

Laborationsrapport

Laboration / namn på laborationsmoment

DT173G, Webbutveckling III

Författare: Emilia Holmström, emho2003@student.miun.se

Termin, år: HT, 2021



Mittuniversitetet
MID SWEDEN UNIVERSITY

Sammanfattning

Beskriv kortfattat vad uppgiften går ut på.

Innehållsförteckning

Sammanfattning.....	2
Beskriv kortfattat vad uppgiften går ut på.....	2
Innehållsförteckning.....	3
1 Frågor	4
1.1 Ecmascript	4
1.1.1 ECMA och ECMAScript	4
1.1.2 ECMAScript historia.....	4
1.2 Förklara följande tekniker	6
1.2.1 Classes	6
1.2.2 High-order functions	7
1.2.3 Spread operators ("spreads").....	9
1.2.4 Destructuring	9
1.2.5 Arrow Functions	10
1.3 Förklara Fetch API och Promises	10
1.4 Bakåt-kompabilitet.....	10
1.5 TypeScript analys	11
1.5.1 Historia	11
1.5.2 Fördelar med TypeScript (över JavaScript) med exempel	11
2 Slutsatser.....	13
3 Källförteckning	14

1 Frågor

1.1 EcmaScript

1.1.1 ECMA och ECMAScript

Ecma står för European Computer Manufacturers Association och är en ideell standardiserings-organisation. Ecma ansvarar för flera olika standarder (och dokumentationer) där några bland annat är JSON (ECMA-404), C # Language Specification (ECMA-334), Office Open XML (ECMA-376) och ECMAScript (ECMA-262) [1] vilket denna rapport är inriktad på.

ECMAScript är ett programmeringsspråk som huvudsakligen har använts på klientsidan på internet (World Wide Web) men med node.js blir det allt vanligare att språket även används till serverapplikationer och tjänster [2].

JavaScript grundar sig på dokumentationen för ECMAScript. Som freeCodeCamp.org nämner i artikeln *"What's the difference between JavaScript and ECMAScript?"* kan man säga att läsa en dokumentation av ECMAScript lär man sig skapa ett skriptspråk medan om man läser en dokumentation av JavaScript så lär man sig använda ett skriptspråk [3].

1.1.2 ECMAScript historia

Utgåva ett till tre

Totalt finns det 11 utgåvor av ECMA-262 utgivna i september 2021. Där de första tre utgåvorna som kom -97, -98 och -99 la grunden till ECMAScript [4].

Fjärde utgåvan

Den fjärde utgåvan skulle bli den stora uppdateringen. I stället kom någon mindre uppdatering mellan 2003 och 2008. Det berodde på att de som skapar ECMAScript hamnade olika lägger med olika versioner om hur det skulle jobba med ECMAScript 4. Där den ena gruppen "ECMAScript 4 Camp" som bestod av Adobe, Mozilla, Opera och Google ville fortsätta jobba med den tänkta stora uppdatering för ECMAScript 4 medan den andra gruppen "ECMAScript 3.1 Camp" bestående av Yahoo och Microsoft ville fixa buggar och mindre uppdateringar för ECMAScript 3 [5][6].

Efter en konferens i Oslo 2008 kom de att enas till att ECMAScript 4 övergavs och i stället skulle ECMAScript 3.1 bli ECMAScript 5, skapa en ny större uppdatering men inte lika omfattande (vilket sedan blev ECMAScript 6) samt att några nya funktioner skulle släppas [7][8].

Några nyheter med utgåva fyra var klasser, modulsystem, generatorer och iteratorer, destruktureringsuppdag, algebraiska datatyper, valfria typkommentarer och statisk typning samt ordnades buggar från tidigare utgåva [9].

ECMAScript 5

Med fortsatta konflikter inom utvecklingsparterna för ECMAScript kom till slut en ny utgåva i december 2009, ECMAScript 5. Utgåvan fokus låg på säkerhets- och biblioteks-uppdateringar, kompatibilitet samt att versionen skulle ha samma syntax och semantik som fanns i förgående version (ECMAScript 4) [9].

Sjätte utgåvan / ECMAScript 2015

Den sjätte utgåvan ECMAScript 6 som också senare fick namnet ECMAScript 2015 handlade om att skapa en ny syntax för komplexa applikationer och klassdeklARATIONER. Annat nytt är iteratorer och loopar, Python-liknande generatorer, pilfunktionsuttryck och nyckelord för deklARATIONER, konstanta lokala deklARATIONER för att nämna några [10].

För att inte upprepa det som hände med fjärde utgåvan (att den riktigt aldrig släpptes) så fick ECMAScript under kodnamnet ECMAScript.next. När det hade mognat klart började man kalla versionen för den sjätte. ECMA hade i samma veva tänkt att det skulle släppas en ny version årligen och därför ändrade man namnet till ECMAScript 2015. Men ECMAScript 6 hade redan fått fäste och blev så många referera till denna version [11].

ECMAScript 2016 / Sjunde utgåvan

Den sjunde upplagen är den första årliga återkommande utgåva och går under namnet ECMAScript 2016 (och inte ECMAScript 7) [12].

Den innehöll funktioner som kan inkludera block-scoping av variabler och funktioner, destruktureringsmönster av variabler och korrekta svansanrop för att nämna några. Något som kom att uppskattas extra mycket var `async/await` som motsvarar `Math.Pow` men med enklare syntax än Python eller Perl [13].

ECMAScript 2017

Nyheter här var inkludering av olika Object funktioner för att kunna hantera objekt enklare och konstruktioner till `async/await` som kan använda generatorer och löften [14].

ECMAScript 2018

Nyheter var spreadoperatören (en enklare kopiering av objektgenskaper), viloparametrar, asynkron iteration `Promise.prototype.finally` och tillägg till `RegExp` [15].

ECMAScript 2019

En uppdatering som gjorde att arrayer blev mer tillgängliga och ökad stabilitet [16].

ECMAScript 2020

Nyheten `BigInt` introducerades och tillåter representation och manipulation av heltal bortanför `Number.MAX_SAFE_INTEGER` [17].

ECMAScript 2021

Den senaste version som finns ute i skrivande stund och är den 12:e utgåvan. Versionen kom med replaceAll-metoden som är till för strängar, Promise.any är en löftekombinator, logiska tilldelningsoperatorer, FinalizationRegistry som kan hantera registrering och avregistrering av saneringsoperationer och AggregateError som kan samla flera fel samtidigt; är några av nyheterna för versionen [18].

1.2 Förklara följande tekniker

1.2.1 Classes

Gör det möjligt med objektorienterad programmering. Man kan konstruera/skapa en kod som sedan går att hämta återkommande i koden.

```
class Fruits {  
  private name: string;  
  private color: string;  
  private price: number;  
  
  constructor(name: string, color: string, price: number) {  
    this.name = name;  
    this.color = color;  
    this.price = price;  
  }  
  
  showFrutis() {  
    return `This is a ${this.name}, its color is  
    ${this.color}. It costs ${this.price} kr/kg!!`;  
  }  
}  
  
let fruit1 = new Fruit('banan', 'yellow', 23);  
let fruit2 = new Fruit('cherry', 'red', 35);  
  
console.log(fruit1.showFrutis());  
console.log(fruit2.showFrutis());
```

Ett annat sätt man kan göra i stället för att använda *return* är att direkt skriva vad som ska hända, i detta fallet ett meddelande till konsolen. Exempel på hur det skulle se ut:

```
showFrutis() {  
  console.log(`This is a ${this.name}, its color is  
  ${this.color}. It costs ${this.price} kr/kg!!`);  
}
```

1.2.2 High-order functions

Är när en funktion använder sig av andra funktioner. Antingen genom att returnera en funktion eller ha dem som argument [19].

Till följande kodexempel används en array som skapades en separat fil som exporteras. Koden i den filen ser ut så här:

```
export const fruitsArr = [  
  {name: 'Banana', color: 'yellow'},  
  {name: 'Apple', color: 'red'},  
  {name: 'Melon', color: 'green'},  
  {name: 'Cherry', color: 'red'},  
  {name: 'Kiwi', color: 'green'},  
  {name: 'Pear', color: 'green'}  
];
```

Och hämtas sedan med kodsnutten:

```
import {fruitsArr} from "./fruitsArr";
```

1.2.2.1 *forEach*

Här skrivs alla frukters namn ut i konsolen:

```
fruitsArr.forEach(fruitsArr => {  
  console.log(fruitsArr.name);  
});
```

1.2.2.2 *map*

```
//High-order function
let fruits = fruitsArr.map(function(fruitsArr) {
  return fruitsArr.color;
})

//Return like booleans
let fruits2 = fruitsArr.map((fruitsArr) => {
  return fruitsArr.color === 'red';
});

//Adds them together as a message
let fruits3 = fruitsArr.map(function(fruitsArr) {
  return fruitsArr.color + ' is the color of ' + fruitsArr.name;
})
```

1.2.2.3 *filter*

Filtreras alla frukter ut som har färgen röd. Använder sig också av en funktion som argument, dvs en high-order function.

```
let fruitRed = fruitColor.filter(function(fruitColor){
  return fruitColor.color === 'red';
});

//Without higher order function
let fruitGreen= fruitsArr.filter((fruitsArr) => {
  return fruitsArr.color === 'green';
});
```


1.2.3 Spread operators ("spreads")

Gör det möjligt att slå ihop flera arrays till en, göra en String till array eller göra en array till en string [20][21]

```
//Slår ihop array
let food = [...fruitsArr, ...veggis];

//String -> array
let banana = "Banana";
let bananaArr = [...banana];

//Array -> string
let foodArr = ['rice', 'nuggets', 'curry'];
let foodString = [...foodArr];
```

1.2.4 Destructuring

Destructuring används när man vill extrahera/hämta värden från en array eller ett objekt [22][23].

```
//Exemple with object
const banana = {
  name: 'banana',
  color: 'yellow',
  type: 'berry'
}

function showFruit({name, color, type}) {
  console.log(`This is a ${color} ${name}
  and it belongs to the family ${type}`);
}

//Exemple with array
const x = [23, 55, 2, 6, 78];
const [y, z] = x;

console.log(y + " " + z);
```

1.2.5 Arrow Functions

Arrow Functions är en mer kompakt funktion som är kortare och som använder *this* på ett annat sätt än vad en vanlig funktion gör [24].

```
const sayHello = (name = 'Agent X') => {  
  console.log(`Hello ${name}!`)  
}  
  
let person1 = "William";  
  
sayHello(person1);  
sayHello();
```

1.3 Förklara Fetch API och Promises

Fetch API är en nyare variant för att hantera XMLHttpRequest. Med hjälp av en url kan man hämta in data som man sedan kan använda i sin kod. Motsvarande är ett AJAX-anrop [25].

Genom att man kan köra flera saker samtidigt (asynchronous programming) kan det hända att man inte hinner att läsa in all data som behövs till olika funktioner. För att då undvika stopp/errors i koden kan man använda sig av ett **Promises**. Den låter koden fortsätta rulla på och "meddelar" funktioner som ännu inte fått sin data att den kommer [26] [27].

1.4 Bakåt-kompabilitet

För att ECMAScript ska fungera på flera webbläsare gör man något som heter transpilering (transpile på engelska). Man låter koden skrivas om till en annan version (än den man själv skrivit i). Om man vill transpilera koden med hjälp av Gulp kan man använda *babel* (har dock stöd för flera olika verktyg) [28].

Git clone https://github.com/97emiria/webbIII_m4.git

1.5 TypeScript analys

TypeScript utvecklats och underhålls av Microsoft och är en öppen källkod som finns på GitHub sedan 2014 (dessförinnan codePlex). TypeScript är superset av JavaScript. Fritt översatt och förklaring på superset är att det fungerar likadant/bygger på samma principer men är bättre och/eller har mer funktionalitet. Med andra ord kan man säga att TypeScript är som JavaScript fas en bättre version. Genom att TypeScript bygger på JavaScript gör att det fungerar där JavaScript fungerar [29].

1.5.1 Historia

Efter två år av utveckling av TypeScript kom det att offentliggöras i oktober 2012 som version 0.8. Miguel de Icaza berömde språket men påpekade bristen på IDE-stöd. Men sedan april 2021 finns numera stöd förutom Microsoft Visual Studio även på Linux och OS X samt textredigerare som Emacs, Vim och Webstrom för att nämna några. Året därpå släpptes en uppdatering, version 0.9 som hade stöd för generics (wikipedia) [30].

2014 kom TypeScript 1.0 och man valde att flytta över den öppna källkoden till GitHub ifrån CodePlex som används tidigare. 2016 kom version 2.0 med nya funktioner, bland annat gjorde man det möjligt att förhindra null världen i variabler [31].

Juli 2018 kom version 3.0 och ökade språk tilläggen och när version 4.0 släpptes 2020 kom det inte med några större förändringar utan mer förbättringar, bland annat på språktilläggen [32].

1.5.2 Fördelar med TypeScript (över JavaScript) med exempel

Tillskillnad från JavaScript som ger felmeddelanden när man testkör koden så ger TypeScript felmeddelanden direkt något är fel och med en bättre förklaring. Det gör att man tidigare kan hitta fel och buggar i koden [33].

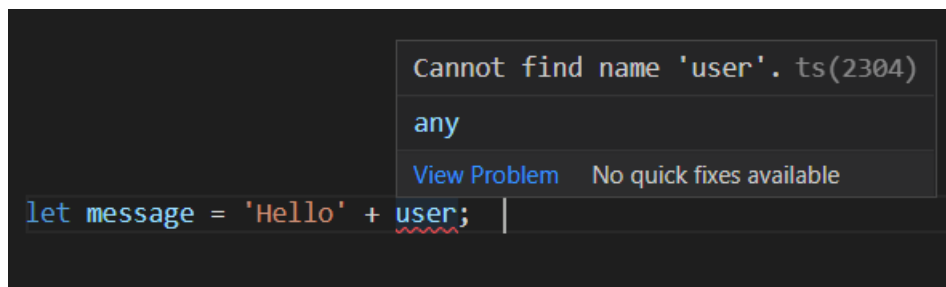
Exempel här har man skapat ett meddelande som använder sig av en variabel som inte finns. I JavaScript säger den ingenting:

```
let message = 'Hello ' + user;
```

Utan felmeddelandet kommer när man testkör koden:

```
C:\Program Files\nodejs\node.exe .\src\javascript\main.js  
> Uncaught ReferenceError: user is not defined  
Process exited with code 1
```

Men däremot så blir variabeln *user* röd markerat i TypeScript för den inte hittar variabeln.



```
let message = 'Hello' + user;
```

En annan bra fördel med TypeScript är att man kan bestämma variabelns typ, om det ska vara en string eller int [34]. Exempel:

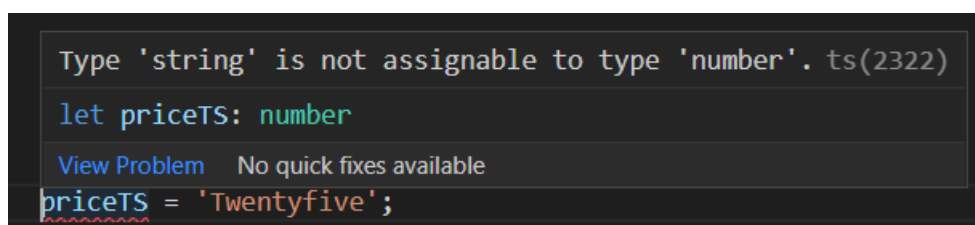
```
let fruitJS;
let fruitTS: string;

fruitJS = 'Banana';
fruitTS = 'Banana';

let priceJS;
let priceTS: number;

priceJS = 25;
priceTS = 25;
```

Om man nu råkar skriva fel värde i en bestämd variabeln kommer felmeddelande:



```
let priceTS: number
priceTS = 'Twentyfive';
```

Ska man ha blandade värden kan man används sig av any:

```
let priceTS: any;
priceTS = 25;
priceTS = 'Twentyfive';
```

1.5.3 Utvecklingsmiljön

För ovanstående exempel med TypeScript har Node.js och NPM-paket används för TypeScript och sedan babel för bakåtkompilering.

2 Slutsatser

Att förstå skillnaden mellan ECMAScript, JavaScript och TypeScript teoretiskt är inte jättesvårt. Men när de väl kommer till de praktiska känns det inte lika självklart då det är samma språkgrund men ändå inte. Tror det kräver lite mer djupdykning inom TypeScript för att verkligen förstå skillnaden och användbarheten med TypeScript gentemot JavaScript.

Har också hunnit att komma i gång med projektarbetet och ibland känns det bara omständligare med TypeScript för det finns mer delar som kan krångla med gulp och konvertering osv men att få felmeddelanden är ju guld värd. Men allt handlar ju om att lära sig det och bli bekant. I höstas (2020) såg jag absolut inte fram emot att läsa mer JavaScript men vet inte vad som ska bli mest spännande JavaScript-kursen eller C# nu närmast.

3 Källförteckning

- [1] Wikipedia, "Ecma International". [Internet]
https://en.wikipedia.org/wiki/Ecma_International
Senast redigerad 21-07-32. Hämtad 2021-09-28.
- [2] Wikipedia, "ECMAScript". [Internet]
<https://en.wikipedia.org/wiki/ECMAScript>
Senast redigerad 21-08-31. Hämtad 2021-09-28.
- [3] freeCodeCamp,
"What's the difference between JavaScript and ECMAScript?". [Internet]
<https://www.freecodecamp.org/news/whats-the-difference-between-javascript-and-ecmascript-cba48c73a2b5/>
Publicerad 2017-10-28. Hämtad 2021-09-28.
- [4] Wikipedia, "ECMAScript". [Internet]
<https://en.wikipedia.org/wiki/ECMAScript>
Senast redigerad 21-08-31. Hämtad 2021-09-28.
- [5] Wikipedia, "ECMAScript". [Internet]
<https://en.wikipedia.org/wiki/ECMAScript>
Senast redigerad 21-08-31. Hämtad 2021-09-28.
- [6] Ben Ilehodu, "History of ECMAScript". [Internet]
<https://www.benmvp.com/blog/learning-es6-history-of-ecmascript/>
Publicerad 205-07-29. Hämtad 2021-09-28.
- [7] Wikipedia, "ECMAScript". [Internet]
<https://en.wikipedia.org/wiki/ECMAScript>
Senast redigerad 21-08-31. Hämtad 2021-09-28.
- [8] Ben Ilehodu, "History of ECMAScript". [Internet]
<https://www.benmvp.com/blog/learning-es6-history-of-ecmascript/>
Publicerad 205-07-29. Hämtad 2021-09-28.
- [9] Wikipedia, "ECMAScript". [Internet]
<https://en.wikipedia.org/wiki/ECMAScript>
Senast redigerad 21-08-31. Hämtad 2021-09-28.
- [10] Wikipedia, "ECMAScript". [Internet]
<https://en.wikipedia.org/wiki/ECMAScript>
Senast redigerad 21-08-31. Hämtad 2021-09-28.

- [11] Ben Ilehodu, "History of ECMAScript". [Internet]
<https://www.benmvp.com/blog/learning-es6-history-of-ecmascript/>
Publicerad 2015-07-29. Hämtad 2021-09-28.
- [12] Wikipedia, "ECMAScript". [Internet]
<https://en.wikipedia.org/wiki/ECMAScript>
Senast redigerad 21-08-31. Hämtad 2021-09-29.
- [13] Wikipedia, "ECMAScript". [Internet]
<https://en.wikipedia.org/wiki/ECMAScript>
Senast redigerad 21-08-31. Hämtad 2021-09-29.
- [14] Wikipedia, "ECMAScript". [Internet]
<https://en.wikipedia.org/wiki/ECMAScript>
Senast redigerad 21-08-31. Hämtad 2021-09-29.
- [15] Wikipedia, "ECMAScript". [Internet]
<https://en.wikipedia.org/wiki/ECMAScript>
Senast redigerad 21-08-31. Hämtad 2021-09-29.
- [16] Wikipedia, "ECMAScript". [Internet]
<https://en.wikipedia.org/wiki/ECMAScript>
Senast redigerad 21-08-31. Hämtad 2021-09-29.
- [17] Wikipedia, "ECMAScript". [Internet]
<https://en.wikipedia.org/wiki/ECMAScript>
Senast redigerad 21-08-31. Hämtad 2021-09-29.
- [18] Wikipedia, "ECMAScript". [Internet]
<https://en.wikipedia.org/wiki/ECMAScript>
Senast redigerad 21-08-31. Hämtad 2021-09-29.
- [19] Haverbeke M. Eloquent Javascript: a modern introduction to programming. Third edition. San Fransisco: No starch press; 2018.
- [20] Mattias Dahlgren, "Ecmascript & Typescript". [Video]
https://www.youtube.com/watch?v=XQ17hkXTPEI&t=1769s&ab_channel=MattiasDahlgr en
Publicerad 2021-05-20. Hämtad 2021-09-30.

- [21] Dave Gray, "Higher Order Functions Javascript | forEach, filter, map, and reduce functions" [Video]
https://www.youtube.com/watch?v=7BeT6lsudL4&t=425s&ab_channel=DaveGray
Publicerad 2020-11-04. Hämtad 2021-10-01
- [22] Mattias Dahlgren, "Ecmascript & Typescript". [Video]
https://www.youtube.com/watch?v=XQ17hkXTPEI&t=1769s&ab_channel=MattiasDahlgr
[en](#)
Publicerad 2021-05-20. Hämtad 2021-09-30.
- [23] dcode, "Object Destructuring in Javascript". [Video]
https://www.youtube.com/watch?v=G4T2ZgJPKbw&ab_channel=dcode
Publicerad 2017-11-30. Hämtad 2021-10-01.
- [24] Mattias Dahlgren, "Ecmascript & Typescript". [Video]
https://www.youtube.com/watch?v=XQ17hkXTPEI&t=1769s&ab_channel=MattiasDahlgr
[en](#)
Publicerad 2021-05-20. Hämtad 2021-09-30.
- [25] Mattias Dahlgren, "Ecmascript & Typescript". [Video]
https://www.youtube.com/watch?v=XQ17hkXTPEI&t=1769s&ab_channel=MattiasDahlgr
[en](#)
Publicerad 2021-05-20. Hämtad 2021-10-01.
- [26] Mattias Dahlgren, "Ecmascript & Typescript". [Video]
https://www.youtube.com/watch?v=XQ17hkXTPEI&t=1769s&ab_channel=MattiasDahlgr
[en](#)
Publicerad 2021-05-20. Hämtad 2021-10-01.
- [27] Alfredo Ryelcius, "JavaScript Promise for Dummies". [Internet]
<https://medium.com/codex/javascript-promise-for-dummies-f3e763c2ec26>
Publicerad 2021-05-31. Hämtad 2021-10-01.
- [28] Mattias Dahlgren, "Ecmascript & Typescript". [Video]
https://www.youtube.com/watch?v=XQ17hkXTPEI&t=1769s&ab_channel=MattiasDahlgr
[en](#)
Publicerad 2021-05-20. Hämtad 2021-10-01
- [29] Wikipedia, "TypeScript". [Internet]
<https://en.m.wikipedia.org/wiki/TypeScript>
Senast redigerad 2021-09-03. Hämtad 2021-10-08.
- [30] Wikipedia, "TypeScript". [Internet]
<https://en.m.wikipedia.org/wiki/TypeScript>
Senast redigerad 2021-09-03. Hämtad 2021-10-08.

- [31] Wikipedia, "TypeScript". [Internet]
<https://en.m.wikipedia.org/wiki/TypeScript>
Senast redigerad 2021-09-03. Hämtad 2021-10-08.

- [32] Wikipedia, "TypeScript". [Internet]
<https://en.m.wikipedia.org/wiki/TypeScript>
Senast redigerad 2021-09-03. Hämtad 2021-10-08.

- [33] Jelvix, "TYPESCRIPT VS JAVASCRIPT". [Video]
https://www.youtube.com/watch?v=3iAx52Y7IEc&ab_channel=Jelvix
Publicerades 2020-08-26. Hämtades 2021-10-25.

- [34] Fireship, "TypeScript in 100 Seconds". [Video]
https://www.youtube.com/watch?v=zQnBQ4tB3ZA&ab_channel=Fireship
Publicerades 2020-11-25. Hämtades 2021-10-25.