

NAAM: _____Gülümsu Sancak_____

DATUM: 25/11/2024

VDO FULL STACK DEVELOPERS

EXAMEN ADVANCED FRONTEND DEEL 1 - 50%

ZOEK DE FOUT

Oefening 1: "Even of Oneven?"

Opgave:

```
const number = 7;

if (number % 2 = 0) {
  console.log("Het getal is even.");
} else {
  console.log("Het getal is oneven.");
}
```

Antwoord:

```
const number = 7;

if (number % 2 === 0) {
  console.log("Het getal is even.");
} else {
  console.log("Het getal is oneven.");
}
```

Oefening 2: "Willekeurig Getal in een Bereik"

Opgave:

```
const min = 5;
const max = 15;
const randomNumber = Math.random * (max - min) + min;

console.log("Willekeurig getal tussen 5 en 15:", randomNumber);
```

Antwoord:

```
const min = 5;
const max = 15;
const randomNumber=Math.floor(Math.random() * (max - min) + min) ;

console.log("Willekeurig getal tussen 5 en 15:", randomNumber);
```

Oefening 3: "Som van Getallen in een Array"

Opgave:

```
const numbers = [1, 2, 3, 4, 5];
let sum = 0;

for (let i = 0; i < numbers.length; i++) {
  sum = numbers[i];
}
console.log("De som is:", sum);
```

```
const numbers = [1, 2, 3, 4, 5];
let sum = 0;
let sumNumbers=[];

for (let i = 0; i < numbers.length; i++) {
  sum = numbers[i];
}
console.log("De som is:", sum);

sumNumbers.push(sum);
console.log(sumNumbers);
```

RECEPTENBOEK

Basisclass Recept

De Recept-class beschrijft algemene recepten met basisinformatie.

Tip: je hebt in deze oefening GEEN set of get nodig.

Eigenschappen:

1. naam - Naam van het recept (bijv. "Spaghetti Bolognese").
2. ingrediënten - Lijst van ingrediënten.
3. bereidingstijd - Bereidingstijd in minuten.

Methodes:

1. getDetails() - Retourneert een string met de details van het recept.
2. voegIngrediëntToe(ingrediënt) - Voegt een ingrediënt toe aan de lijst.
3. verwijderIngrediënt(ingrediënt) - Verwijdert een ingrediënt uit de lijst.

Code

```
class Recept {  
  //code  
}
```

```
// Test de basisclass
```

```
const spaghetti = new Recept('Spaghetti Bolognese', ['Spaghetti',  
'Tomatensaus', 'Gehakt'], 30);  
console.log(spaghetti.getDetails()); // Details over het recept  
spaghetti.voegIngrediëntToe('Parmezaanse kaas');  
console.log(spaghetti.getDetails()); // Parmezaanse kaas toegevoegd  
spaghetti.verwijderIngrediënt('Gehakt');  
console.log(spaghetti.getDetails()); // Gehakt verwijderd
```

Subclasses

We maken twee subclasses:

1. **Nagerecht** - Voor desserts, met een extra eigenschap zoetheidsniveau.
2. **Hoofdgerecht** - Voor hoofdgerechten, met een eigenschap porties.

Code

```
// Subclass voor Nagerechten  
//code
```

```
// Test de subclasses
```

```
const taart = new Nagerecht('Chocoladetaart', ['Bloem', 'Suiker',  
'Chocolade', 'Eieren'], 60, 8);  
console.log(taart.getDetails()); // Details van de taart  
  
const curry = new Hoofdgerecht('Thaise Curry', ['Rijst', 'Kokosmelk',  
'Groenten'], 40, 4);  
console.log(curry.getDetails()); // Details van de curry
```

Receptenboek

De class Receptenboek beheert alle recepten en biedt functionaliteit om recepten toe te voegen, te zoeken, te verwijderen en te sorteren.

Eigenschappen:

1. recepten - Een array van recepten.

Methodes:

1. voegReceptToe(recept) - Voegt een nieuw recept toe.
2. toonRecepten() - Geeft een lijst van alle recepten.
3. zoekReceptOpNaam(naam) - Zoekt naar een recept op naam. Tip: find
4. verwijderRecept(naam) - Verwijdert een recept uit het boek. Tip: filter
5. sorteerReceptenOpTijd() - Sorteert recepten op bereidingstijd. Tip: sort

Code

```
// Class voor het beheren van recepten
```

```
class Receptenboek {  
  //code  
}
```

```
// Test het receptenboek
```

```
const receptenboek = new Receptenboek();  
receptenboek.voegReceptToe(spaghetti);  
receptenboek.voegReceptToe(taart);  
receptenboek.voegReceptToe(curry);
```

```
console.log('--- Alle recepten ---');  
receptenboek.toonRecepten();
```

```
console.log('--- Zoek "Thaise Curry" ---');  
const gevondenRecept = receptenboek.zoekReceptOpNaam('Thaise Curry');  
console.log(gevondenRecept ? gevondenRecept.getDetails() : 'Recept niet gevonden.');
```

```
console.log('--- Sorteer op bereidingstijd ---');  
receptenboek.sorteerReceptenOpTijd();  
receptenboek.toonRecepten();
```

```
console.log('--- Verwijder "Spaghetti Bolognese" ---');  
receptenboek.verwijderRecept('Spaghetti Bolognese');  
receptenboek.toonRecepten();
```

Opgave: Pokémon Filter App (met Axios)

Doel van de opdracht:

- Maak een dynamische webapplicatie die:
 1. Een lijst van Pokémon toont in **Bootstrap-cards**.
 2. Een filter biedt via een **dropdown-menu** op basis van types.
 3. Bij het klikken op een card een **modal** opent met gedetailleerde informatie over de Pokémon.

Wat is al gegeven?

- De HTML-structuur inclusief Bootstrap-styling is al geschreven.
- Axios is via een `<script>`-tag in de HTML opgenomen:

```
<script  
src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

Stappenplan: Hoe ga je te werk?

1. Laad de basisgegevens van de Pokémon API

- Gebruik **axios** om de basislijst van Pokémon op te halen via het endpoint:
`https://pokeapi.co/api/v2/pokemon?limit=50`
- **Doel:** Haal een lijst van 50 Pokémon op.
- De bootstrap code voor het beginstuk krijg je hier:

```
<!DOCTYPE html>  
  
<html lang="nl">  
  
<head>  
  
  <meta charset="UTF-8">  
  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  
  <title>Pokémon Filters</title>  
  
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css">  
  
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons/font/bootstrap-icons.css">  
  
  <style>  
  
    body {  
  
      font-family: Arial, sans-serif;  
  
      margin: 20px;  
  
    }  
  
    .pokemon-card img {  
  
      max-height: 150px;  
  
      object-fit: contain;  
  
    }  
  
    .filter-container {
```

```

    margin-bottom: 20px;
  }
</style>
</head>
<body>
  <div class="container">
    <h1 class="text-center mb-4">Pokémon Filters</h1>

    <!-- Filteropties -->
    <div class="filter-container">
      <label for="type-filter" class="form-label"><strong>Filter op type:</strong></label>
      <select id="type-filter" class="form-select">
        <option value="">Alle types</option>
        <!-- Dynamische opties komen hier -->
      </select>
    </div>

    <!-- Pokémon Cards -->
    <div id="pokemon-list" class="row g-3">
      <!-- Dynamische inhoud hier -->
    </div>
  </div>

  <!-- Modal voor details -->
  <div class="modal fade" id="pokemonModal" tabindex="-1" aria-labelledby="pokemonModalLabel" aria-hidden="true">
    <div class="modal-dialog modal-lg">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="pokemonModalLabel">Pokémon Details</h5>
          <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Sluiten"></button>
        </div>
        <div class="modal-body">
          <div class="text-center">

```

```

<img id="modal-pokemon-img" src="" class="img-fluid mb-3" alt="Pokémon afbeelding"
style="max-height: 200px;">

</div>

<h4 id="modal-pokemon-name" class="text-center"></h4>

<ul class="list-group">

  <li class="list-group-item"><strong>Type(s):</strong> <span id="modal-pokemon-
types"></span></li>

  <li class="list-group-item"><strong>Hoogte:</strong> <span id="modal-pokemon-height"></span>
m</li>

  <li class="list-group-item"><strong>Gewicht:</strong> <span id="modal-pokemon-
weight"></span> kg</li>

  <li class="list-group-item"><strong>Statistieken:</strong>

    <ul id="modal-pokemon-stats" class="mt-2"></ul>

  </li>

</ul>

</div>

</div>

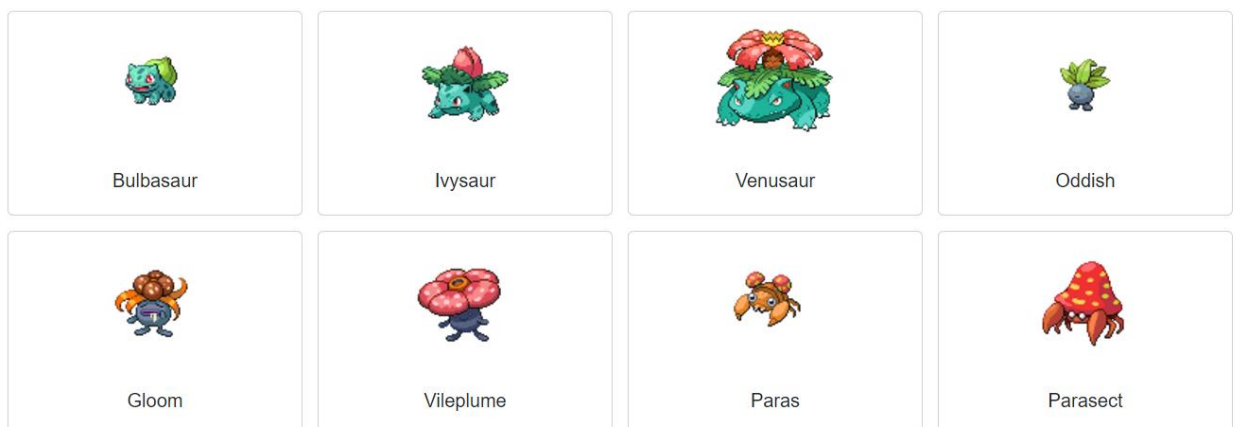
</div>

</div>

```

2. Toon alle Pokémon in de cards

- Maak een functie `renderPokemonCards(pokemonList)` om de Pokémon weer te geven in **Bootstrap-cards**.
- **Doel:** Toon elke Pokémon met:
 - Een afbeelding.
 - De naam van de Pokémon.

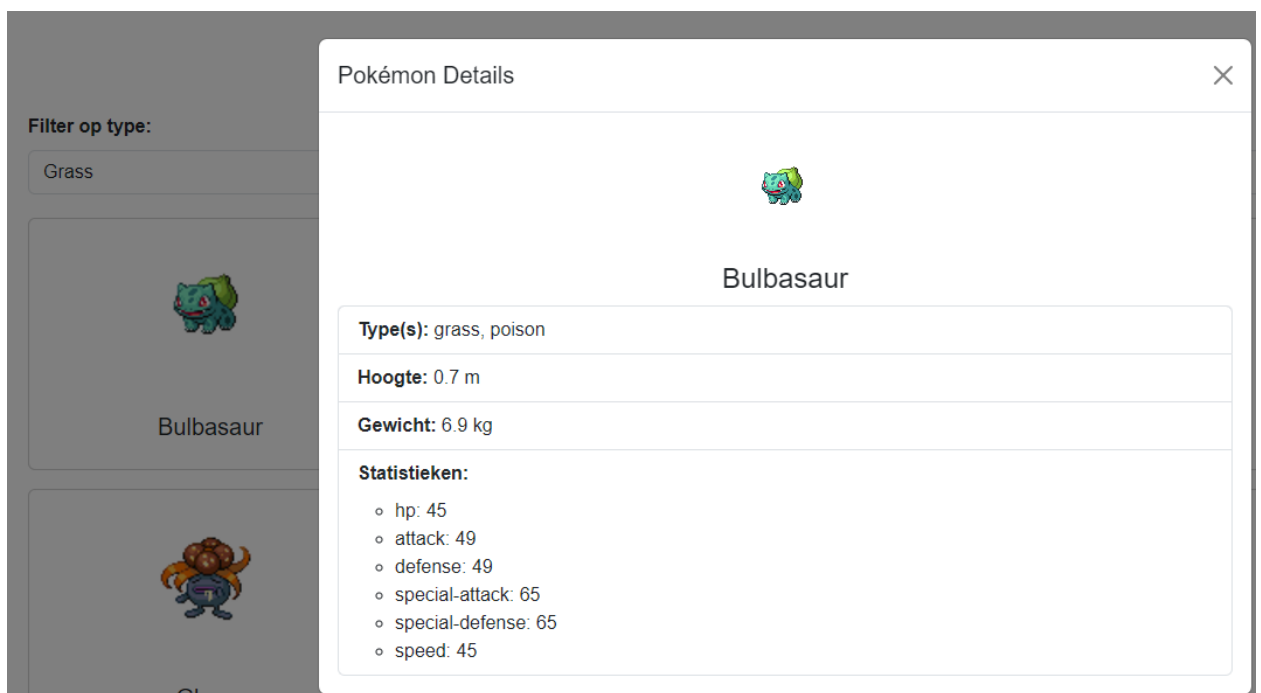


3. Haal de details op voor elke Pokémon

- De basis-API geeft alleen de naam en URL naar gedetailleerde data.
- Gebruik deze URL's in combinatie met `axios.get()` om de details van elke Pokémon op te halen.
- **Doel:** Voor elke Pokémon haal je de volgende gegevens op:
 - Naam
 - Afbeelding (sprite)
 - Types
 - Hoogte
 - Gewicht
 - Statistieken

4. Maak de modal functioneel

- Voeg een klikbare functionaliteit toe aan de cards.
- **Doel:** Open een **modal** met details over de Pokémon.
- **Wat moet erin staan:**
 - Afbeelding.
 - Naam.
 - Types.
 - Hoogte.
 - Gewicht.
 - Statistieken.
- **Wat te doen:**
 1. Voeg een `click`-event listener toe aan elke card.
 2. Vul de modal dynamisch met de details van de aangeklikte Pokémon.
 3. Open de modal.



5. Voeg een filteroptie toe (grass, poison, fire,...)

- Maak een **dropdown** met types als filteroptie.
- **Doel:** Toon alleen Pokémon van een bepaald type.
- **Wat te doen:**
 1. Verzamel alle unieke types.
 2. Genereer de opties in de **dropdown**.
 3. Voeg een change-event listener toe om de lijst van Pokémon te filteren.

Filter op type:

Grass

Alle types

Grass

Poison

Fire

Flying

Water





Bug

Normal

Electric

Ground

Fairy

			
Gloom	Vileplume	Paras	Parasect

Veel succes!