

Voor- en achternaam:

Opleiding **Bachelor Toegepaste Informatica, SE**Opleidingsonderdeel **Web Technology**Lectoren **Dieter Mourisse**Academiejaar **2024-2025**

trajectschijf 3

Datum en uur

Proefexamen

EXAMEN

RESULTAAT

/50

ALGEMENE INSTRUCTIES EXAMEN

Dit examen bestaat uit twee delen, een toets op Leho en een laptop gedeelte.

Vul eerst je naam en voornaam in **op alle pagina's** die je gaat indienen. Doe dit voor je aan het examen begint! Alle ruimte niet voorzien voor antwoorden mag gebruikt worden als kladpapier.

Leg identiteits- of studentenkaart klaar ter controle. Bij afgeven **teken je de aanwezigheidslijst af**. Na het indienen van de toets op Leho krijg je de vragen die op de laptop dienen uitgewerkt te worden. Deze **dien je in op LEHO in de voorziene opdracht**.

TOEGELATEN HULPMIDDELEN

Iedere vorm van (online) communicatie tussen studenten of met andere partijen (**inclusief AI chatbots**) is **strikt verboden**. Dit is een individuele opdracht. Iedere vaststelling van onregelmatigheid (o.a. GSM, spieken, afkijken, kopiëren van bestanden van een medestudent, hacking, gebruik van sociale media, ...) wordt conform het OER gemeld aan de betrokken student en aan de voorzitter van de examencommissie.

Mobiele apparaten zijn niet toegelaten bij het examen! Ook niet op zak. Zet ze uit voor je ze opbergt.

VOOR DE LEHO TOETS

Enkel schrijfgerief. NIETS anders. **Geen enkel eigen papier**, bedrukt noch leeg!!! Kladpapier kan gevraagd worden.

VOOR DE VRAGEN OP LAPTOP

Je **mag gebruik maken van je eigen en/of verzamelde materiaal op je laptop**, zonder audio. Daarnaast is het **gebruik van het internet ook toegelaten**.

Communiceren met andere partijen en AI chat systemen is echter **niet toegelaten**. Automatische code completion van bijvoorbeeld copilot is wel toegestaan. Als je twijfelt of je gebruik mag maken van iets, vraag het dan na.

LEHO TOETS

/20

Zie Leho

LAPTOP GEDEELTE

/30

1. THE JAVASCRIPT LANGUAGE

/5

Schrijf de volgende functies in het bestand **question1.js**.

Implementeer deze functies **zo kort mogelijk** en probeer zoveel mogelijk gebruik te maken van **moderne javascript technieken**. Verduidelijk overal in commentaar goed wat er gebeurt.

- Schrijf een functie **calculateTotalCost** die gebruikt kan worden om de totaalprijs van alle producten uit een winkelmandje te berekenen.

```
const shoppingBasket = {Apples: 3.45, Bananas: 4.05, Lemons: 2.00};  
console.log(calculateTotalCost(shoppingBasket)) // 9.5
```

- Schrijf een functie **calculateStatistics** die een array van waarden als parameter meekrijgt. Deze functie berekent het minimum, maximum, som en gemiddelde van alle waarden en geeft dit terug als object.

```
const values = [5, 3, 4];  
  
console.log(calculateStatistics(values));  
// {minimum: 3, maximum: 5, average: 4, total: 12}
```

Voeg, door gebruik te maken van de functie **calculateStatistics** één lijn code op de aangegeven plaats toe zodat je het gewenste resultaat krijgt.

```
const values = [5, 3, 4];  
// voeg hier 1 lijn code toe  
console.log(min, max, total, average); // 3, 5, 12, 4
```

2. OBJECT ORIENTED PROGRAMMING

/ 5

Voor deze oefening zullen we een initiële implementatie (zonder user interface) maken van het spel **de betoverende doolhof (labyrinth)**.

Er staan ook punten op code cleanliness. Zorg dat je je code **opsplitst in meerdere bestanden** in de map *labyrinth*. Je hoeft enkel rekening te houden met data encapsulatie als dat expliciet gevraagd wordt.

Voorzie de klassen **Player**, **Tile**, **Game** en **TeleportationTile** die de volgende functionaliteiten hebben.

Player

Voor een speler hou je enkel de naam en de kleur van de pion bij

Tile

Tegels zijn vierkant en kunnen tussen de 0 en 4 muren hebben (boven/links/rechts/onder). Dit kan voorgesteld worden als een array van booleans.

```
const walls = [true, false, false, true];
```

Op deze manier kun je bijvoorbeeld zeggen dat een tegel een muur bovenaan en links heeft.

Naast muren kan een tegel ook een schat (**treasure**) bevatten. De mogelijke **treasures** zijn

```
const treasures = ["Bag of Gold Coins", "Bat", "Book with Clasp", "Dragon", "Ghost (in bottle)", "Ghost (waving)", "Gold Crown"];
```

Voorzie daarnaast ook een manier om een random tegel te genereren. Om random muren en een random treasure te kiezen kun je onderstaande code gebruiken.

```
const walls = [Math.random() < 0.2, Math.random() < 0.2, Math.random() < 0.2, Math.random() < 0.2];  
const treasure = Math.random() > 0.5 ? undefined : treasures[Math.floor(Math.random() * treasures.length)];
```

Game

Bij het aanmaken van een **game** geef je mee uit hoeveel rijen en kolommen het spelbord moet bestaan. Vervolgens wordt er een doolhof (**maze**) aangemaakt dat bestaat uit het **opgegeven aantal rijen en kolommen**. Ieder vakje van het doolhof bevat een random gegenereerde tegel.

Zorg ervoor dat je aan een game kunt vragen **uit hoeveel rijen en kolommen het doolhof bestaat**, maar dat je deze waarden niet kunt aanpassen buiten de klasse **Game** zelf.

Naast het doolhof bevat de klasse ook voor iedere speler de positie op het bord. Een speler wordt toegevoegd aan het spel via een methode waaraan je de **naam en de pionkleur van die speler meegeeft**. Een speler wordt initieel altijd op positie **[0, 0]** geplaatst.

Teleportation Tile

Voorzie ook functionaliteit voor een speciaal soort teleportatietegel. Een teleportatietegel heeft dezelfde eigenschappen als een **Tile** uitgebreid met een extra eigenschap, de tegel waarnaar je geteleporteerd wordt als je op deze tegel terecht komt.

3. ASYNCHRONOUS PROGRAMMING

/10

De gegeven server (zie server map) bevat enkele endpoints die nodig zijn voor deze oefening:

- **/ts/albums** die een lijst van alle Taylor Swift albums teruggeeft
- **/ts/albums/{id}** die de tracks op een album teruggeeft
- **/ts/songs/{id}** die de details van een song teruggeeft

Je zal de volgende functies moeten implementeren:

- Een functie **getAlbumTitles** die de titels (en enkel de titels) van ieder album teruggeeft.
- Een functie **getDetailsOfFirstOfAlbum** die de details van de eerste track van een album teruggeeft.
- Een functie **getFullAlbum** die de details van alle tracks van een album teruggeeft.
- Een functie **getAveragePopularityForAlbum** die de gemiddelde populariteit van alle tracks van het album teruggeeft.
- Een functie **getMostPopularAlbum** die het meest populaire album teruggeeft.

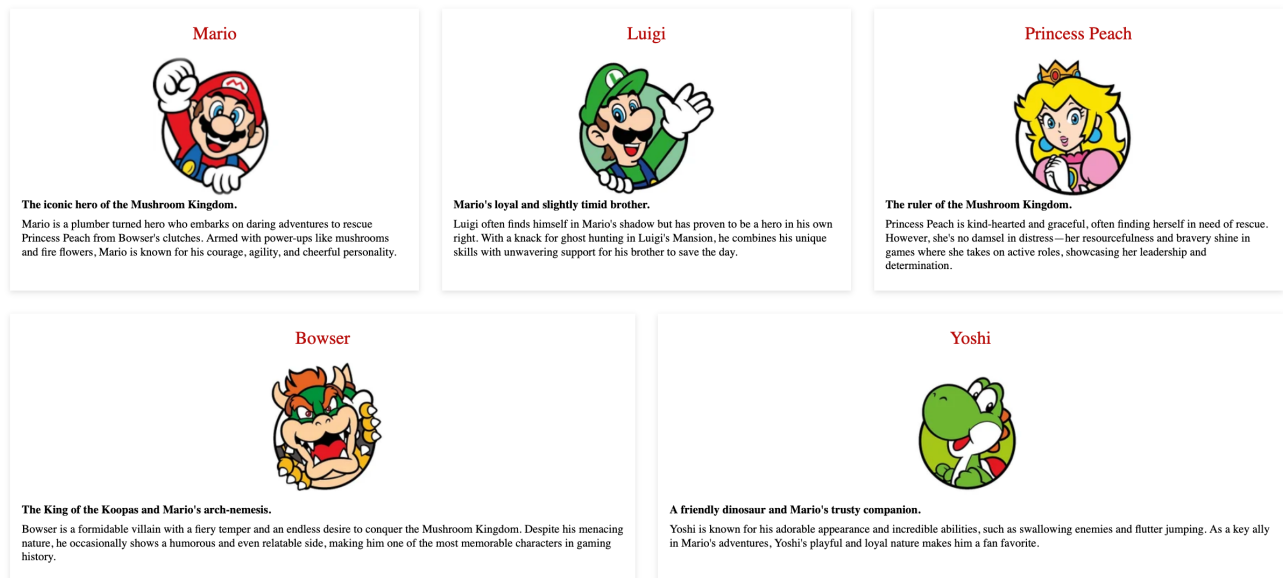
In `taylor-swift/question3a.js` implementeer je deze functies **zonder** gebruik te maken van `async/await`. In `taylor-swift/question3b.js` implementeer je dezelfde functies maar maak je zo weinig mogelijk van **then**.

Om deze functies te implementeren zal je het **node-fetch package** nodig hebben, hiervoor zul je eerst het `npm install` commando moeten uitvoeren in de `taylor-swift` directory.

4. RESPONSIVE DESIGN / SASS / TYPESCRIPT

/10

Voor deze oefening zul je een **responsive website** maken met behulp van **Sass** en **TypeScript**. Je schrijft dus zelf geen CSS/JavaScript. Het eindresultaat moet er zo uit zien:

Super Mario Characters

Enkele zaken om rekening mee te houden

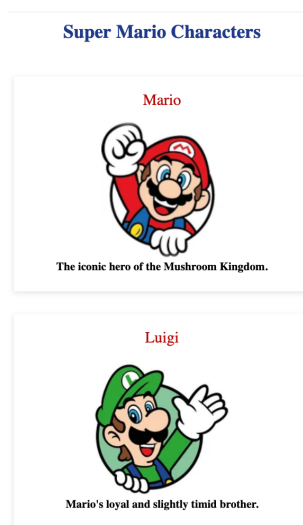
- Je opent enkel de super-mario map in je editor
- De *package.json* en *tsconfig.json* zijn al gegeven, vergeet niet om een **npm install** te doen
- Sass files voeg je toe in de sass map, TypeScript files in de ts map
- Sass moet compileren naar *assets/css/style.css*, de *tsconfig.json* zorgt er voor dat de TypeScript omgezet wordt naar *assets/js/index.js*. Deze files zijn al opgenomen in de html
- Voeg overal waar kan **types** toe aan je TypeScript

Sass (/ 3)

- Je maakt gebruik van **tenminste 3** sass bestanden *main.scss*, *article.scss* en *constants.scss*.
- De html pagina bevat al wat stijlregels, voeg deze toe aan de sass bestanden.
- **h1** titels moeten de kleur **#1E3888** krijgen, **h2** titels **#B80C09**. Sla deze twee kleuren op in een **variabele** in *constants.scss*.

Responsive Design (/ 5)

- Voor de **fontgrootte** van de *h1* en *h2* titels maak je gebruik van een **modulaire schaal** die zich aanpast aan de grootte van de viewport.
- Probeer zoveel mogelijk *article* elementen op dezelfde rij te krijgen. Ze moeten een **minimumbreedte van 30rem** hebben, maar wanneer er extra plaats is mogen ze groter worden zodat de volledige viewportbreedte in beslag genomen wordt.
- Wanneer het volledige article niet meer op het scherm past, mag het krimpen maar dan moet de tekst binnen de **laatste p tag wegvallen**.



TypeScript (/ 2)

De verschillende articles staan momenteel **statisch** in de html. Ze kunnen echter ook opgehaald worden via het endpoint `/supermario/characters` van de gegeven server.

Haal ze daar op en gebruik de **template tag** met id `character-template` om ze **dynamisch** toe te voegen aan de webpagina.

De articles die al in de html staan mogen verwijderd worden.