

## Cloud oefeningen – week 1

---

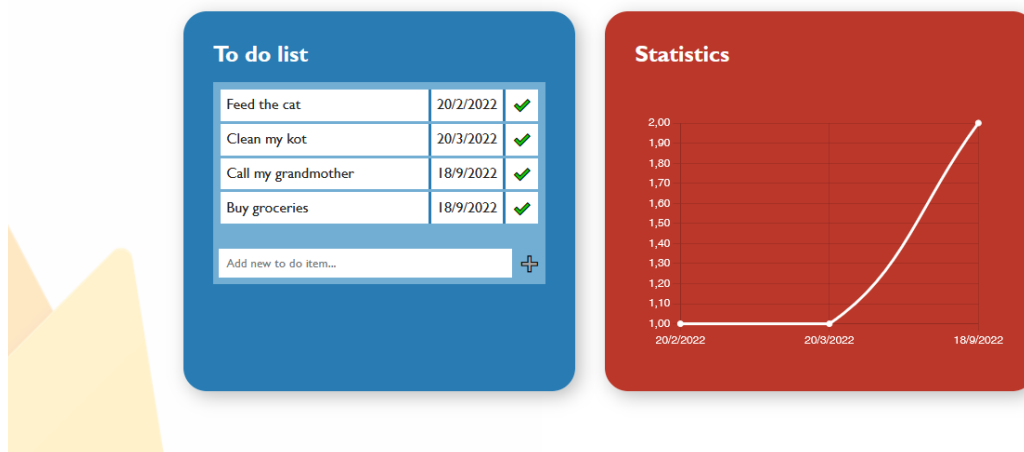
### 1 Introductie

Als oefening gaan we over de loop van 3 weken een cloud applicatie maken waarmee we een To Do lijst kunnen bijhouden. De cloud technologieën die we gaan gebruiken zijn steunen op het Firebase platform:

- Cloud Firestore (databank)
- Firebase Authentication (authenticatie/autorizatie)
- Firebase hosting (hosten van onze frontend)

Onze applicatie zelf gaat volledig geschreven zijn in HTML, CSS en Javascript.

#### Todo app - Firebase integration

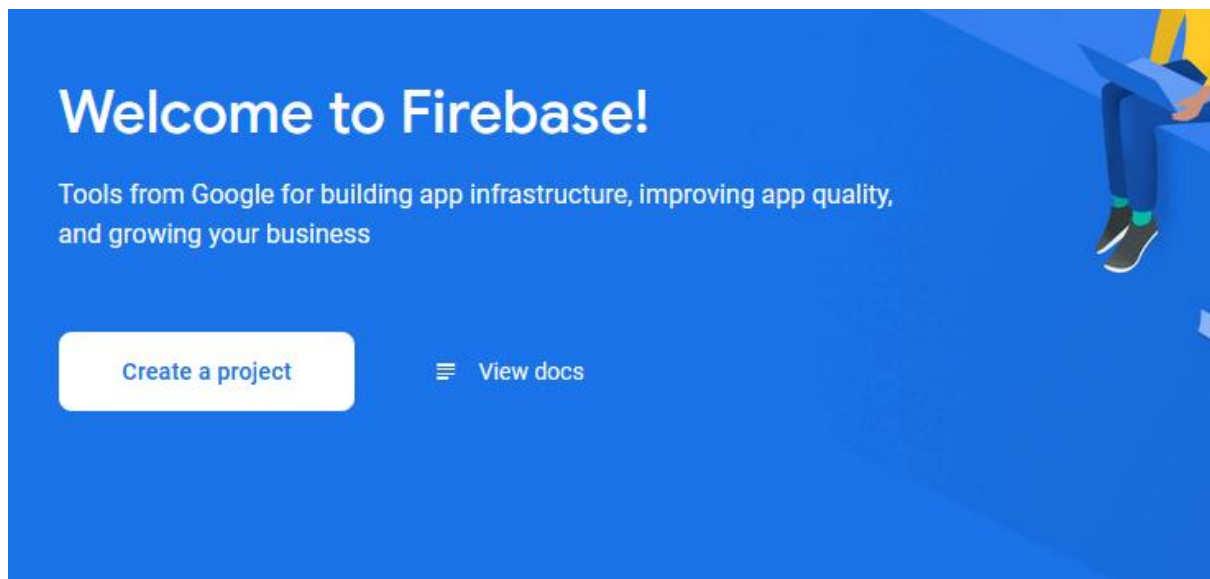


## 2 Opzetten van de applicatie

### 2.1 Firebase project aanmaken

Voor we beginnen aan het programmeren gaan we eerst een applicatie opzetten in Firebase.

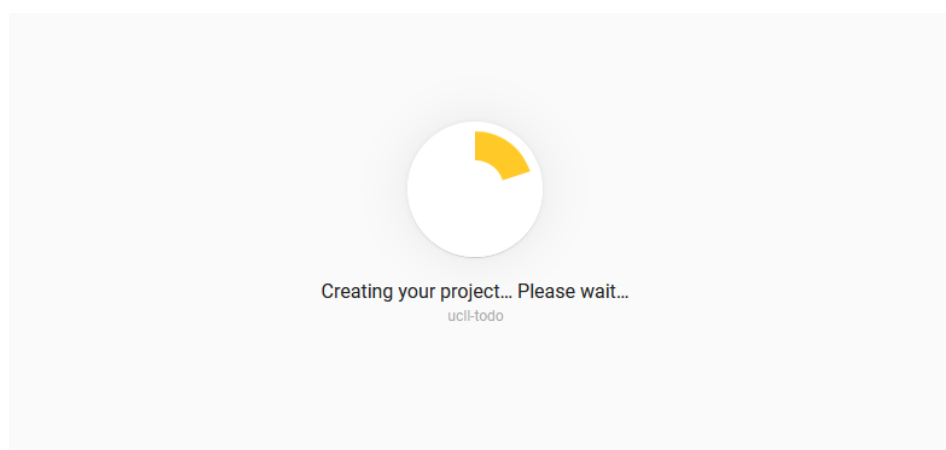
Ga naar <https://console.firebase.google.com> en meld je aan met een google account.



Je krijgt bovenstaand scherm te zien, klik op “Create a project”. Doorloop de stappen en vul de volgende informatie in:

1. Naam van het project (ik heb gekozen voor ucll-todo)
2. Accepteer de Firebase terms and conditions
3. Accepteer dat je Firebase enkel gaat gebruiken voor doeleinden die in lijn zijn met ons vakgebied.
4. Kies of je Google Analytics integratie wilt (ik heb dit uitgezet)
5. Creëer je project.

Firebase zal nu alles klaarzetten voor ons project.

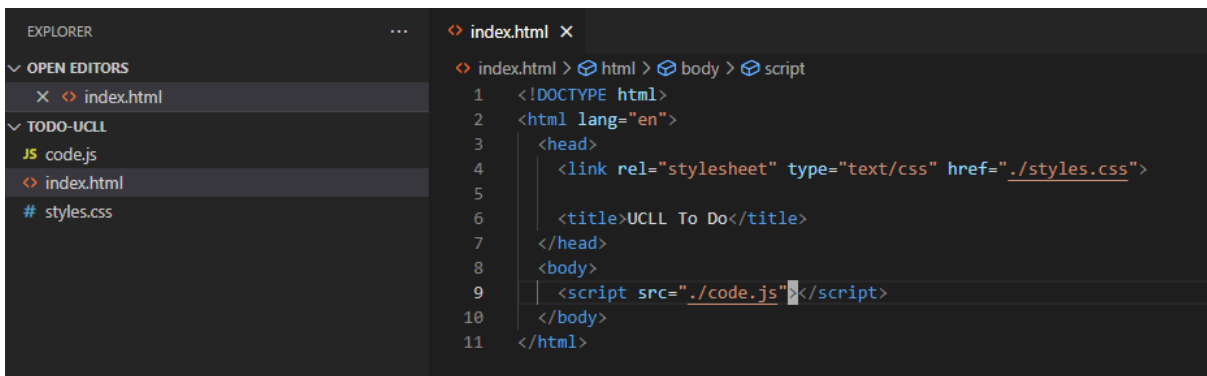


## 2.2 Frontend

Zet lokaal een nieuw HTML-project op via jouw favoriete IDE. We starten met 3 bestanden:

- index.html
- styles.css
- code.js

Start met de basis opzet van jouw html pagina en laadt de css en javascript in:



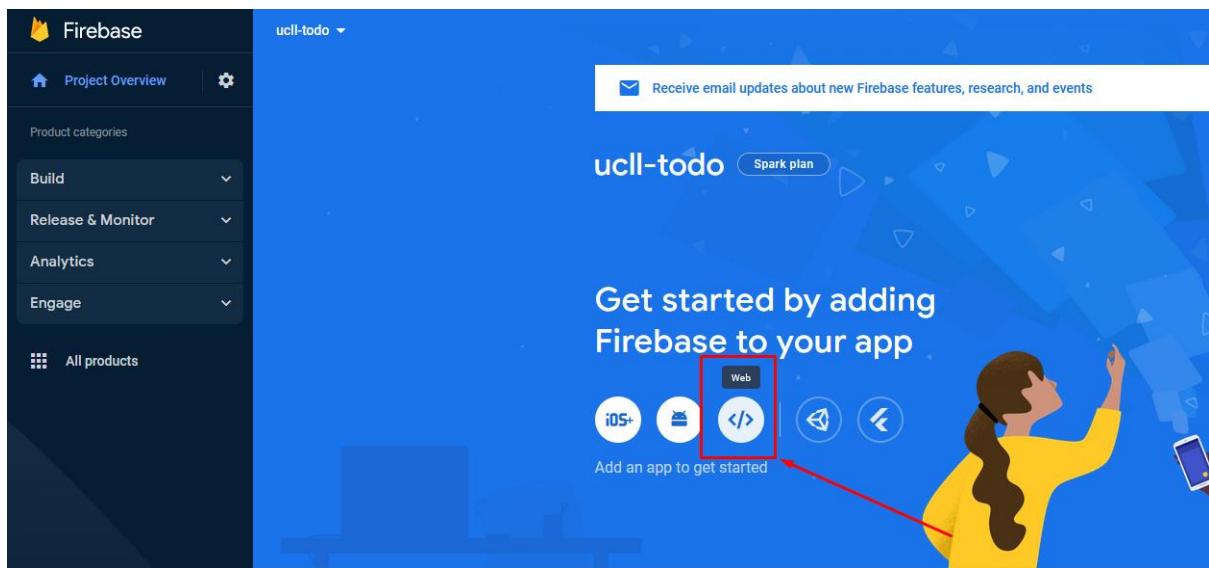
```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <link rel="stylesheet" type="text/css" href="./styles.css">
5
6     <title>UCLL To Do</title>
7   </head>
8   <body>
9     <script src="./code.js"></script>
10  </body>
11 </html>
```

De focus van deze opdracht zal de integratie met Firebase zijn, dus je bent vrij om de applicatie te stijlen zoals je zelf wilt.

## 2.3 Firebase configuratie inladen in frontend

Nu dat we ons Firebase project en onze basis HTML/CSS/Javascript hebben gaan we de 2 met elkaar verbinden. Zo kunnen we via de JS SDK van Firebase bepaalde Cloud functies gebruiken.

Klik op de “Web” knop om je project te configureren voor een website:



Je zal nu een naam moeten kiezen voor je frontend applicatie:

## × Add Firebase to your web app

- 1 Register app**

App nickname ⓘ

☐ Also set up **Firebase Hosting** for this app. [Learn more](#) ⓘ

Hosting can also be set up later. There is no cost to get started anytime.

**Register app**
- 2 Add Firebase SDK**

Firebase hosting gaan we later opzetten, dus vink dit nog niet aan. Ga naar de volgende stap en selecteer “use a <script> tag”. Kopieer de code en zet deze onderaan je <body> in de HTML-pagina.

×

Add Firebase to your web app

✓

Register app

2

Add Firebase SDK

☐

Use npm

☒

Use a <script> tag

Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:

```
<script type="module">
  // Import the functions you need from the SDKs you need
  import { initializeApp } from "https://www.gstatic.com/firebasejs/9.10.0/firebase-app.js";
  // TODO: Add SDKs for Firebase products that you want to use
  // https://firebase.google.com/docs/web/setup#available-libraries

  // Your web app's Firebase configuration
  const firebaseConfig = {
    apiKey: "AIzaSyA...",
    authDomain: "...",
    projectId: "...",
    storageBucket: "...",
    messagingSenderId: "...",
    appId: "..."
  };

  // Initialize Firebase
  const app = initializeApp(firebaseConfig);
</script>
```

Are you using npm and a bundler like webpack or Rollup? Check out the [modular SDK](#).

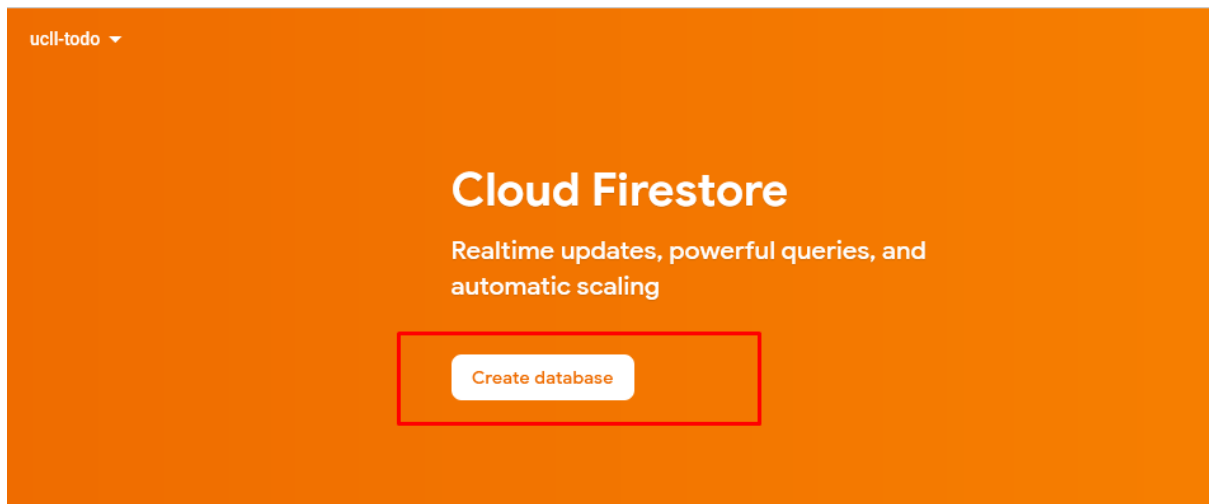
Learn more about Firebase for web: [Get Started](#), [Web SDK API Reference](#), [Samples](#)

Onze webpagina is nu gelinkt aan ons Firebase project, en we kunnen beginnen met het gebruik van de verscheidene diensten die Firebase ons aanbiedt.

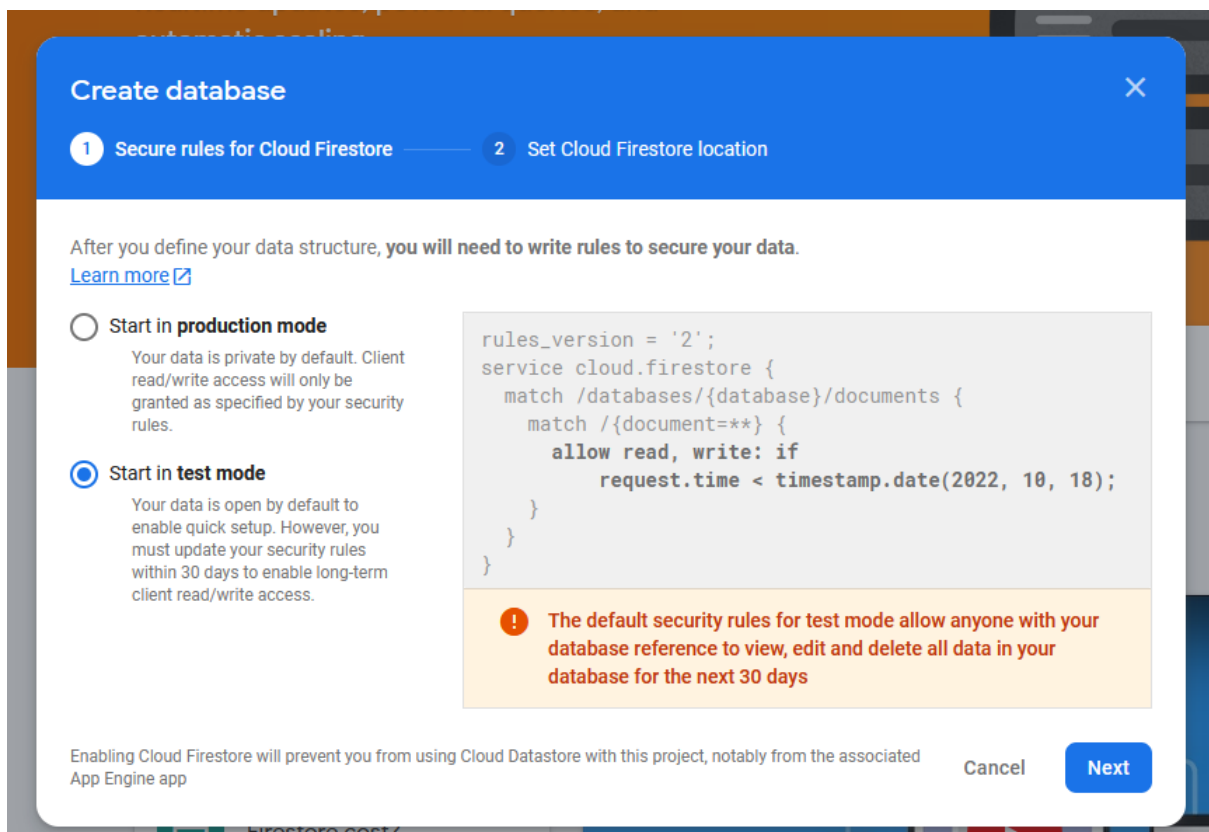
## 3 Ophalen van databankgegevens

### 3.1 todo-items collectie aanmaken in Cloud Firestore

Voor het opslaan van onze To Do items gaan we gebruik maken van Cloud Firestore. Firestore is een NoSQL document database (zoals MongoDB). Navigeer naar de Firestore console in firebase en maak een nieuwe databank aan:



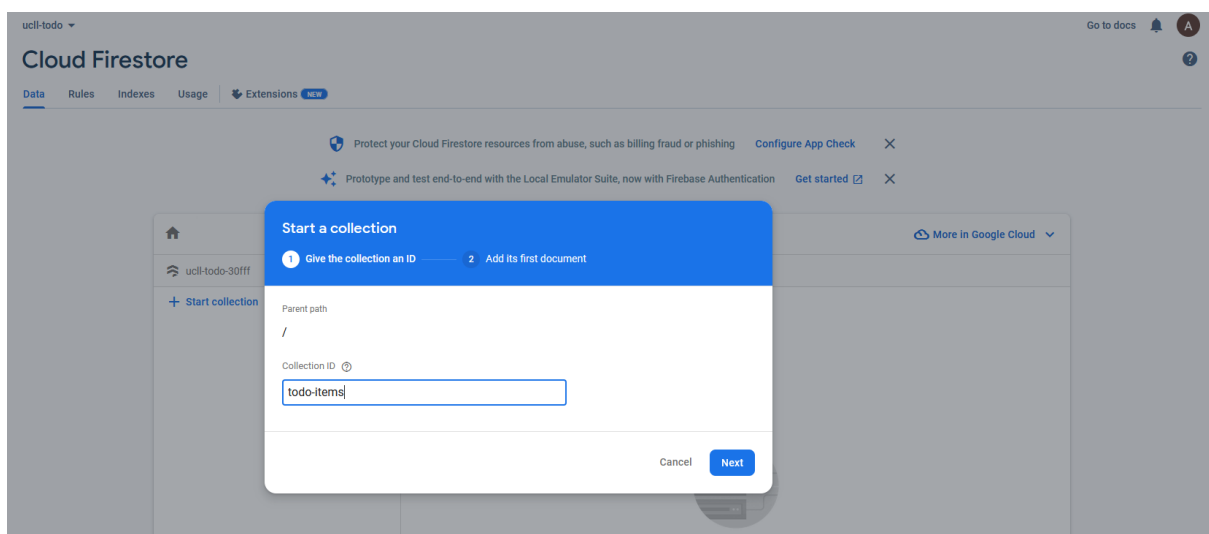
We gaan starten in **test mode**:



In test mode kunnen we vrij documenten opslaan, laden en verwijderen zonder dat we al rekening moeten houden met authenticatie en autorisatie. In volgende stappen zullen we dit toevoegen en onze databank dus beveiligen.

Stel de locatie in op “eur3 (europe-west)”, zo zijn we zeker dat onze data binnen Europa blijft.

Nadat we de databank hebben aangemaakt kunnen we starten met collecties. Voor de eerste stap van onze applicatie gaan we 1 collectie gebruiken: “todo-items”. In deze collectie worden dus al onze To Do items opgeslagen:



De volgende stap is een eerste document aanmaken, zodat onze collectie niet leeg is. Ons eerste To Do item zal 2 velden hebben:

- description
- creationTimestamp

Als ID kan je gewoon “auto ID” aanvinken. Elk document in onze collectie heeft een unieke ID nodig.

De creationTimestamp is gewoon de huidige datetime in UNIX epoch time.



Om de huidige unix time te krijgen kan je de MDN documentatie pagina van het “Date” object doorzoeken.

Start a collection

✓ Give the collection an ID — 2 Add its first document

Document parent path  
/todo-items

Document ID ⓘ  
6W80mapSljN6EelbKe8r

Field	Type	Value
description	string	Create frontend
creationTimestamp	string	1663494864196

Cancel Save

Sla het eerste document op. Je kan nu zoveel documenten toevoegen of verwijderen als je zelf wilt.

ucil-todo-30fff	todo-items	GneZTqrU0zozI1qPK79k
+ Start collection	+ Add document	+ Start collection
todo-items	GneZTqrU0zozI1qPK79k	+ Add field
	0svIW52WYhx47qk3VdDF	creationTimestamp: "1663495098152"
		description: "Load firestore items in frontend"



## 3.2 todo-items ophalen in Frontend

Nu dat we onze collectie hebben in Cloud Firestore, kunnen we de data van deze collectie ophalen en gebruiken in onze Frontend.

We starten met de Firestore SDK in onze Frontend in te laden [volgens de documentatie](#). We voegen ook een `<ul>` toe waar we al onze To Do items in gaan tonen:

```
<ul id="todo-items-list"></ul>

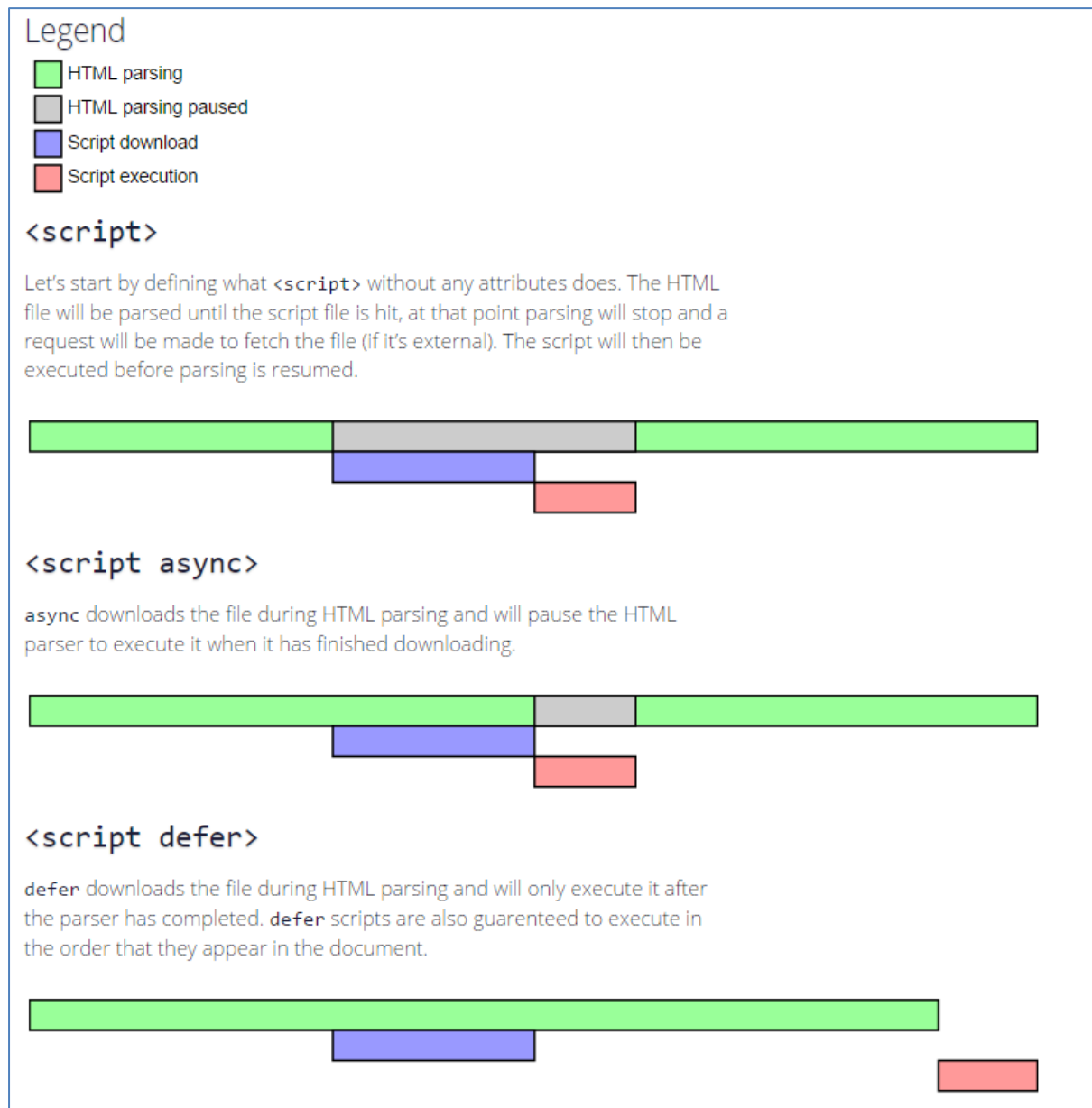
<script type="module">
  // Import the functions you need from the SDKs you need
  import { initializeApp } from "https://www.gstatic.com/firebasejs/9.10.0/firebase-app.js";
  import { getFirestore, collection, query, where, getDocs } from "https://www.gstatic.com/firebasejs/9.10.0/firebase-firestore.js";

  // TODO: Add SDKs for Firebase products that you want to use
  // https://firebase.google.com/docs/web/setup#available-libraries
  // Your web app's Firebase configuration
  const firebaseConfig = {
    apiKey: " ",
    authDomain: " ",
    projectId: " ",
    storageBucket: " ",
    messagingSenderId: " ",
    appId: " "
  };
  // Initialize Firebase
  const app = initializeApp(firebaseConfig);
  const db = getFirestore(window.app);
```

De volgende stap is het [initialiseren van een Firestore object](#) waarmee we kunnen communiceren met Firebase. We gaan ook onze firebase en firestore objecten op het **window** object zetten zodat we deze kunnen gebruiken in andere scripts:

```
28     appId: " "
29   };
30   // Initialize Firebase
31   const app = initializeApp(firebaseConfig);
32   const db = getFirestore(app);
33
34   window.firestore = {
35     collection,
36     query,
37     where,
38     getDocs
39   }
40
41   window.firebase = {
42     isReady: function () {
43       return !!app && !!db;
44     },
45     db,
46     app
47   };
48
49   </script>
50   <script src="./code.js" defer></script>
51 </body>
```

Zet zeker ook het **defer** keyword bij onze eigen script tag, zodat onze code geladen wordt na de firebase code:



Firebase en Firestore zijn opgezet! Nu kunnen we in **code.js** onze eigen code gaan toevoegen om data op te halen en te laten zien in onze webpagina.

Voeg de volgende code toe in **code.js**:

```
index.html JS code.js X
JS code.js > ...
1  (async () => {
2    while(!window && !window.firebase && !window.firebase.isReady()) {
3      // wait for firebase to load
4    }
5    console.log("Firebase is ready!");
6
7    const {
8      collection,
9      getDocs
10   } = window.firestore;
11
12   const {
13     db
14   } = window.firebase;
15
16
17   async function loadTodoItems() {
18     const todoList = document.querySelector("#todo-items-list");
19     const todoResult = await getDocs(collection(db, "todo-items"));
20
21     todoList.innerHTML = "";
22     todoResult.forEach(todoItem => {
23       todoList.innerHTML += `<li>${todoItem.data().description}</li>`;
24     });
25   }
26
27   await loadTodoItems();
28 }());
```

Hier gebeurt heel wat. Heel onze code is gewikkeld in een asynchrone functie die zichzelf uitvoert. Dit doen we zodat we [gebruik kunnen maken van async/await](#). Probeer het zelf eens uit in de console van jouw browser:

```
>> (async () => { console.log("Hello"); })()
Hello
< > Promise { <state>: "fulfilled", <value>: undefined }

>> async function executeSomething() { console.log("Hello"); }
< undefined
>> executeSomething();
Hello
< > Promise { <state>: "fulfilled", <value>: undefined }

>> |
```

Deze 2 voorbeelden doen exact hetzelfde

We bekijken even wat onze code doet:

- Lijn 1 tot 5: We wachten tot onze firebase en firestore objecten volledig ingeladen zijn.
- Lijn 7 tot 14: We halen verschillende objecten uit **window.firestore** en **window.firebase** zodat we ze makkelijk kunnen gebruiken in de rest van onze code. De manier waarop we dit doen heet [object destructuring](#).
- Lijn 17 tot 26: We maken een asynchrone functie aan die data gaat ophalen van firestore en deze in deze gaat laten zien in onze **<ul>** die we eerder in onze HTML-pagina hebben gezet.

Voor meer informatie over de Firebase code kan je alles vinden op de [documentatie website](#).

We zien nu onze To Do items verschijnen in onze webpagina:

- Create frontend
- Load firestore items in frontend

Voeg zelf stijlen toe om de applicatie wat mooier te maken.

**Maak een zip-bestand van je projectmap met bestandsnaam Achternaam-Voornaam.zip met daarin je volledige opdracht in. Laadt dit zip-bestand op naar Toledo.**