JS 3 - (DOM2)

1 Formulieren en invoer met JavaScript

datasource

? Leerdoelen

- Je weet hoe je gegevens uit een formulier leest met JavaScript.
- Je kunt reageren op een `submit`-event.
- Je weet wat `preventDefault()` doet en waarom je het gebruikt.

? Uitleg

Formulieren worden normaal automatisch verstuurd, waarbij de pagina automatisch **ververst**. Dat is standaard zo in HTML, maar soms wil je dat niet. Dan kan je in JavaScript het formulier ook “afhandelen” zonder dat de pagina opnieuw **ververst**.

Dat doe je door te luisteren naar het `submit`-event en vervolgens `preventDefault()` te gebruiken om het standaardgedrag tegen te houden.

Wat is het submit-event?

Het `submit`-event wordt getriggerd als je een formulier probeert te versturen, bijvoorbeeld door op een knop te klikken of op Enter te drukken in een veld. Je kunt dan JavaScript laten reageren op dat moment.

Wat doet preventDefault()?

Deze functie voorkomt dat het formulier echt verzonden wordt en de pagina herlaadt. Daardoor kun je de gegevens gebruiken in JavaScript zonder dat alles verdwijnt.

Voorbeeldformulier:

```
<form id="mijnForm">
  <input type="text" id="naam" placeholder="Typ je naam">
```

```
  <button type="submit">Verstuur</button>
</form>

<div id="resultaat"></div>

<script>
  document.getElementById("mijnForm").addEventListener("submit", function(e) {
    e.preventDefault(); // voorkomt verversen
    const naam = document.getElementById("naam").value;
    document.getElementById("resultaat").textContent = "Hallo " + naam + "!";
  });
</script>
```

?? Opdracht – Formulier verwerken

1. Maak een bestand `dom8.html`.
2. Voeg een formulier toe met een tekstveld voor een bericht en een verstuurknop.
3. Laat het formulier bij klikken niet verversen door `preventDefault()` te gebruiken.
4. Laat het ingevoerde bericht onder het formulier verschijnen in een <p>-element.
5. **Bonus:** Voeg meerdere berichten toe onder elkaar (zoals een eenvoudige chatgeschiedenis).
6. **Extra bonus:** Voeg een tweede input toe voor een gebruikersnaam en laat “Naam zegt: Bericht” zien.

? Reflectie

- Wat doet `preventDefault()` en waarom gebruik je het?
- Wat is het verschil tussen een `click`-event en een `submit`-event?
- Hoe lees je de waarde van een inputveld?

? Inleveren

1. Lever je bestand `dom8.html` in.
2. Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

2 Gegevens bewaren met localStorage

? Leerdoelen

- 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, dan moet je van een lijst een string maken dat kan met `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"]
```

- **`JSON.stringify()`** gebruik je om een array om te zetten in een (JSON) string: **array -> JSON**
- **`JSON.parse()`** gebruik je om een (JSON) string weer om te zetten in een array: **JSON -> array**

Waarom moet je JSON gebruiken bij lijsten?

(denk hier eerst zelf over na voordat je hert antwoord open klikt)

Waarom moet je JSON gebruiken bij lijsten?

In localStorage kun je alleen tekst (strings) opslaan. Als je probeert een lijst of object direct op te slaan, krijg je [object Object] of een fout.

Met JSON.stringify() verander je een array in een string die opgeslagen kan worden.

Als je de lijst later weer wilt gebruiken, gebruik je JSON.parse() om hem terug te veranderen naar een echte array.

?? 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 verwijdert.

? Reflectie

- Wat is het voordeel van `localStorage`?
- Waarom moet je JSON gebruiken bij het opslaan van lijsten?
- Wat gebeurt er als je `JSON.stringify()` vergeet bij het opslaan van een array?
- Wat zou je nog meer kunnen opslaan in een webapp?

? Inleveren

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

3 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 **API** is een soort digitale service waarmee programma's informatie kunnen opvragen of sturen. Denk aan een digitale menukaart: je vraagt iets op, en krijgt data terug.

Type in je browser maar eens deze url in: https://jsonplaceholder.typicode.com/users

Dit kan ook met code, 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);
  });
});
```

Let op: `fetch()` werkt asynchroon. Dat betekent dat de code niet wacht tot de data binnen is. Je gebruikt daarom `.then()` om verder te gaan zodra de data is geladen.

?? 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 <li>.
5. Maak de code zo kort en **eenvoudig** mogelijk, de output kan er dus zo uitzien:

   image.png
   én de HTML code is niet meer dan:

   ```
   <h1>Externe Gebruikerslijst</h1>
   <ul id="users-list"></ul>
   ```

   Natuurlijk moet je de list met id=users-lis met JS code vullen door gegevens uit de api te halen.

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?
- Wat gebeurt er als de API niet beschikbaar is of een fout geeft?

? Inleveren

1. Lever je bestand `dom10.html` in.
2. Beantwoord de reflectievragen in een .txt of .pdf bestand.
3. Lever een screenshot aan waarop de opgehaalde gebruikers zichtbaar zijn in je browser.

4 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>Vogelbekdier</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();
      if (inhoud.includes(tekst)) {
        item.style.display = "list-item";
      } else {
        item.style.display = "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

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

5 Formulier validatie en foutmeldingen tonen

? Leerdoelen

- Je weet hoe je controleert of invoervelden correct zijn ingevuld.
- Je kunt foutmeldingen tonen met JavaScript.
- Je gebruikt `if`-statements om beslissingen te maken.

? Uitleg

In een echt formulier wil je natuurlijk dat gebruikers wél iets invullen – en liefst ook correct. Met JavaScript kun je dit controleren voordat het formulier wordt verwerkt. Dit heet **validatie**.

Voorbeeld: naam en e-mailadres verplicht

```
<form id="mijnForm">
  <input type="text" id="naam" placeholder="Naam"><br>
  <input type="email" id="email" placeholder="E-mail"><br>
  <button type="submit">Verstuur</button>
</form>

<div id="foutmelding" style="color:red;"></div>

<script>
  document.getElementById("mijnForm").addEventListener("submit", function(e) {
    e.preventDefault();
    const naam = document.getElementById("naam").value.trim();
    const email = document.getElementById("email").value.trim();
    const foutvak = document.getElementById("foutmelding");

    if (naam === "" || email === "") {
      foutvak.textContent = "Vul zowel je naam als e-mailadres in.";
    } else {
      foutvak.textContent = "";
      alert("Formulier verstuurd!");
    }
  });
</script>
```

Wat zie je hier gebeuren?

- Het formulier wordt onderschept met `preventDefault()`.
- De waarden worden gecontroleerd op leegte.
- Als iets ontbreekt, verschijnt een foutmelding in het rood.

?? Opdracht – Formulier met foutcontrole

1. Maak een bestand `dom13.html`.
2. Voeg een formulier toe met invoervelden: naam, e-mailadres en een kort bericht.
3. Voeg een knop toe met “Verstuur”.
4. Controleer met JavaScript of alle velden zijn ingevuld.
5. Laat een foutmelding zien als iets ontbreekt (kleur = rood).
6. Laat bij succes een melding zien: “Bedankt voor je bericht!”
7. **Extra:** Controleer of het e-mailadres een @ bevat.
8. **Bonus:** Toon fouten onder elk invoerveld apart.

? Reflectie

- Waarom is inputvalidatie belangrijk?
- Waarom zou je inputvalidatie op de front-end (door de browser) willen maken terwijl je het ook op de back-end (met PHP) kan implementeren?
- Wat gebeurt er als je `preventDefault()` vergeet?
- Hoe weet je of een e-mailadres geldig is?
- Wat zou je nog meer kunnen controleren behalve lege velden?

? Inleveren

1. Lever je bestand `dom13.html` in via Teams of Canvas.
2. Beantwoord de reflectievragen in een .txt of .pdf bestand en lever deze ook in.
3. Stuur een screenshot mee waarop een foutmelding zichtbaar is.

6 Eindopdracht - Boodschappenlijst

? Leerdoelen

- Je kunt met JavaScript items toevoegen aan een lijst op de pagina.
- Je kunt elk item ook verwijderen met een verwijderknop.
- Je zorgt ervoor dat de lijst bewaard blijft via `localStorage`.

? Uitleg

Stel je voor: je staat in de keuken en je denkt: “O ja, ik moet nog melk kopen.” Je opent snel je browser en typt het in een eenvoudige boodschappenlijst. Later op je laptop kun je hetzelfde lijstje nog bekijken. Dat werkt dankzij `localStorage`.

In deze opdracht bouw je een boodschappenlijst die je zelf kunt beheren: toevoegen en verwijderen, en alles blijft bewaard in `localStorage`.

?? Opdracht – Boodschappenlijst met bewaren en verwijderen

1. Maak een bestand `dom12.html`.
2. Voeg een invoerveld en een knop toe om een boodschap toe te voegen.
3. Laat elke boodschap in een lijst onder het formulier verschijnen.
4. Voeg bij elk item een □□-knop toe waarmee het verwijderd kan worden.
5. Zorg ervoor dat de lijst bij het laden van de pagina terugkomt uit `localStorage`.
6. Als je iets verwijdert, moet het ook uit `localStorage` verdwijnen.
7. Vraag om bevestiging bij verwijderen ("Weet je zeker...?").
8. Laat het aantal boodschappen zien bovenaan de lijst.

□□ Real-life toepassing Deze opdracht lijkt simpel, maar vormt de basis voor veel echte apps: todo-lijsten, favorieten, bestellijsten, of bijvoorbeeld een leeslijst. Grote websites gebruiken exact dit soort technieken om gegevens tijdelijk of permanent op te slaan in je browser! □□ maak een **responsive design** dat ook op een telefoonscherm) goed werkt. Je kun hier dan heel eenvoudig een mobiele app van maken. □□wil je de data delen tussen je mobiel en je laptop dan moet je die in een database opslaan en via een API ophalen en updated. Nadat je het volgende blok (databases) hebtNaam zegt: gedaan kan je deze aanpasasing in deze mobile app maken.	image.png

? Reflectie

- Wat gebeurt er als je `localStorage` vergeet bij het verwijderen?
- Hoe weet JavaScript welk item je wilt verwijderen?
- Wat zou je nog meer kunnen doen met zo'n lijst?

? Inleveren

1. Lever je bestand `dom12.html` in..
2. Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.
3. Stuur een screenshot mee waarop te zien is dat je lijst werkt en bij het herladen nog bestaat.

7 Begripsvragen – JavaScript en DOM

Maak deze vragen voor jezelf om te controleren of je alles goed hebt begrepen.

1. Wat doet `preventDefault()` bij een formulier?

Het voorkomt dat het formulier automatisch wordt verstuurd en de pagina herlaadt. Hierdoor kun je met JavaScript eerst controleren of de invoer klopt of iets anders doen met de data.

2. Wat is het verschil tussen een `submit`-event en een `click`-event?

`submit` wordt geactiveerd wanneer een formulier wordt verstuurd (ook bij Enter). `click` is alleen voor het klikken op een specifieke knop. `submit` werkt dus voor het hele formulier.

3. Waarom gebruik je `JSON.stringify()` bij localStorage?

Omdat `localStorage` alleen tekst (strings) kan opslaan. Als je een array of object opslaat zonder stringify, krijg je iets als `[object Object]` of een fout.

4. Wat doet `JSON.parse()`?

Het zet een opgeslagen JSON-string weer terug om in een JavaScript-array of object. Je gebruikt het bij het laden van data uit `localStorage`.

5. Wat doet de `fetch()` functie in JavaScript?

Met `fetch()` haal je gegevens op van een externe server (API). De data komt meestal als JSON terug, die je dan met `.json()` omzet naar bruikbare data.

6. Wat gebeurt er als een API niet beschikbaar is bij `fetch()`?

Dan krijg je een fout of lege gegevens terug. Je kunt dit opvangen met `.catch()` of `try/catch` om de fout netjes af te handelen, bijvoorbeeld met een foutmelding in je UI.

7. Hoe filter je een lijst op basis van invoer?

Je leest de invoer van een `input`-veld en vergelijkt dit met de tekst van elk item in de lijst. Als de tekst matcht, toon je het item, anders verberg je het met `style.display = "none"`.

8. Hoe verwijder je een item uit een lijst én uit `localStorage`?

Je verwijdert het item uit de array in je JavaScript-code (bijv. met `splice()`) en daarna schrijf je de nieuwe lijst terug naar `localStorage` met `setItem()`.

9. Waarom is inputvalidatie belangrijk bij een formulier?

Om te voorkomen dat gebruikers lege of onjuiste gegevens versturen. Dat zorgt voor betere data, minder fouten en een betere gebruikerservaring.

10. Hoe kun je controleren of een e-mailadres geldig is?

Een simpele manier is te controleren of het een `@` bevat. Voor geavanceerdere controles kun je reguliere expressies gebruiken, maar dat is voor later.

? Reflectie

- Welke vraag mis jij in deze lijst (en is wel onderdeel van deze module)?

? Inleveren

- Beantwoord de reflectievraag in een .txt bestand.

  Dus je levert in: de vraag én het antwoord.,

8 Mini-project – Community Board

Deze opgave is een **eindopdracht**. Je mag AI gebruiken, maar je moet de code begrijpen en **mondeling** kunnen toelichten.

Als je klaar bent maak je een afspraak met een docent en die neemt kort de code met je door. De docent kan vragen wat bepaalde code doet of kan je bijvoorbeeld vragen om een klleine aanpassing te maken.

? Leerdoelen

- Je combineert alle technieken die je in deze module hebt geleerd.
- Je gebruikt formulieren, validatie, localStorage en fetch() in één project.
- Je maakt een interactieve webapplicatie met JavaScript en de DOM.

? Uitleg

In deze les bouw je een **mini Community Board** – een eenvoudige webapp waar gebruikers een naam en bericht kunnen plaatsen. De berichten blijven bewaard met `localStorage` en bij het laden wordt een willekeurige gebruiker opgehaald via een API.

Onderdelen die je gebruikt:

- `addEventListener()` – om te reageren op formulier-acties
- `preventDefault()` – om te voorkomen dat de pagina ververst
- `localStorage` – om berichten op te slaan
- `fetch()` – om data van een externe API te halen
- `innerHTML` en `createElement()` – om elementen dynamisch te maken

?? Opdracht – Bouw je eigen mini community board

1. Maak een nieuw bestand `dom14.html`.
2. Voeg een formulier toe met:
   - een invoerveld voor **naam**
   - een invoerveld voor **bericht**
   - een knop met de tekst **Verstuur**
3. Gebruik `preventDefault()` zodat de pagina niet herlaadt bij verzenden.
4. Controleer of beide velden zijn ingevuld. Toon een foutmelding onder het juiste veld.
5. Laat elk bericht onder het formulier verschijnen in een lijst.
6. Sla de berichten op in `localStorage`, zodat ze blijven staan bij herladen.
7. Voeg een knop toe om alle berichten te wissen (`localStorage.clear()`).
8. Gebruik `fetch("https://jsonplaceholder.typicode.com/users")` om bij het laden van de pagina een willekeurige gebruiker te tonen met zijn naam en e-mailadres bovenaan.

9. Zorg voor een overzichtelijke, responsive opmaak (mag met basis-CSS).
10. Voeg iets 'persoonlijks' toe: dit mag in het ontwerp zijn, maar het mnag ook iets functioneels zijn.

Bonus

- Voeg een zoekveld toe waarmee je berichten kunt filteren.
- Voeg een □□-knop toe bij elk bericht om het te verwijderen.

? Reflectie

- Welke technieken uit eerdere lessen heb je gebruikt?
- Wat werkte direct goed, en wat moest je debuggen?
- Hoe heb je gecontroleerd of `localStorage` goed functioneert?
- Welke persoonlijke 'touch' heb jij gegeven?

? Inleveren

- Lever je bestand `dom14.html` in via Teams of Canvas.
- Lever ook een screenshot in waarop minimaal drie berichten en één API-gebruiker zichtbaar zijn.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever dit mee in.

Nadat je alles heb ingeleverd, wordt je uitgenodigd voor een gesprek.

Tijdens het gesprek laat je zien dat je begrijpt wat je code doet en kun je kleine aanpassingen uitvoeren als de docent daarom vraagt.

Revision #22
Created 2025-06-06 14:38:30 UTC by Max
Updated 2026-01-15 09:55:29 UTC by Max