Presentation av Lodash.js + proof of concept

FED19G

JavaScript 2
2020-02-24

Skapad av: Elin Alm
Examinator: David Jensen

Inledning
Lodash är ett referensbibliotek gjort med JavaScript. Det är uppföljaren till underscore.js. Lodash används främst för att göra hanteringen av objekt enklare i och med att det erbjuder många metoder för det. Javascriptkod blir enklare i och med att det tar ut krånglet i koden. Lodashs modulära metoder är bra för itererande listor, objekthantering och stränghantering, manipulering och testning av värden och för att skapa kompositfunktioner (Lodash.com, u.å). 2015 var Lodash det mest nedladdade referensbiblioteket i npm (Dwyer 2015).

Bakgrund
Som sagt så är Lodash en uppföljare från underscore.js. Underscore.js släpptes tidigt under 2009 av Jeremy Ashkenas och blev direkt ett populärt verktygsbibliotek för JavaScript tillsammans med Backbone och Angular. Underscore erbjuder över 100 funktioner som hjälper att manipulera listor, objekt och annat (Hanchett 2017). Inom ett par år började utvecklingen av underscore stanna av och det var då projektet Lodash tog sin form. Detta var då år 2012. Med Lodash kunde man lägga till många fler hjälpfunktioner och metoder än det som redan fanns tillgängligt ute på marknaden. Underscore.js finns fortfarande men Lodash har en betydligt högre användarskara på grund av dess pågående utveckling och dess överlägsna uppsättning av funktioner (Morera 2018). John-David Dalton heter skaparen av Lodash. Han skapade biblioteket för att ha ett mer konsekvent iterationsstöd för olika miljöer, exempelvis strängar för strängar och objekt (Hanchett 2017).

När är det lämpligt att använda sig av Lodash?
Lodash består av många filer av javascriptkod som är uppdelat i funktioner. Användare kan helt fritt använda sig av koden och det kan delas in i 7 kategorier. Det första är verktyg som simplifierar vanligt återkommande uppgifter så som att fastställa datatyper eller att göra simpla matematiska beräkningar. Den andra kategorin är funktioner som förenklar uppbyggnaden och exekveringen av bland annat ”debouncing” och ”throttling”. Debouncing och throttling är två tekniker som kontrollerar hur många gånger vi tillåter en funktion att köras över tid. Att ha en debounced eller throttled version av en funktion är användbart när vi fäster funktionen i ett DOM-event. Det hjälper oss att ta kontroll över vad som händer mellan eventet och funktionen som körs (Corbacho 2018). När vi använder Throttle tillåter vi inte vår funktion att köras mer än en gång varje X millisekund. När vi använder Debounce grupperas sekventiella anrop till ett och samma anrop (Corbacho 2018). Den tredje kategorin är stränghanteringen som ger tillgång till funktioner för att kunna göra det mesta inom stränghantering så som att kunna hantera gemener och versaler och även kunna dela upp strängar till substrängar. Den fjärde kategorin är listhantering som gör det möjligt att kunna kombinera, skapa, splittra, ändra och komprimera listor. I bilaga 1 och 2 finns två kodexempel som visar de fördelarna av bland annat att komprimera koden när det kommer till listhanering. Den femte kategorin handlar även den om listor men inkluderar att kunna sortera, filtrera och splittra listor. Den sjätte kategorin handlar om att ge åtkomst, utökning, sammanfogning och manipulering av objekt. Den sjunde kategorin ger möjligheten för sekvensmanipulationer som filtrering och testning (Thörnberg 2018).

Diskussion
<u>Fördelar</u>
De stora fördelarna med Lodash står ut i mängden i och med att det reducerar repetitiv kod vilket gör kvalitén på koden bättre vilket i sin tur leder till att strukturen i applikationen blir snyggare. Koden förenklas och gör att det blir mer läsbart för en tredje part (Dominguez u.å). Det Lodash gör bäst är att skapa byggstenar för metoder och funktioner som används ofta på ett sätt som funkar i många olika webbläsare (Morera 2018). Det leder vidare till en annan

fördel som är att Lodash funkar till och med ner på Internet Explore 6. Exempelvis så funkar inte en forEach-loop i native javascript på IE8 men en _.foreach funkar på listor, objekt och strängar (Dwyer 2015).

En tredje fördel är möjligheten att kunna följa Dont Repeat Yourself-principen (DRY) eftersom att du inte behöver upprepa något av din kod för att det redan är skrivet åt dig (Dwyer 2015).

En fjärde fördel med Lodash är att användare kan bygga sina egna anpassade bibliotek genom att välja ut funktioner för att på så sätt kunna bygga ett egenvalt verktygsbibliotek (Cambridge 2013).

Nackdelar

Det finns en oro kring att Lodash inte längre behövs i samma utsträckning idag i och med att JavaScript har utvecklats vid sidan om. Nya utgåvor har släppts och idag kan JavaScript erbjuda några av de verktyg som bara brukade finnas möjliga att bruka genom externa bibliotek som Lodash (Hanchett 2017). Enligt @Vaskemaskine (2019) som är en av de som skrivit i tråden ”Are Lodash and Underscroe still relevant in 2019. Which one should I focus on more?” på reddit.com. Det han tillför i diskussionen är att Lodash i en viss mån fortfarande är relevant men att många funktioner idag finns i ES6+. Idag är det en bättre idé att bara importera de funktioner du verkligen skulle behöva.

Prestandan är nästa nackdel som inte gynnar Lodash. Lodash är extremt väloptimerat men det går aldrig att bli lika snabbt som native JavaScript implementeringar (Areknawo 2019). Stepanov (2018) testade prestandan på två olika funktioner, ”find” och ”reduce” både i Lodash och i native JavaScript. Lodash fick en tid på 140 medan native JavaScript fick 0 som byggtid (Stepanov 2018).

React och Lodash

För det första är React precis som Lodash väldigt enkelt visuellt att se vad koden gör. Lodash är inkluderat i create-react-app och kan användas direkt när man startar ett react-projekt. Det som behövs är att skriva ”import chunck from ”lodash/chunk”; för att kunna använda metoderna som finns tillgängliga i Lodash (Pin 2019). React-lodash tillåter att använda lodash-funktioner som en reactkomponent, t.ex. genom att skriva såhär: import {isEmpty, Map} from ”react-lodash” om react även är importerat (npmjs.com 2019).

Framtid

Utifrån de fördelar och nackdelar som jag har upplevt att användare av Lodash anser så är framtiden för ramverket både positiv och negativ. Det finns vissa som anser att Lodash ger mycket utvecklingsmöjligheter till JavaScript i och med att man kan skriva mer komprimerad kod med mindre ansträngning. Det finns idag mycket som går att göra med ren Javascript som tidigare endast gick i Lodash och det kommer säkert bara bli mer som ren Javascript tar över i framtiden. Det som Lodash gör är att det pushar utvecklingen av JavaScript framåt vilket är positivt (Zhang 2017).

Referenser

@veksenn & @zthall, (u.å). Lodash.com. Länk: https://lodash.com/.
Besökt: [2020-02-24]

Dwyer, R (2015) Lodash Tutorial: How to use Lodash. Länk:
https://www.youtube.com/watch?v=cqw2i0HIj74. Besökt: [2020-02-24]

Hanchett, E (2017). 10 Javscript libraries and frameworks you should know about. Länk:
https://learntocodewith.me/posts/javascript-libraries-frameworks/#UnderscoreLodash
Besökt: [2020-02-24]

Morera, N (2018). A quick introduction to Lodash. Länk: https://upcity.com/blog/quick-introduction-lodash/. Besökt: [2020-02-24]

Dominguez, J.L (u.å). Länk: https://www.agiliacenter.com/lodash-library/?lang=en#What_is_a_Lodash_Library. Besökt: [2020-02-24]

Cambridge, K (2013) Länk:
https://web.archive.org/web/20180101093627/http://kitcambridge.be/blog/say-hello-to-lo-dash/
besökt: [2020-02-24]

@Vaskemaskine (2019).
Länk:https://www.reddit.com/r/learnjavascript/comments/alkkqo/are_lodash_and_underscore_still_relevant_in_2019/. Besökt:[2020-02-24]

Areknawo (2019). Lodash and usefulnes of utility libraries.
Länk: https://areknawo.com/lodash-and-usefulness-of-utility-libraries/ besökt: [2020-02-24]

Stepanov, V (2018). Why you shouldn't use lodash anymore and use pure JavaScript instead. Länk:
https://codeburst.io/why-you-shouldnt-use-lodash-anymore-and-use-pure-javascript-instead-c397df51a66. Besökt: [2020-02-24]

Corbacho, D (2018). Debouncing and throttling explained through examples. Länk: https://css-tricks.com/debouncing-throttling-explained-examples/ besökt: [2020-02-24]

Thörnberg, J (u.å). Effektivisera och reducera kod med Lodash. Länk:
http://media.hv.se/kurser/informatik-ail/old-vri401-vru401/effektivisera-och-reducera kod-med-lodash/. Besökt: [2020-02-24]

Pin, C (2019). Today I Learned: Lodash library is included in Create-React-App. Länk:
https://medium.com/@chyanpin/today-i-learned-lodash-library-is-included-in-create-react-app-34174aa479d0. Besökt: [2020-02-24]

Npmj.com (2019). React-Lodash. Länk: https://www.npmjs.com/package/react-lodash
Besökt: [2020-02-24]

Zhang, J (2017). Why use Lodash when ES6 is available. Länk:
http://shzhangji.com/blog/2017/03/13/why-use-lodash-when-es6-is-available/ besökt: [2020-02-24]

Bilaga 1

```
let pets = [
        {name: 'Fluffy', age: 9, color: 'brown'},
        {name: 'Sandie', age: 6, color: 'brown'},
        {name: 'Fia', age: 6, color: 'brown'},
        {name: 'Molly', age: 8, color: 'sand'},
        {name: 'Sally', age: 6, color: 'blue'},
        {name: 'blabla', age: 2, color: 'red'},
        {name: 'bläblä', age: 6, color: 'brown'},
    ];

function dogs(){

        //const Numbers = document.getElementById('Numbers')
        var matches = [];
            pets.forEach(function (pet) {
            if (pet.age === 6 && pet.color === 'brown') {
                matches.push(pet);
            }
            console.log(matches)
            return matches
    });

}

function dogsLodash(){
    var matches = _.filter(pets, {age: 6, color: 'brown'});
    console.log(matches)
}
```

Bilaga 2

```
let pets = [
        {name: 'Fluffy', age: 9, color: 'brown'},
        {name: 'Sandie', age: 6, color: 'brown'},
        {name: 'Fia', age: 6, color: 'brown'},
        {name: 'Molly', age: 8, color: 'sand'},
        {name: 'Sally', age: 6, color: 'blue'},
        {name: 'blabla', age: 2, color: 'red'},
        {name: 'bläblä', age: 6, color: 'brown'},
    ];

function dogs(){

        var matches = [];
            pets.forEach(function (pet) {
            if (pet.age === 6 && pet.color === 'brown') {
                matches.push(pet);
            }
            console.log(matches)
            return matches
    });

}

function dogsLodash(){
    var matches = _.filter(pets, {age: 6, color: 'brown'});
    console.log(matches)
}
```