

# JS - (DOM) - Extra

## Gegevens bewaren met *localStorage*

### Doelstellingen

- Je weet wat `localStorage` is en wanneer je het gebruikt.
- Je kunt gegevens opslaan in de browser.
- Je kunt opgeslagen gegevens bij het laden van de pagina weer tonen.

### Uitleg

`localStorage` is een opslagruimte in de browser. Alles wat je daarin zet, blijft bewaard – ook als je de pagina sluit of opnieuw opent.

Je gebruikt het bijvoorbeeld zo:

```
// iets opslaan
localStorage.setItem("naam", "Ali");

// iets ophalen
const naam = localStorage.getItem("naam");

// iets verwijderen
localStorage.removeItem("naam");
```

Let op: je kunt alleen strings opslaan. Wil je een lijst opslaan? Gebruik dan `JSON.stringify()` en `JSON.parse()` :

```
const lijst = ["bananen", "appels"];
localStorage.setItem("boodschappen", JSON.stringify(lijst));

const terug = JSON.parse(localStorage.getItem("boodschappen"));
console.log(terug); // ["bananen", "appels"]
```

## Opdracht – Opslaan wat je invult

1. Maak een bestand `dom9.html`.
2. Maak een invoerveld waar de gebruiker een hobby, taak of naam kan invullen.
3. Als de gebruiker iets toevoegt, verschijnt het in een lijst op de pagina.
4. De lijst moet bewaard blijven via `localStorage` zodat deze zichtbaar blijft bij herladen.
5. Bonus: Voeg een knop toe om alles te wissen (via `localStorage.clear()`).

### Tips:

- Lees bij het laden van de pagina eerst de gegevens uit `localStorage`.
- Update `localStorage` telkens als je iets toevoegt of verwijdt.

## Reflectie

- Wat is het voordeel van `localStorage`?
- Waarom moet je JSON gebruiken bij het opslaan van lijsten?
- Wat zou je nog meer kunnen opslaan in een webapp?

## Inleveren

- Lever je bestand `dom9.html` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.
- Toon in een screenshot dat je lijst bewaard blijft bij herladen.

# Gegevens ophalen met fetch()

## Leerdoelen

- Je weet wat `fetch()` doet in JavaScript.
- Je kunt externe gegevens ophalen en tonen op een webpagina.
- Je begrijpt hoe je met JSON-data werkt en deze verwerkt met de DOM.

## Uitleg

Met `fetch()` kun je gegevens ophalen van een externe bron zoals een API. Vaak krijg je dan JSON-data terug: een soort tekstversie van een JavaScript-object of array.

Een voorbeeld:

```
fetch("https://jsonplaceholder.typicode.com/users")
  .then(response => response.json())
  .then(data => {
    console.log(data); // Hier kun je nu iets mee doen
  });
```

Je kunt daarna bijvoorbeeld een lijst maken van namen:

```
fetch("https://jsonplaceholder.typicode.com/users")
  .then(res => res.json())
  .then(users => {
    users.forEach(user => {
      const p = document.createElement("p");
      p.textContent = user.name;
      document.body.appendChild(p);
    });
  });
```

## Opdracht – Externe gebruikerslijst

1. Maak een bestand `dom10.html`.

2. Haal gegevens op van `https://jsonplaceholder.typicode.com/users`.
3. Laat van elke gebruiker de naam en e-mailadres zien in de browser.
4. Maak van elke gebruiker een eigen `<div>` of `<li>`.
5. Bonus: laat de gegevens pas zien als je op een knop “Laad gebruikers” hebt geklikt.

## Extra uitdaging:

- Voeg bij elk item een knop “verwijder” toe waarmee dat item uit de DOM verdwijnt.

## ☐☐ Reflectie

- Wat doet `fetch()` precies?
- Wat is een API, en wat kun je ermee?
- Wat zou een risico zijn als je data van andere websites gebruikt?

## ☐☐ Inleveren

- Lever je bestand `dom10.html` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand.
- Lever een screenshot aan waarop de opgehaalde gebruikers zichtbaar zijn in je browser.

## ☐☐ *Lijsten filteren op basis van invoer*

## ☐☐ Leerdoelen

- Je weet hoe je gebruikersinvoer gebruikt om iets te filteren.
- Je kunt elementen verbergen of tonen met JavaScript.
- Je past een `input`-event toe om live te reageren.

## ☐☐ Uitleg

Je kunt met JavaScript elementen tonen of verbergen op basis van wat de gebruiker intypt.

Voorbeeld – een zoekveld dat een lijst filtert:

```
<input type="text" id="zoekveld" placeholder="Zoek een dier...">

<ul id="dierenlijst">
  <li>Hond</li>
  <li>Kat</li>
  <li>Papegaai</li>
  <li>Vogelbektier</li>
</ul>

<script>
  const zoekveld = document.getElementById("zoekveld");
  const items = document.querySelectorAll("#dierenlijst li");

  zoekveld.addEventListener("input", function() {
    const tekst = zoekveld.value.toLowerCase();
    items.forEach(function(item) {
      const inhoud = item.textContent.toLowerCase();
      item.style.display = inhoud.includes(tekst) ? "list-item" : "none";
    });
  });
</script>
```

## Opdracht – Live filter maken

1. Maak een bestand `dom11.html`.
2. Voeg een lijst toe met minstens 10 items (bijv. landen, games, fruitsoorten).
3. Voeg een zoekveld toe boven de lijst.
4. Laat de lijst automatisch filteren terwijl je typt.
5. Bonus: maak de zoekopdracht hoofdletterongevoelig en toon “Geen resultaten gevonden” als niets matcht.

## Reflectie

- Wat gebeurt er bij het `input`-event?
- Hoe kun je ervoor zorgen dat je filter hoofdletterongevoelig is?
- Wat zou je nog kunnen verbeteren aan deze zoekfunctie?

## Inleveren

- Lever je bestand `dom11.html` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

---

Revision #4

Created 6 June 2025 14:38:30 by Max

Updated 6 June 2025 20:55:19 by Max