# Verschillende classes

* Database.js
  + Hier komen methodes die simpelweg iets van de FYS.Cloud willen gebruiken of willen ophalen. Deze zal het meest worden gebruikt
* Notification.js
  + Hier staan methodes waarmee je meldingen kan laten weergeven. Handig hulpje en bespaart wat tijd.
* User.js
  + Hier komen gebruiker gerelateerde methodes zoals ***getUserData()*** of ***updateUserData().***
  + Elke methode wat de huidige ingelogde persoon betreft zal deze class moeten gebruiken voor bijvoorbeeld ***user.userID (Gebruikers primary sleutel)***
* Translation.js
  + Vertaling methodes en de daar bijhorende methodes. Aangezien dit al is gemaakt zal hier vrijwel niet in gewerkt worden.

# App.js

## App.js

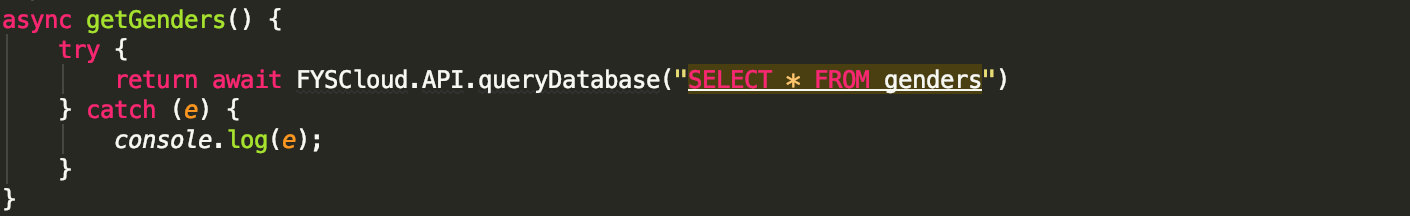
Hier komen de JQuery dollar functies en de initialisatie functie. Eigenlijk moet je dit bestand gewoon zien als de plek waar de classes en de HTML samen komen om daadwerkelijke resultaat te creeëren.

## Initialisatie functie

in de app.js komen alle bovenstaande classes samen en worden ze geinitialiseerd in de ***init().*** In de Init functie kun je ook gegevens ophalen waarvoor je geen klik of een bepaalde handeling voor wilt uitvoeren.

## Async functies

Meestal voer je een methode uit waarin de FYS.Cloud wordt aangeroepen. Er is dan sprake van een HTTP request en deze kostte wat tijd voordat ze ingeladen zijn.

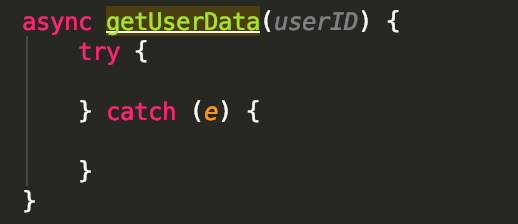
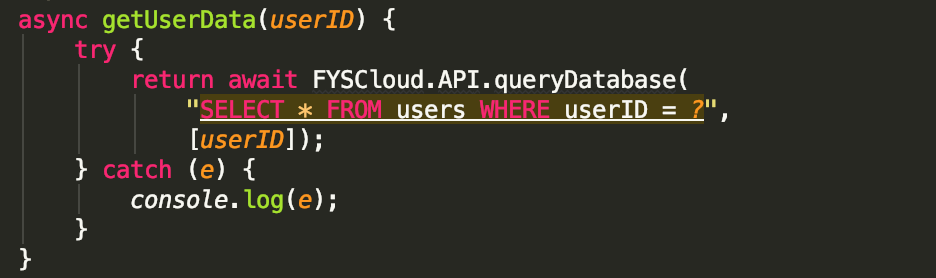
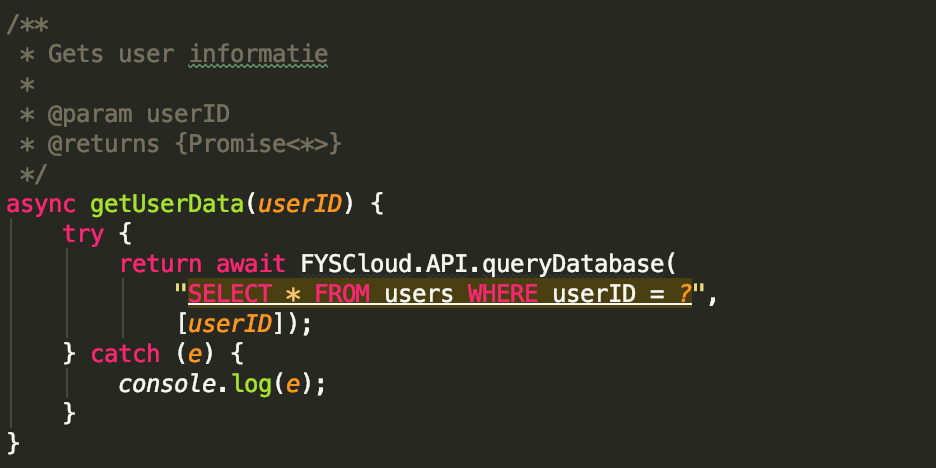
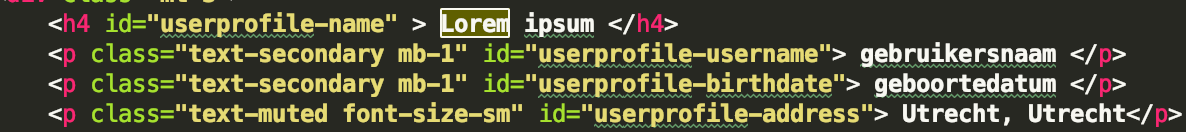
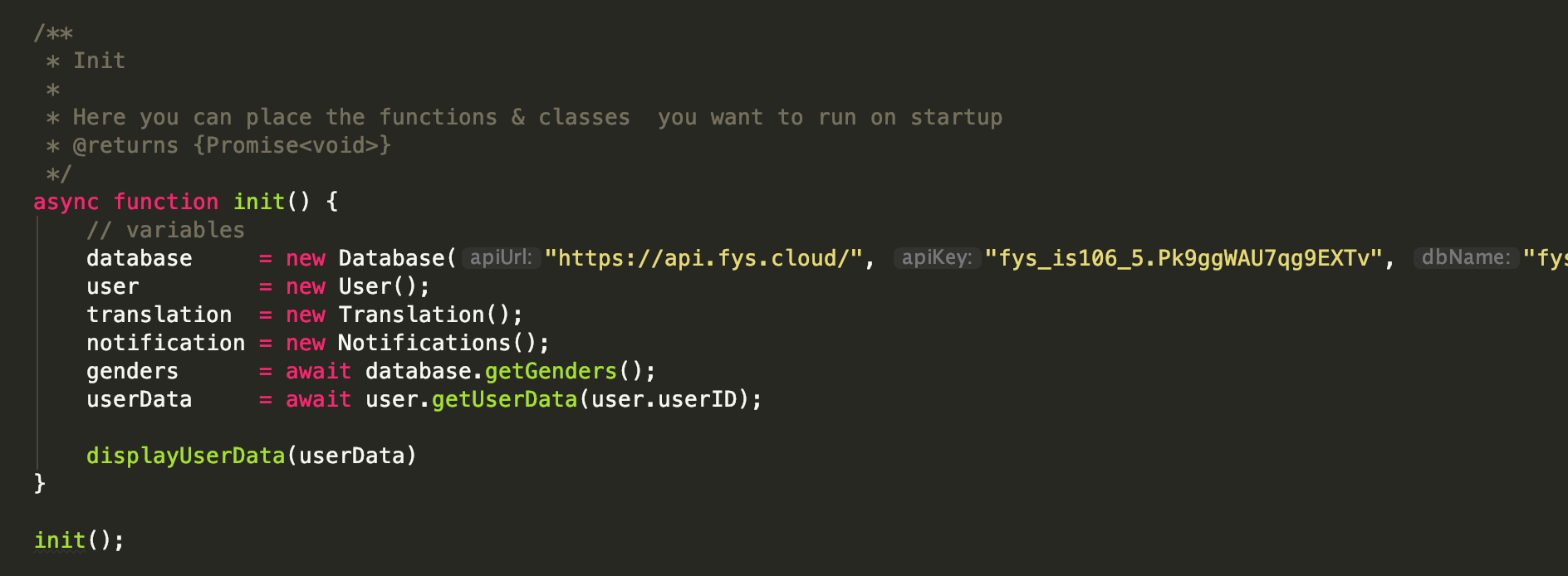
In ons geval kan het zijn dat wij een verzoek doen om gebruikersgegevens op te halen maar als wij de query vertonen via console.log staat er “**UNDEFINED**” dit betekent in principe dat de resultaten nog niet zijn ingeladen. Bekijk deze voorbeeld in **Database.js**: 

Je kunt zien dat ik eerst aangeef dat deze functie asynchroon is doormiddel van ***async*** te gebruiken.

Vervolgens maak ik een try/catch blok wat eigenlijk inhoudt *“****Probeer*** *dit uit te voeren of* ***vang*** *anders de foutmelding op en weergeef dit op de console”.*

En tot slottikken we een FYS.Cloud query in met daarvoor een **await** dit betekent letterlijk “**Wacht** met doorgaan in de code totdat deze resultaat is ingeladen”.

# Stappenplan bouwen functionaliteit

1. Maak eerst een async methode in de juiste class. 
2. Zet bij de **try** jouw query en bij de **catch** een console.log voor de foutmelding als die er is (Vergeet geen **await** te gebruiken).
3. Rond je functie af door commentaar en identatie goed te zetten.
4. Maak nu gebruik van ID’s om de elementen die je wilt gaan gebruiken bereikbaar te maken voor jquery.
5. Gebruik de initialisatie functie om gegevens uit je class methode te halen.
6. Maak een functie aan die al jou gegevens kan verwerken in de site. (Deze functies staan onder de initialisatie functie)
7. En tot slot zet je de opgehaalde data uit je class (zie: **userData** variabel) in de gemaakte functie dat je elementen vult met informatie (zie **displayUserData** functie).

# Extra informatie

De stappenplan zal niet voor elke functie werken en dat is het ding met programmeren. Er zijn zoveel manieren om iets te doen maar dat kan enorm chaotisch zijn in samenwerking.

Hoe dan ook, dit is ongeveer de wijze waarop je het zou doen. Google vooral wat sommige onbekende termen betekenen want zo leer je het snelst.

**LET OP: In de code is het belangrijk dat je niks verwijderd omdat je de nut er niet van kan in zien. Vraag van te voren of iemand weet wat een bepaald stukje code bij hoort, zo voorkom je errors.**