

# Blok 5 - Database en JS (DOM)

- [Prompt Engineering 2](#)
- [JS - \(DOM1\)](#)
- [JS - \(DOM2\)](#)
- [Java Script Challenge](#)

# Prompt Engineering 2

## 1, *Introductie*

We hebben in prompt engineering 1 geleerd waaraan een goede prompt moest voldoen.

Dit zijn de basis kenmerken van een goede prompt. De eerste drie kenmerken moet je prompt **altijd** aan voldoen!

1. **Context** - een goede prompt heeft voldoende context.
2. **Details/Specifiek** - een goede prompt heeft voldoende details en is zo specifiek mogelijk.
3. **Duidelijkheid** - een goede prompt is duidelijk.
4. **Doelgericht** - een goede prompt is doelgericht.
5. **Vorm** - in een goede prompt kan je de output in een bepaalde vorm vragen.
6. **Toon** - door in de prompt de toon op te nemen, bepaal je de vorm van het antwoord.

In deze module gaan we 6 **advanced prompt-technieken** leren. Deze technieken heb je niet altijd nodig maar het is handig om deze technieken te kennen.

Bovendien zijn de meeste technieken ook toepasbaar in als '**problem solving**' technieken.

1. **Isolate the problem**  
Focus alleen op het onderdeel dat opgelost moet worden.  
Laat overbodige context of code weg, zodat de AI zich op het juiste richt.
2. **Provide lists in bullet points**  
Structuur helpt de AI om overzichtelijke en duidelijke antwoorden te geven.
3. **Provide the order if you ask for multiple tasks**  
Geef een logische volgorde bij samengestelde opdrachten.
4. **Geef voorbeelden (few-shot prompting)**  
Laat zien wat je bedoelt met een input/output-voorbeeld.

## 5. **Stel voorwaarden of beperkingen**

Geef grenzen aan zoals "gebruik max. 100 woorden" of "geen disclaimers".

## 6. **Werk in stappen (chain-of-thought prompting)**

Vraag de AI om stap voor stap te redeneren of eerst een plan te maken.

# 1. *Isolate the problem*

## Uitleg

Richt je prompt op het exacte probleem. In plaats van een hele codepagina te geven, geef alleen het stuk code of de situatie waar het om draait. Hoe minder ruis, hoe beter de AI kan helpen.

Je ziet dus dat je voor een goede prompt de code goed moet kunnen lezen.

## Opdracht

Je hebt een PHP-pagina met een formulier dat soms geen gegevens doorstuurt.

☐ Maak een prompt voor ChatGPT waarin **alleen het relevante deel van de code opneemt** en waarin je de AI vraagt om te helpen bij het vinden van de fout.

Houd nog steeds rekening met de **context, details en duidelijkheid**.

In dit voorbeeld zou je makkelijk de hele code kunnen verturen, maar in de echte wereld heb je veel meer code en daarom is dit een goede oefening in het isoleren van code die relevant is.

Tip: je kan ook andersom redeneren en alle code waarvan je zeker weet die goed is of niets met het formulier te maken heeft weghaald.

```
<?php
$bericht = "";
$success = false;

if ($_SERVER["REQUEST_METHOD"] === "POST") {
    if (!empty($_POST["naam"])) {
        $naam = $_POST["naam"];
```

```
    $bericht = "Hallo, " . htmlspecialchars($naam) . "!";  
    $success = true;  
} else {  
    $bericht = "Je moet je naam invullen."  
}  
}  
?>
```

```
<!DOCTYPE html>  
<html lang="nl">  
<head>  
    <meta charset="UTF-8">  
    <title>Contactpagina</title>  
    <style>  
        body { font-family: Arial; background-color: #f0f0f0; margin: 2rem; }  
        .container { background: white; padding: 2rem; border-radius: 8px; max-width: 600px; margin: auto; }  
        .error { color: red; }  
        .success { color: green; }  
        footer { margin-top: 4rem; font-size: 0.8em; text-align: center; color: #666; }  
    </style>  
    <script>  
        function checkForm() {  
            const naam = document.getElementById("naam").value;  
            if (naam.trim() === "") {  
                alert("Naam is verplicht!");  
                return false;  
            }  
            return true;  
        }  
    </script>  
</head>  
<body>  
  
<div class="container">  
    <h1>Neem contact met ons op</h1>  
  
    <form method="get" action="" onsubmit="return checkForm();">  
        <label for="naam">Naam:</label><br>  
        <input type="text" id="naam" name="naam" placeholder="Vul je naam in"><br><br>
```

```
<label for="email">E-mail:</label><br>
<input type="email" id="email" name="email" placeholder="voorbeeld@domein.nl"><br><br>
<label for="vraag">Je vraag:</label><br>
<textarea id="vraag" name="vraag"></textarea><br><br>
<button type="submit">Verzenden</button>
</form>

<?php if ($bericht): ?>
    <p class="<?= $success ? 'success' : 'error' ?>"><?= $bericht ?></p>
<?php endif; ?>
</div>

<footer>
    &copy; 2025 Webdev Company | <a href="#">Privacybeleid</a>
</footer>

</body>
</html>
```

# Inleveren

De prompt die het probleem uiteindelijk vond en waarin **alleen** de relevante code staat.

De prompt voldoet tevens minimaal aan de basis kenmerken van een prompt: **context, details en duidelijkheid**.

## 2, Provide lists in bullet points

## Uitleg

Als je meerdere dingen van een AI vraagt, is het belangrijk om structuur aan te brengen in je prompt. Door **bullet points of genummerde lijsten** te gebruiken:

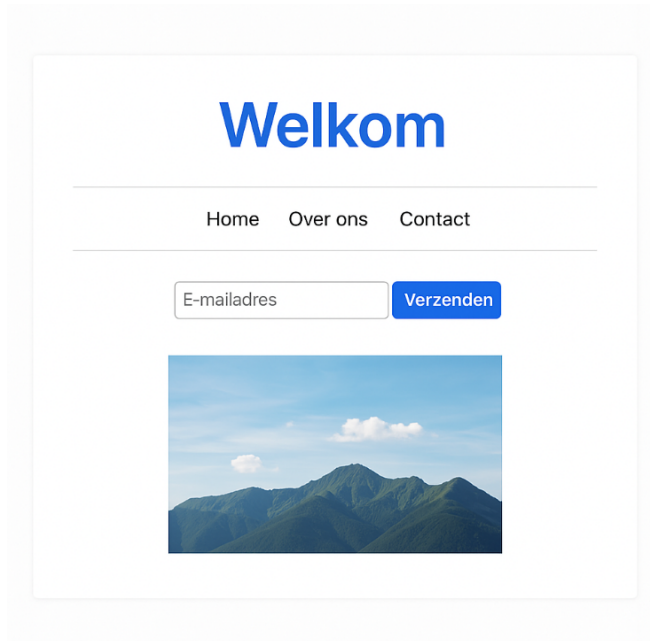
- Maak je je prompt overzichtelijk.
- Zorg je ervoor dat de AI geen onderdelen vergeet.
- Help je jezelf om duidelijk te verwoorden wat je wilt.

Bullet points zijn dus niet alleen netjes — ze zijn slim.

# Opdracht

Hieronder zie je een afbeelding van een eenvoudige webpagina met meerdere onderdelen.

Het plaatje kan je hier downloaden: [Berglandschap.png](#).



1. **Bestudeer de afbeelding goed.** Wat zie je allemaal op de pagina?
2. **Bedenk wat je aan de AI zou vragen om precies deze pagina te laten maken.**
3. **Noteer minstens 4 onderdelen van de pagina als bullet points** in je prompt.

# Inleveren

1. Prompt
2. Resultaat in code
3. Schermafdruk van resultaat

*3. Provide the order if you ask for multiple tasks*

# Uitleg

Als je aan de AI vraagt om **meerdere dingen tegelijk te doen**, dan is het belangrijk dat je duidelijk maakt in **welke volgorde** dat moet gebeuren.

AI volgt jouw instructies letterlijk — dus als de volgorde onduidelijk is, krijg je soms een rommelig of onvolledig resultaat.

Door een logische **nummering** of expliciete volgorde te geven, help je de AI om stapsgewijs te werken.

## Voorbeeld (vaag):

*Maak een invoerpagina met HTML en verwerk de gegevens met PHP.*

## Voorbeeld (duidelijker):

1. Maak eerst een HTML-pagina met een formulier waarin wordt gevraagd naar de tijd en de klantnaam
2. Voeg daarna de PHP-code toe die de ingevulde gegevens toont.

# Opdracht

Je wilt de AI vragen om je te helpen met het maken van een eenvoudige contactpagina.

Die moet bestaan uit:

- Een HTML-formulier waarin je je naam en e-mailadres kunt invullen
  - Een PHP-script dat de gegevens verwerkt en netjes op het scherm toont
1. Schrijf een prompt waarin je deze twee onderdelen vraagt in de juiste **volgorde**.
  2. Gebruik een **genummerde lijst** of maak duidelijk met woorden wat “eerst” en “daarna” moet gebeuren.
  3. Houd rekening met de **context, details en duidelijkheid**.

# Inleveren

1. Je volledige prompt

2. De gegenereerde HTML + PHP code
3. Een screenshot van het resultaat in de browser

## 4. Geef voorbeelden (few-shot prompting)

### Uitleg

Als je wilt dat de AI **op een specifieke manier code genereert of uitlegt**, dan helpt het enorm als je eerst **een paar voorbeelden** geeft. Dit heet **few-shot prompting**.

Je laat zien wat jij bedoelt — de AI herkent het patroon en volgt het. Dit is heel handig bij het ontwerpen van knoppen, formulieren of herhalende structuren.

### Voorbeeld prompt

```
<button style="background-color: red;">Verwijderen</button><button  
  style="  
    background-color: #dc3545;  
    color: white;  
    border: none;  
    padding: 10px 16px;  
    font-size: 16px;  
    border-radius: 4px;  
    cursor: pointer;  
    transition: background-color 0.3s ease;  
  "  
  onmouseover="this.style.backgroundColor='#c82333'"  
  onmouseout="this.style.backgroundColor='#dc3545'"  
  onclick="handleDelete()"  
>  
  Verwijderen  
</button>
```

Deze knop ziet er professioneel uit: rood met witte tekst, afgeronde hoeken en een vloeiend hover-effect dat de kleur donkerder maakt wanneer je erover beweegt.

Klikt de gebruiker op de knop, dan wordt de JS code `handleDelete()` aangeroepen.

Maak op dezelfde manier een blauwe knop 'toevoegen'.

# Opdracht

Je wilt een aantal drop down menu's maken. Hier is een voorbeeld.

```
<style>
.form-control {
  width: 250px;
  padding: 8px 12px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 6px;
  background-color: #f9f9f9;
  appearance: none; /* verwijder standaardstijl in sommige browsers */
}

.form-control:focus {
  border-color: #007BFF;
  outline: none;
  background-color: #fff;
  box-shadow: 0 0 5px rgba(0, 123, 255, 0.3);
}
</style>

<select id="taalSelect" name="taal" class="form-control">
  <option value="en">Engels</option>
  <option value="nl" selected>Nederlands</option>
  <option value="fr">Frans</option>
</select>
```

Maak een multi-shot prompt waarbij je dit menu als voorbeeld gebruikt en waarbij je het volgende menu maakt:

Tip: denk goed na over wat je de AI precies als voorbeeld geeft, **alleen** de HTML of HTML **met** CSS? Denk hierbij ook aan *Isolate the problem*.

## Inleveren

1. Je volledige prompt inclusief voorbeeld(en)
2. De gegenereerde knop + uitleg van de AI
3. Een screenshot van de knop in de browser

## 5. Stel voorwaarden of beperkingen

### Uitleg

Soms wil je meer controle over **hoe** de AI iets oplevert. Je kunt dan in je prompt **voorwaarden of beperkingen** opnemen.

Bijvoorbeeld:

- Beperk het aantal woorden of regels code
- Vermijd bepaalde technieken (zoals frameworks)
- Vraag om iets op een specifieke manier te doen (bijv. alleen in-line CSS)

Door deze randvoorwaarden op te nemen, stuur je de AI veel gericht aan.

## Opdracht

Je wilt een eenvoudige HTML-formulierpagina laten genereren, maar je stelt een aantal duidelijke voorwaarden.

1. **Bedenk minstens 3 voorwaarden** waaraan de code moet voldoen. Bijvoorbeeld:
  - Geen externe CSS-bestanden of frameworks zoals Bootstrap
  - Maximaal 30 regels HTML
  - Formulier moet naam en e-mailadres bevatten, met eenvoudige styling
2. **Schrijf een prompt** waarin je deze voorwaarden duidelijk formuleert.
  - Leg de context uit: *“Ik wil een eenvoudig formulier...”*
  - Som je voorwaarden op in een bulletlijst of genummerde lijst
  - Sluit duidelijk af met je verzoek: *“Geef alleen de HTML-code”*
3. Zorg dat je prompt voldoet aan de **context, details en duidelijkheid**.

## Inleveren

1. Je prompt (inclusief de voorwaarden)
2. De gegenereerde HTML-code
3. Een screenshot van het formulier

## 6. Chain-of-thought prompting 1

### Uitleg

Bij opdrachten met meerdere onderdelen is het slim om de AI **eerst te laten nadenken en plannen**. Dat noem je *chain-of-thought prompting*.

Je vraagt de AI dus om niet meteen te bouwen, maar eerst te analyseren:

- Wat moet er gebeuren, welke stappen moet ik volgen?
- In welke volgorde?
- Wat zijn de componenten en technieken die ik moet gebruiken?

Daarna kan de AI **gestructureerd** de code aanleveren. Zo voorkom je onvolledige of chaotische antwoorden.

# Voorbeeld

## Prompt

*Ik wil een formulier dat een naam opslaat.  
Leg eerst uit welke stappen ik daarvoor nodig heb.  
Geef dan de HTML-code.  
Geef daarna de PHP-code.  
Leg elke stap kort uit.*

## Verwachte AI-reactie

1. Je hebt een HTML-formulier nodig om de naam in te voeren.
2. Je hebt een PHP-script nodig dat de naam opvangt en verwerkt.
3. Je moet het formulier met method="post" versturen.
4. Daarna geef ik de HTML en PHP code apart, met uitleg.

Als je een stap niet snapt, of het niet eens bent, vraag dan verheldering. Vraag uitleg, bijvoorbeeld:

## Prompt

*Wat gebeurt er in stap 3 en waarom moet ik ene post gebruiken, zijn er alternatieven?*

Op deze manier leer je een probleem in stappen opdelen en daarna de stappen één voor één op te lossen. Nu lijkt dat wellicht overbodig, maar bij grotere projecten is deze methode noodzakelijk.

# Opdracht

Je wil een AI vragen om een simpele "Contact opnemen"-pagina te maken met de volgende onderdelen:

- Een **HTML-formulier** met velden voor naam, e-mail en bericht
- Een **mooie layout met CSS** (geen framework)
- Een **PHP-bestand** dat het formulier verwerkt en de data netjes toont
- **De gegevens mogen niet verstuurd worden zonder geldige invoer**

**Maar:** je wil niet meteen de code, je wil dat de AI eerst in stappen uitlegt *hoe je dit moet aanpakken*.

1. **Schrijf een prompt** waarin je aan de AI vraagt om:
  - Eerst te analyseren wat deze opdracht inhoudt
  - Daarna een overzicht in stappen te geven van wat er allemaal moet gebeuren (HTML-structuur, input validatie, PHP-verwerking, opmaak)
  - Pas daarna per stap de juiste code te geven, met uitleg
2. Gebruik termen als:
  - “Geef eerst een analyse van de opdracht”
  - “Schrijf daarna de stappen uit in logische volgorde”
  - “Geef dan pas de bijbehorende code, met uitleg per stap”
3. Voeg in je prompt beperkingen of voorkeuren toe, zoals:
  - “Gebruik geen externe CSS-frameworks”
  - “Gebruik eenvoudige PHP, geen database”
  - “Geef geen code voordat alle stappen duidelijk zijn”
4. Zorg dat je prompt voldoet aan de kenmerken: **context, details, duidelijkheid, volgorde**

## Inleveren

1. Je volledige prompt(en)
2. Het AI-antwoord (mag opgeknipt zijn in stappen)
3. De uiteindelijke werkende HTML- en PHP-code
4. Screenshot van de werkende pagina

## 7. Chain-of-thought prompting 2

In de vorige opdracht heb je een formulier laten maken.

Wat voor method heb je gebruikt in het formulier?

Vraag AI om uit te leggen welke methoden er zijn en wat de voor en nadelen zijn van bedien methoden?

# Inleveren

1. Je volledige prompt.
2. In een txt bestand: Jouw eingevoerde aanbeveling: wat zou je het beste hebben kunnen gebruiken bij de vorige opdracht. Leg uit waarom!

# JS - (DOM1)

## *1 Elementen ophalen en aanpassen*

### Doelstellingen

- Je weet wat de DOM is.
- Je kunt HTML-elementen selecteren met JavaScript.
- Je kunt de inhoud en stijl van elementen aanpassen via JavaScript.

### Uitleg

De DOM (Document Object Model) is de structuur van je HTML-document zoals de browser die begrijpt. Met JavaScript kun je deze structuur lezen en aanpassen.

Voorbeeld – een paragraaf veranderen:

```
<p id="mijnParagraaf">Oude tekst</p>
<script>
  const p = document.getElementById("mijnParagraaf");
  p.textContent = "Nieuwe tekst!";
  p.style.color = "blue";
</script>
```

### Opdracht – Tekst aanpassen

1. Maak een nieuw bestand aan met de naam `dom1.html`.
2. Maak hierin een kopje, een paragraaf met een id, en een knop.

3. Als je op de knop klikt, moet de tekst in de paragraaf veranderen naar iets anders (bijv. "Hallo wereld!").

Gebruik bijvoorbeeld:

```
document.getElementById("knop").addEventListener("click", function() {  
    document.getElementById("mijnParagraaf").textContent = "Hallo wereld!";  
});
```

## ☐☐ Reflectie

- Wat is de DOM in eigen woorden?
- Wat doet `getElementById` precies?
- Waarom zou je de stijl van een element met JavaScript aanpassen en niet met CSS?

## ☐☐ Inleveren

- Lever je bestand `dom1.html` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever deze mee in.

# 2 Meerdere elementen aanpakken

## ☐☐ Doelen

- Je kunt meerdere elementen selecteren met `querySelectorAll`.
- Je kunt met `forEach` een actie uitvoeren op elk element.
- Je kunt een class toevoegen of verwijderen met `classList`.

## ☐☐ Uitleg

Als je meerdere elementen tegelijk wilt aanpakken (zoals alle `<p>`-elementen of alle knoppen), gebruik je `querySelectorAll`. Dit geeft je een lijst (een zogenaamde "NodeList") van alle elementen die matchen.

Met `forEach` kun je vervolgens over deze lijst heen lopen en elk element iets laten doen:

```
<p>Item 1</p>
<p>Item 2</p>
<p>Item 3</p>

<script>
  const alleP = document.querySelectorAll("p");
  alleP.forEach(function(p) {
    p.style.color = "green";
  });
</script>
```

Je kunt ook classes toevoegen of weghalen met `classList`:

```
p.classList.add("actief");
p.classList.remove("verborgen");
p.classList.toggle("geselecteerd");
```

## ☐☐ Opdracht – items markeren

1. Maak een bestand `dom2.html`.
2. Maak een lijst van minimaal 5 `<p>`-elementen met een class `item`.
3. Maak een knop met de tekst “Markeer alles”.
4. Wanneer je op de knop klikt, moeten alle `<p class="item">` elementen de class `actief` krijgen.

Gebruik bijvoorbeeld:

```
document.getElementById("knop").addEventListener("click", function() {
  document.querySelectorAll(".item").forEach(function(el) {
    el.classList.add("actief");
  });
});
```

Stijl de class `actief` in je `<style>` met bijvoorbeeld een achtergrondkleur of rand.

## ☐☐ Reflectie

- Wat doet `querySelectorAll(".item")` precies?
- Wat is het verschil tussen `getElementById` en `querySelectorAll`?
- Waarom gebruik je `forEach` bij een `NodeList`?

## ☐ Inleveren

- Lever het bestand `dom2.html` in via Teams of Canvas.
- Beantwoord de reflectievragen in een `.txt` of `.pdf` bestand en lever die ook in.

# 3 Interactie met knoppen en events

## ☐ Doelstellingen

- Je begrijpt wat een event is in JavaScript.
- Je kunt reageren op een klik of muisactie met `addEventListener`.
- Je kunt een actie koppelen aan meerdere elementen.

## ☐ Uitleg

Een event is iets wat gebeurt in de browser: een klik, het bewegen van de muis, een toets indrukken...

Met `addEventListener` kun je zeggen: "Als dit gebeurt, doe dan dat."

```
const knop = document.getElementById("klikMij");
knop.addEventListener("click", function() {
  alert("Je klikte op de knop!");
});
```

Ook andere events zijn mogelijk, zoals `mouseover`, `mouseout`, `keydown` enzovoort.

Je kunt ook meerdere elementen selecteren en daar een event aan koppelen:

```
document.querySelectorAll(".kleurvak").forEach(function(el) {  
  el.addEventListener("mouseover", function() {  
    el.style.backgroundColor = "yellow";  
  });  
});
```

## Opdracht – Events in actie

1. Maak een nieuw HTML-bestand `dom3.html`.
2. Voeg 5 divjes toe met de class `kleurvak`, elk met een vaste afmeting en een andere begin-keur.
3. Als je met de muis over een vakje gaat, verandert de achtergrondkeur in geel.
4. Als je erop klikt, moet de tekst in het vakje veranderen naar “Geklikt!”.

Gebruik zowel `mouseover` als `click` events.

Voorbeeld CSS:

```
.kleurvak {  
  width: 100px;  
  height: 100px;  
  display: inline-block;  
  margin: 10px;  
  text-align: center;  
  line-height: 100px;  
  background-color: lightblue;  
}
```

## Reflectie

- Wat is een event in je eigen woorden?
- Wat doet `addEventListener` precies?
- Wat is het verschil tussen `mouseover` en `click`?

## Inleveren

- Lever je bestand `dom3.html` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

## 4 Elementen toevoegen met JavaScript

### Leerdoelen

- Je kunt een nieuw HTML-element aanmaken met `createElement`.
- Je kunt dat element toevoegen aan de DOM met `appendChild`.
- Je kunt invoer van de gebruiker gebruiken om dynamisch iets te maken.

### Uitleg

Je kunt nieuwe HTML-elementen maken en ze toevoegen aan je pagina met JavaScript. Dit is handig als je bijvoorbeeld automatisch lijstjes wilt uitbreiden of reacties wilt tonen.

```
const nieuwElement = document.createElement("p");
nieuwElement.textContent = "Hallo, ik ben nieuw!";
document.body.appendChild(nieuwElement);
```

Je kunt ook iets maken op basis van wat de gebruiker invoert:

```
<input type="text" id="tekstvak">
<button id="voegToe">Voeg toe</button>
<div id="resultaat"></div>

<script>
document.getElementById("voegToe").addEventListener("click", function() {
  const invoer = document.getElementById("tekstvak").value;
  const nieuwP = document.createElement("p");
  nieuwP.textContent = invoer;
  document.getElementById("resultaat").appendChild(nieuwP);
});
```

```
});  
</script>
```

## Opdracht – Invoer toevoegen

1. Maak een bestand `dom4.html`.
2. Voeg een invoerveld toe waarin de gebruiker een hobby, film of favoriet eten kan typen.
3. Voeg een knop toe met de tekst “Toevoegen”.
4. Telkens wanneer je klikt, moet er een nieuw `<p>`-element met de ingevoerde tekst verschijnen onder een `lijstdiv`.

Bonus: Als je het leuk vindt, laat de invoervelden na het klikken automatisch leeglopen.

## Reflectie

- Wat doet `createElement` precies?
- Wat is het verschil tussen `textContent` en `innerHTML`?
- Waarom moet je `appendChild` gebruiken?

## Inleveren

- Lever je bestand `dom4.html` in via Teams of Canvas.
- Beantwoord de reflectievragen in een `.txt` of `.pdf` bestand en lever die ook in.

## 5 Elementen verwijderen of aanpassen via `event.target`

## Leerdoelen

- Je begrijpt wat `event.target` doet.

- Je kunt een klik koppelen aan een specifiek element dat je wilt aanpassen of verwijderen.
- Je kunt met JavaScript elementen verwijderen uit de DOM.

## □□ Uitleg

Als een event plaatsvindt (zoals een klik), kun je met `event.target` achterhalen welk element precies geklikt is.

Voorbeeld – klikbare lijst waarin een item verdwijnt:

```
<ul id="lijst">
  <li>Appel</li>
  <li>Banaan</li>
  <li>Peer</li>
</ul>

<script>
  document.querySelectorAll("#lijst li").forEach(function(item) {
    item.addEventListener("click", function(event) {
      event.target.remove();
    });
  });
</script>
```

Of met `this` als shorthand:

```
item.addEventListener("click", function() {
  this.remove();
});
```

## □□ Opdracht – Klik en verwijder

1. Maak een bestand `dom5.html`.
2. Voeg een lijst toe (bijv. `<ul>`) met minstens 5 items (bijv. films, dieren of snacks).
3. Schrijf JavaScript die ervoor zorgt dat je een item uit de lijst verwijdert zodra je erop klikt.

4. Bonus: Toon boven de lijst hoeveel items er nog over zijn.

Gebruik `event.target.remove()` of `this.remove()` binnen je event handler.

## ☐☐ Reflectie

- Wat is `event.target` en waar gebruik je het voor?
- Wat is het verschil tussen `event.target` en `this` in een event?
- Waarom zou je een lijst dynamisch willen kunnen aanpassen?

## ☐☐ Inleveren

- Lever je bestand `dom5.html` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

# 6 *Klassennamen wisselen met* *`classList.toggle`*

## ☐☐ Leerdoelen

- Je kunt met JavaScript classes toevoegen of verwijderen.
- Je weet wat `classList.toggle` doet.
- Je kunt styling aanpassen afhankelijk van de class van een element.

## ☐☐ Uitleg

Met `classList` kun je een class toevoegen, verwijderen of omwisselen (“toggelen”). Dit is handig om styling of gedrag van elementen aan te passen wanneer de gebruiker iets doet.

Bijvoorbeeld: klik op een element om het te markeren:

```
<ul id="taken">
  <li>Boodschappen doen</li>
  <li>Afwassen</li>
  <li>Hond uitlaten</li>
</ul>

<style>
  .afgevinkt {
    text-decoration: line-through;
    color: grey;
  }
</style>

<script>
  document.querySelectorAll("#taken li").forEach(function(taak) {
    taak.addEventListener("click", function() {
      this.classList.toggle("afgevinkt");
    });
  });
</script>
```

## ☐ Opdracht – Actieve selectie

1. Maak een bestand `dom6.html`.
2. Maak een lijst (bijv. favoriete games, liedjes, sporters).
3. Als je op een item klikt, moet de class `geselecteerd` worden toegevoegd of verwijderd.
4. Stijl de class `geselecteerd` in CSS met bijvoorbeeld een andere kleur en achtergrond.

Voorbeeld CSS:

```
.geselecteerd {
  background-color: lightgreen;
  font-weight: bold;
}
```

Gebruik `element.classList.toggle("geselecteerd")` bij het klikken.

## ☐☐ Reflectie

- Wat is het voordeel van `toggle` ten opzichte van `add` en `remove`?
- Wat gebeurt er als je meerdere keren op hetzelfde item klikt?
- Waar zou je dit in een echte webapp kunnen gebruiken?

## ☐☐ Inleveren

- Lever het bestand `dom6.html` in via Teams of Canvas.
- Beantwoord de reflectievragen in een `.txt` of `.pdf` bestand en lever die ook in.

# JS - (DOM2)

## 1 Formulieren en invoer met JavaScript

### Leerdoelen

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

### Uitleg

Formulieren worden normaal automatisch verstuurd en de pagina verversen. Maar in JavaScript kun je het formulier ook “afhandelen” zonder te verversen.

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 + "!";
});
```

## Opdracht – Formulier verwerken

1. Maak een bestand `dom7.html`.
2. Voeg een formulier toe met een tekstveld voor een bericht en een verstuurknop.
3. Laat het formulier bij klikken niet verversen.
4. Laat het ingevoerde bericht onder het formulier verschijnen in een `<p>`-element.
5. Bonus: Voeg meerdere berichten toe (zoals een eenvoudige chat).

## 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

- Lever je bestand `dom7.html` in via Teams of Canvas.
- 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? 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.

# 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 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.

## 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();
  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.

# Java Script Challenge

## DOM Challenge – Bouw jouw eigen mini-app

### Leerdoelen

- Je past alle basisvaardigheden toe rondom DOM-manipulatie.
- Je maakt een interactieve webapp met HTML, CSS en JavaScript.
- Je leert werken met gebruikersinvoer, events en eventueel localStorage.
- Je leert een eenvoudige planning te maken voor je project.
- Je leert AI bewust inzetten en daarover verantwoording afleggen.

### Uitleg

In deze les ontwerp je zelf een kleine interactieve DOM-app. Je past de technieken toe uit de vorige lessen. Kies één van de volgende apps, of verzin een eigen variant:

### Mogelijke projecten:

- **Todo-lijst:** gebruiker voert een taak in, kan deze toevoegen, afvinken (class toggle), en verwijderen (event.target.remove()).
- **Quiz:** gebruiker kiest een antwoord en krijgt direct feedback.
- **Poll/stemming:** klik op een optie, zie het aantal stemmen stijgen.
- **Chatbox:** gebruikersberichtjes invoeren die onder elkaar verschijnen.
- **Boekenkast:** boeken toevoegen, afvinken als gelezen, verwijderen, zoeken én opslaan in localStorage.

**Let op:** Bij elk project gebruik je minimaal deze onderdelen:

- Een invoerveld + knop
- Een lijst of reeks elementen die via JavaScript groeit
- Event-handling (bijv. click, submit)
- `classList.toggle()` of `remove()`

Bij het project **Boekenkast** moet je daarnaast ook werken met `localStorage` en een live zoekveld.

## Opdracht – Kies en bouw jouw mini-app

1. Maak een bestand met een duidelijke naam, bijv. `dom-project.html`.
2. Kies welk klein DOM-project je maakt (zie hierboven).
3. Maak een **planning** van je project in een tabel (zie voorbeeld hieronder).
4. Werk stap voor stap: begin met HTML, voeg daarna JavaScript toe.
5. Gebruik technieken uit eerdere lessen zoals `querySelector`, `addEventListener`, en `innerHTML`.
6. Stijl je pagina met CSS zodat deze overzichtelijk is.

### Voorbeeld planning

Onderdeel	Tijd	Status
HTML structuur	20m	
JS: items toevoegen	30m	
JS: verwijderen/afvinken	20m	
Styling	15m	
Reflectie schrijven	15m	

## Reflectie

- Wat vond je het makkelijkst en het moeilijkst?
- Welke technieken heb je toegepast? Noem er minstens drie.
- Waar heb je AI voor gebruikt?
- Wat zou je in de toekomst nog willen verbeteren of leren?

# ☐☐ Inleveren

- Lever je `dom-project.html` in, samen met je CSS-bestand (indien apart).
- Lever je **planning** in als .txt of .pdf.
- Lever je **reflectie** in als .txt of .pdf.
- Lever je **AI-logboek** in: geef aan welke prompts je gebruikte, wat je codeerde met hulp van AI, en wat je zelf schreef of aanpaste.
- Eventueel: voeg screenshots toe van je werkende webapp als je trots bent op het resultaat!