamt det är att bygga och framställa HTML-element inom dokumentobjektmodellen (eng. Document Object Model, DOM). Applikationer måste sköta om data, ofta referat som *tillstånd* (eng. state), och framställa varje ändring i tillståndet för användaren. Dessa ändringar är betydligt mer enkla att göra med webbramverk via de inbyggda funktionerna som sköter om dokumentobjektmodellen. [1]

Webbramverk har även starkt stöd från utvecklarsamhällen som deltar i att förbättra ramverken med programbibliotek. React och Angular utvecklades ursprungligen som interna verktyg hos Facebook och Google, men har sedan dess blivit mer mångsidiga tack vare det aktiva samhället. Det finns ett stort utbud på externa programbibliotek som erbjuder ny funktionalitet eller nya verktyg för utvecklare. [1]

2.3 Programspråk och syntax

Webbsidor baserar sig på HTML och JavaScript, där HTML som märkspråk bestämmer hur webbsidan ser ut med hjälp av olika element (kallas även för taggar). JavaScript används för webbsidans logik och funktionalitet och klassificeras som ett programmeringsspråk. Webbramverk använder sig av olika former av HTML och JavaScript som kallas för domänspecifika språk (eng. DSL, domain-specific language). Domänspecifika språk gör det lättare att utveckla program inom specifika områden

genom att erbjuda en syntax som är designad för det specifika området. Det är även möjligt att bygga applikationer utan domänspecifika språk, men det är något som inte rekommenderas. [1]

React använder sig av något som kallas för JSX, ett domänspecifikt språk som blandar ihop HTML och JavaScript. Med JSX är det således möjligt att skriva HTML inom JavaScript och vice versa. Angular baserar sig på TypeScript (figur 2), som är en slags variation av JavaScript med tillsatt statisk typkontroll. All TypeScript-kod transkompileras till vanlig JavaScript då det körs. [2] Vue använder sig av en HTML-baserad syntax (figur 3), men stöder också användningen av JSX. [3] De olika domänspecifika språken kommer att behandlas mer detaljerat i sina respektive kapitel.

2.4 Node.js

Innan de själva webbramverken behandlas noggrannare är det värt att kort berätta om JavaScript-exekveringsmiljön *Node.js*, som är byggd ovanpå Google Chromes V8-motor. Node.js används på serversidan och gör det möjligt att skriva serversidans kod i JavaScript, således kombinerande klient -och serversidans kod till samma språk. [4] Node.js förhindrar också koden från att hamna i ett dödläge (eng. deadlock) med sitt asynkrona beteende, vilket stöder utvecklingen av skalbara applikationer. [5]

Node.js erbjuder också en pakethanterare som kallas för *npm* (kort för *node package manager*). Pakethanteraren npm tillåter utvecklare att ladda ner, publicera och hantera projektets programbibliotek via kommandoraden. Bakom npm finns ett stort och aktivt samhälle av programmerare som delar med sig programbibliotek för alla ändamål. [6]

React, Angular och Vue använder alla Node.js för ett JavaScript-bakände (serversida) och npm för att ladda ner och hantera olika programbibliotek. Node.js erbjuder också en hel del olika ramverk för utvecklare att välja mellan, där ett av de mest populära är Express, ett minimalistiskt ramverk för att hantera webbuppkopplingar. [7]

3 React

React är officiellt ett komponentbaserat JavaScript-programbibliotek för att skapa lätta, interaktiva användargränssnitt. [8] Jämfört med de andra webbramverken har React ingen inbyggd funktionalitet för triviala egenskaper såsom dirigering och global tillståndshantering, utan dessa sköts av s.k. kamratprogrambibliotek (eng. companion library). [9] Trots detta används React för samma syften som de andra webbramverken och med de kompletterande programbiblioteken kan React även tolkas som ett webbramverk. För att undvika några missförstånd behandlas React i denna avhandling som ett webbramverk.

I februari 2019 introducerade React som en del av version 16.8 ny funktionalitet som kallas för *hooks*. Hooks är kort förklarat ett nytt sätt att hantera tillstånd samt livstidsegenskaper inom funktionella komponenter utan att behöva skriva klasser. [10] All React som gås igenom i de kommande kapitlen baserar sig på de nya hooks istället för klasser, eftersom användningen av hooks är rekommenderat sedan version 16.8.

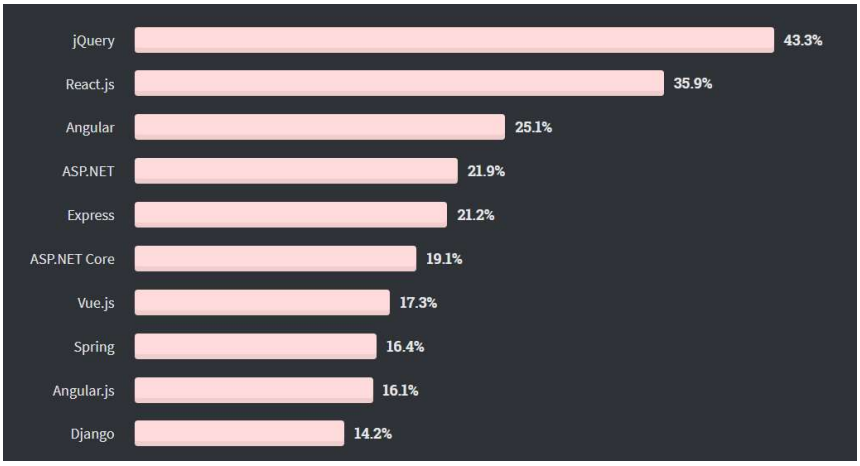
3.1 Bakgrund

Jordan Walke, en mjukvaruutvecklare hos Facebook, utvecklade år 2011 en tidig prototyp av React som kallades för FaxJS. Facebook mötte problem med skalbarhet inom deras Facebook Ads-service vilket ledde till vidare utveckling av FaxJS, med första versionen av React som blev introducerad år 2012. Under samma år skiftades också Facebooks fokus till att förvandla React som öppen källkod, eftersom den nyligen uppköpta Instagram hade uttryckt intresse för React. [11]

År 2013 publicerades React för världen med en hel del kritik runtom sig. Trots att React marknadsförde sig själv som det bästa alternativet var det många som förhöll sig

skeptiska. Med tiden minskade dock de skeptiska rösterna, och tre år senare hade React etablerat sig inom industrin som ett av de främsta alternativen, med företag som Netflix och Airbnb som användare. [11]

Vid stunden av skrivandet, 12 mars 2021, har React varit ute i nästan 8 år. Den senaste versionen, v17.0.1, publicerades i oktober 2020. Figur 2 visar tydligt hur React har under de 8 åren stigit till ett av de ledande webbramverken med 35,9% av svararen som använder React. JavaScript-programbiblioteket jQuery är fortfarande den populäraste med 43,3% andel, men har nu stadigt tappat popularitet under de senaste åren.



Figur 1: "Skärmdump på de 10 mest använda webbramverken bland 42 279 utvecklare tagen från Stackoverflows senaste kartläggning (Stack Overflow Developer Survey 2020)." [12]

3.2 Komponenter och JSX

Som det redan nämndes i kapitel 2.3 använder sig React av det domänspecifika språket JSX som utvidgar syntaxen för vanlig JavaScript. Istället för att skriva HTML och JavaScript skilt i sina egna filer är idén med JSX att skriva dem ihop till något som kallas för *komponenter*. I React är komponenter självständiga och återanvändbara delar, som kan ta emot argument och returnera det som bör synas i användargränssnittet. [13]

Det finns olika typer av komponenter i React, varav de två viktigaste är klasskomponenter samt funktionskomponenter. Före hooks blev introducerat kunde tillstånd endast användas i klasskomponenter, vilket gav funktionskomponenter deras dåtida benämning *funktionella tillståndsfria komponenter*. Klasskomponenter uttrycks således som klasser med konstruktörer, medan funktionella komponenter uttrycks endast som funktioner. Då hooks senare introducerades i version 16.8 blev det möjligt att använda tillstånd inom funktionskomponenter istället för klasskomponenter, en ändring som har lett till att många utvecklare inte längre använder sig av klasskomponenter i sina nya React-applikationer. [14]

Figur 2 demonstrerar ett enkelt exempel på en funktionskomponent som returnerar JSX. Det är möjligt att uttrycka funktionskomponenter på två olika sätt i React, antingen med nyckelordet *function* (ex. function HelloWorld()) eller som en pilfunktion (eng. arrow function, se figur 2). Pilfunktioner introducerades i ES6, den nya versionen av JavaScript som publicerades år 2015 och som införde en hel del ny funktionalitet till syntaxen. Variabeln *const* uttrycker en konstant, det vill säga ett värde som inte kan ändras, och används således för funktionsuttryck eftersom funktioner alltid är konstanta. [15]

Själva komponenten består av JavaScript och JSX, där JavaScript används utanför returuttrycket och JSX innanför returuttrycket. I figur 2 skapas den konstanta variabeln *greeting,* som sedan anropas inuti klammerparenteser med instruktioner för hur användargränssnittet bör visa upp elementet. JSX är med andra ord JavaScript-uttryck som skrivs inuti HTML-taggar, vilket i sin tur skapar vad som kallas för React-element. Klammerparenteser godkänner alla giltiga JavaScript-uttryck, inklusive funktionsanrop. [16]

Komponenter hänvisas till på samma sätt som taggar inom dokumentobjektmodellen, det vill säga med vinkelparenteser. Att hänvisa till komponenten i figur 2 skulle således se ut som följande:

<HelloWorld /> eller *<HelloWorld></HelloWorld>*

```
const HelloWorld = () => {
  const greeting = 'Hello World!'
  return (
    <div>
      <h1>{greeting}</h1>
    </div>
  )
}
```

Figur 2: "Exempel på en komponent i React."

3.3 Hooks

Hooks introducerades i februari 2019 för att lösa flera olika problem som React hade inom tillståndsbaserad logik. Kapitel 3.2 nämnde hur tillstånd kunde förut endast användas i klasskomponenter, vilket i större applikationer ledde till invecklade och uppsvällda klasskomponentstrukturer eftersom det inte gick att uppdela tillståndsrelaterad logik i mindre delar. Det var också svårt att återanvända tillståndsbaserad logik mellan komponenter, eftersom all användning av samma logik krävde komponentens omstrukturering. För det sista såg nybörjare klasser i React som ett svårt koncept att förstå, vilket skapade krav på en ny lösning. [17]

Hooks är en samling av JavaScript-funktioner som tillåter användningen av Reacts kärnegenskaper (tillstånd, livscykel, kontext) genom funktionsanrop. Hooks fungerar endast i funktionskomponenter, och kan således inte utnyttjas i klasskomponenter. Det finns olika inbyggda hooks, men de två viktigaste är *useState* och *useEffect*. [18]

Tillståndshooken useState deklarerar en tillståndsvariabel som returnerar tillståndet och en funktion för att uppdatera tillståndet, där tillståndet bevaras längs programmets exekvering. Då useState initialiseras ges typen av data som ett argument, vilket kan exempelvis vara ett heltal, en teckensträng, ett booleskt värde eller en lista. [19]

Livscykelhooken useEffect används för att utföra någon specifik uppgift efter varje framställning (eng. render), såsom att uppdatera dokumentobjektmodellen eller att hämta data. Som förval körs useEffect efter varje framställning, men det är även möjligt att använda useEffect konditionellt, det vill säga endast då ett visst värde ändras. [20]

```
const [value, setValue] = useState(0)

useEffect(() => {
  // do something
})
```

Figur 1: "Exempel på useState och useEffect. Tillståndsvariabeln godkänner endast siffror eftersom den har skapats med ett heltal som argument."

3.4 Kompletterande programbibliotek för React

Eftersom React egentligen är ett programbibliotek för användargränssnitt saknar det mycket av den funktionalitet som de andra webbramverken erbjuder, där bland de viktigaste är URL-dirigering och global tillståndshantering. För att lösa detta har flera olika externa programbibliotek introducerats för att fylla i den saknande funktionaliteten, således förvandlande React till en mer komplett helhet.

Exempel på några av de mest populära kompletterande programbiblioteken är React Router för URL-dirigering, Redux för bättre tillståndshantering och React Bootstrap för följsam webbdesign. Vissa av programbiblioteken har sin bakgrund hos företag eller organisationer, medan andra har blivit skapade av privatpersoner. Gemensamt med dessa är den öppna källkoden och den öppna atmosfären där alla kan bidra och förbättra koden. Utan de kompletterande programbiblioteken skulle React inte njuta av den popularitet som den nu gör.

4 Angular

```
const HelloWorld = (name: string) => {
  return 'Hello, ' + name
}
let greeting = 'World'
document.body.textContent = HelloWorld(greeting)
```

Figur 3: "Hello World exempel med TypeScript"

4.1 Bakgrund

5 Vue

```
<template>
  <div>
    <h1>{{ msg }}</h1>
  </div>
</template>

<template>
  <HelloWorld msg="Hello World!" />
</template>
```

Figur 4: "Hello World i Vue."

5.1 Bakgrund

Källhänvisning

[1] MDN , ”Introduction to client-side frameworks,” 19 February 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction. [Använd 18 February 2021].

[2] TypeScript docs, ”TypeScript for the New Programmer,” Microsoft, 10 March 2021. [Online]. Available: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html. [Använd 10 March 2021].

[3] Vue.js, ”Template Syntax,” Vue.js, 8 Januari 2021. [Online]. Available: https://v3.vuejs.org/guide/template-syntax.html. [Använd 10 Mars 2021].

[4] OpenJS Foundation, ”Introduction to Node.js,” [Online]. Available: https://nodejs.dev/learn. [Använd 29 Mars 2021].

[5] OpenJS Foundation, ”About Node.js,” 8 Januari 2020. [Online]. Available: https://nodejs.org/en/about/. [Använd 29 Mars 2021].

[6] npm, Inc, ”About npm,” [Online]. Available: https://www.npmjs.com/about. [Använd 29 Mars 2021].

[7] expressjs, ”Express,” OpenJS Foundation, 2017. [Online]. Available: https://expressjs.com/. [Använd 30 Mars 2021].

[8] Facebook Inc, ”React,” 2021. [Online]. Available: https://reactjs.org/. [Använd 11 Mars 2021].

[9] Vue.js, ”Comparison with Other Frameworks,” [Online]. Available: https://vuejs.org/v2/guide/comparison.html#React. [Använd 11 Mars 2021].

[10] Facebook Inc, ”Introducing Hooks,” 2021. [Online]. Available: https://reactjs.org/docs/hooks-intro.html. [Använd 11 Mars 2021].

[11] F. Hámori, ”The History of React.js on a Timeline,” RisingStack, 4 April 2018. [Online]. Available: https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/. [Använd 12 Mars 2021].

[12] Stack Overflow, ”2020 Developer Survey,” 2020. [Online]. Available: https://insights.stackoverflow.com/survey/2020#technology-web-frameworks. [Använd 12 Mars 2021].

[13] Facebook Inc, ”Components and Props,” 2021. [Online]. Available: https://reactjs.org/docs/components-and-props.html. [Använd 16 Mars 2021].

[14] R. Wieruch, ”Types of React Components,” 12 Mars 2019. [Online]. Available: https://www.robinwieruch.de/react-component-types. [Använd 19 Mars 2021].

[15] w3schools.com, ”ECMAScript 2015 - ES6,” [Online]. Available: https://www.w3schools.com/js/js_es6.asp. [Använd 19 Mars 2021].

[16] Facebook Inc, ”Introducing JSX,” 2021. [Online]. Available: https://reactjs.org/docs/introducing-jsx.html. [Använd 20 Mars 2021].

[17] Facebook Inc, ”Introducing Hooks,” 2021. [Online]. Available: https://reactjs.org/docs/hooks-intro.html. [Använd 23 Mars 2021].

[18] D. Abramov, ”Making Sense of React Hooks,” 2 November 2018. [Online]. Available: https://dev.to/dan_abramov/making-sense-of-react-hooks-2eib. [Använd 23 Mars 2021].

[19] Facebook Inc, ”Using the State Hook,” 2021. [Online]. Available: https://reactjs.org/docs/hooks-state.html. [Använd 23 Mars 2021].

[20] Facebook Inc, ”Using the Effect Hook,” 2021. [Online]. Available: https://reactjs.org/docs/hooks-effect.html. [Använd 24 Mars 2021].