Cursus Robbe Magerman Web Dev II

Robbe Magerman

14/05/2024

Inhoudstafel

1 Hoofdstuk 5: Functioneel Programmeren

1.1 Arrays

1.1.1 ForEach

- Retourneert een nieuwe array en verloopt alle elementen uit een array op een makkelijke manier
- Heeft 3 waarden: element, index en de array zelf

```
const animals = [
  { name: "cat", size: "small", weight: 5 },
  { name: "dog", size: "small", weight: 10 },
  { name: "lion", size: "medium", weight: 150 },
  { name: "elephant", size: "big", weight: 5000 },
];

// ================================================ //

animals.forEach((dier, i, array) => {
  console.log(
    `Van de array animals is dit dier ${i}/${array.length} met naam: ${dier}`
  );
});
```

1.1.2 Filter

- Retourneert een nieuwe array met een bepaalde voorwaarde
- Heeft 3 waarden: element, index en de array zelf

```
const underTonAnimals = animals.filter(
  (a) => a.weight >= 0 && a.weight <= 1000
);
// OF //
const underTonAnimalsBis = animals.filter(
  ({ weight }) => weight >= 10 && weight <= 1000
);
```

1.1.3 Map

- Retourneert een nieuwe array met specifieke elementen/data uit een array
- Heeft 3 waarden: element, index en de array zelf

```
// VB - maak een array met de namen van alle animals
const namesOfAnimals = animals.map(({ name }) => name);
console.log(`Names of animals: ${namesOfAnimals}`);

// DIY - maak een array met 'naam: size' van alle animals
const namesOfAnimalsWithSize = animals.map(
  ({ name, size }) => `${name}: ${size}`
);
```

1.1.4 Reduce

- Retourneert een nieuwe array, wordt gebruikt voor een berekening
- Heeft 3 waarden: element, index en de array zelf

```
// VB - log het totale gewicht van alle animals naar de console
```

```
console.log(
  `Total weight of animals = ${animals.reduce(
    (pv, { weight }) => pv + weight,
    0
  )}`
);

// DIY - hoeveel animals hebben size die small is
const numberOfSmallAnimals = animals.reduce(
  (previousElement, element) => (element.size === "small" ? pv + 1 : pv),
  0
);
```

1.1.5 Find(FindIndex)

- Zoekt een specifiek iets in een array

```
const users = [
  {
    id: 1,
    firstname: "Jan",
    lastname: "Janssens",
  },
  {
    id: 2,
    firstname: "Eva",
    lastname: "De Smet",
  },
  {
    id: 3,
    firstname: "Pieter",
    lastname: "Martens",
  },
];

const user = users.find((item) => item.id === 1);
console.log(user);

const indexuser = users.findIndex((item) => item.id === 1);
console.log(indexuser); // 0
```

1.1.6 Sort

- Retourneert een nieuwe array en soorteert op alfabetische volgorde
- De manier hoe je sorteert kan je ook aanpassen

```
console.log(fruit.sort());
// OF //
fruit.sort((a, b) => a.length - b.length);
console.log(fruit);
```

1.2 Maps en Sets

1.2.1 Maps

- Een map houdt key-waarden bij (niet per se uniek)

```
// een map maken en waarden in stoppen
let mijnMap = new Map([
  ["a", 1],
  ["b", 2],
  ["c", 3],
]);

// waarden toevoegen aan een map
mijnMap.set("d", 4);

// Waarde ophalen uit een map op basis van de sleutel
console.log(mijnMap.get("a")); // geeft 1 terug

// Controleren of een sleutel in de map zit
console.log(mijnMap.has("b")); // geeft true terug

// Sleutel verwijderen uit een map
mijnMap.delete("c");

// De grootte van de map ophalen
console.log(mijnMap.size); // geeft het aantal sleutel-waardeparen in de map
    terug

// Itereren over de sleutels van de map
for (let key of mijnMap.keys()) {
  console.log(key);
}

// Itereren over de waarden van de map
for (let value of mijnMap.values()) {
  console.log(value);
}

// Itereren over zowel de sleutels als de waarden van de map
for (let [key, value] of mijnMap.entries()) {
  console.log(key, value);
}
```

1.2.2 Sets

- Een set houdt unieke key-waarden bij

```
// Een lege set maken
let mijnSet = new Set();

// Een set maken met enkele waarden
let mijnAndereSet = new Set([1, 2, 3, 4, 5]);

// Elementen toevoegen aan een set
mijnSet.add(1);
mijnSet.add(2);
mijnSet.add(3);

// Elementen verwijderen uit een set
mijnSet.delete(2);

// Controleren of een element in de set zit
```

```
console.log(mijnSet.has(1));

// De grootte van de set ophalen
console.log(mijnSet.size);

// De set omzetten naar een array
return [...new Set(this.#producten.map((el) => el.categorie))];
```

2 Hoofdstuk 6: Webopslag

2.1 Data opslaan in de browser

2.1.1 get...FromStorage() via een id van de HTML dat *niet* in een String moet worden gezetn

```
#getJaarFromStorage() {
    if (this.#storage.getItem("jaarGefietsteKilometer")) {
      document.getElementById("jaar").value = this.#storage.getItem(
        "jaarGefietsteKilometer"
      );
    }
  }
```

2.1.2 set...InStorage() via een id van de HTML dat *niet* in een String moet worden gezet

```
#setJaarInStorage() {
    this.#storage.setItem(
      "jaarGefietsteKilometer",
      document.getElementById("jaar").value
    );
  }
```

2.1.3 set...InStorage() via een id van de HTML dat *wel* in een String moet worden gezet

```
#getJaarFromStorage() {
    if (this.#storage.getItem("jaarGefietsteKilometer")) {
      document.getElementById("jaar").value = this.#storage.getItem(
        "jaarGefietsteKilometer"
      );
    }
  }
```

2.1.4 set...InStorage() via een id van de HTML dat *wel* in een String moet worden gezet

```
#setJaarInStorage() {
    this.#storage.setItem(
      "jaarGefietsteKilometer",
      document.getElementById("jaar").value
    );
  }
```

3 Hoofdstuk 7: DOM

3.1 HTML tags maken via JavaScript

3.1.1 Aan een bestande tag extra htmlcode toevoegen

```
#categorieenToHtml(categorieen) {
  categorieen.forEach((c) =>
  document.getElementById("categorie").insertAdjacentHTML("beforeend", `<option
      value=${c}>${c}</option>`));
}
```

3.1.2 In een repository iets toevoegen

```
  voegVacatureToe(id, titel, functieomschrijving, profiel, bedrijf, plaats) {
    return this.#vacatures.push(
      new Vacature(id, titel, functieomschrijving, profiel, bedrijf, plaats)
    );
  }
```

3.1.3 Inhoud van een bestaande volledig vervangen van een htmlElement

```
  #productenToHtml(producten) {
    document.getElementById(
      "aantalProducten"
    ).innerHTML = `<h4>Aantal producten: ${producten.length}</h4>`;
  }
```

3.1.4 Een volledig nieuwe tag aanmaken

```
// element creeren + wijzigingen aanbrengen
// dit kan ook gemaakt worden met insertAdjacentHTML, je mag op het examen
   kiezen
producten.forEach((p) => {
  const divEl = document.createElement("div");
  divEl.setAttribute("id", p.id);
  const imgEl = document.createElement("img");
  imgEl.src = `images/${p.id}/thumbs/thumb_${p.afbeeldingen[0]}.jpg`;
  imgEl.alt = p.titel;
  divEl.appendChild(imgEl);
  const pEl = document.createElement("p");
  pEl.innerText = p.titel;
  divEl.appendChild(pEl);
  divEl.onclick = () => {
    this.#productDetailsToHtml(p);
    divEl.classList.add("tekstVet");
  };
  document.getElementById("overzichtProducten").appendChild(divEl);
});
```

3.1.5 Css classe toevoegen

```
document.getElementById("productDetails").classList.add("verbergen");
```

3.1.6 QuerySelector

4 Hoofdstuk 8: Asynchroon programmeren

4.1 Synchroon vs Asynchroon

- Synchroon:
  - De éne stap gebeurt na de andere
- Asynchroon:
  - Er gebeuren meerdere stappen tegelijk (API-requesten kunnen soms meerdere seconden duren waardoor de website lang zal laden en de gebruiker kan dan niks doen)

4.2 setTimeout(function() => {}, 3000)

- Deze *callback*-functie wordt dan 3 seconden uitgevoerd

4.3 AJAX (Asyncrhoon JavaScript And XML)

- Maakt gebruikt van HTML en CSS voor de presentatie

4.4 Data ophalen via een API

```
function getData() {
  fetch("https://v2.jokeapi.dev/joke/Any?")
    .then((response) => {
      if (!response.ok) {
        throw new Error("fetching joke failed");
      }
      return response.json();
    })
    .then((json) => {
      toHtml(json);
    });
}
```

4.5 Promises - Single

```
fetch("./data/data1.json")
  .then((response) => {
    if (!response.ok) {
      throw new Error(`HTTP error: ${response.status}`);
    }
    return response.json();
  })
  .then((json) => {
    console.log("name:", json.name);
    console.log("age:", json.age);
  })
  .catch((error) => alert(error));
```

4.6 Promises - Promise.all

```
const promiseArray = [
  fetch("../data/part1.txt").then((response) => response.text()),
  fetch("../data/part2.txt").then((response) => response.text()),
  fetch("../data/part3.txt").then((response) => response.text()),
];
```

```
Promise.all(promiseArray).then((array) => {
  boektekst.insertAdjacentText("beforeend", array[0]);
  boektekst.insertAdjacentText("beforeend", array[1]);
  boektekst.insertAdjacentText("beforeend", array[2]);
});
```

4.7 Async - Await

```
  async #getData() {
    try {
      const response = await fetch(this.#url);
      if (!response.ok) {
        throw new Error(
          `Failed fetching trivias, reason ${response.statusText}`
        );
      }
      const jsonResponse = await response.json();
      this.#triviaRepository.addTrivias(json);
      console.log(jsonResponse);
    } catch (error) {
      alert(error);
    }
  }
```