Blok 5 - Database en JS (DOM)

- Prompt Engineering 2
- JS 2 - (DOM1)
- JS 3 - (DOM2)
- Java Script Challenge
- Kennis Check Blok 5

Prompt Engineering 2

0, Introductie

source

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 goed 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 bepalade 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.

??Opdracht

Situatie

Een student wil een AI laten helpen bij het schrijven van een korte uitleg over **variabelen in Python** voor een klas van 14-jarigen. Hieronder staan drie prompts.

Lees de drie prompts en bepaal bij **welke prompt** de **context**, de **details**, of de **duidelijkheid** het sterkst aanwezig is.

Opgave 1

> Leg in eenvoudige taal uit wat een variabele is in Python. Gebruik korte zinnen en vermijd moeilijke woorden.

□□ **Vraag:** Waar blinkt deze prompt vooral in uit?

- Context
- Details
- Duidelijkheid

Opgave 2

> Ik geef les aan 14-jarige leerlingen die net beginnen met programmeren in Python. Ze begrijpen wat print() doet, maar hebben nog nooit met variabelen gewerkt. Leg uit wat een variabele is, geef een eenvoudig voorbeeld en gebruik taal die past bij hun leeftijd.

□□ **Vraag:** Waar blinkt deze prompt vooral in uit?

- Context
- Details

- Duidelijkheid

Opgave 3

> Schrijf een uitleg van 100 woorden over variabelen in Python. Gebruik de termen “waarde”, “naam”, en “toewijzing”. Voeg twee codevoorbeelden toe: één met een getal en één met tekst. Sluit af met een korte quizvraag.

◻◻ **Vraag:** Waar blinkt deze prompt vooral in uit?

- Context
- Details
- Duidelijkheid

? Inleveren

Geef in een txt bestand (of in het tkstveld) weer van elke van de **drie opgaven** welk onderdeel, **context**, **details**, of **duidelijkheid** het sterkst aanwezig is.

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 je **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.

image.png

1. **Bestudeer de web pagina 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 (txt)
2. Resultaat in code (html)
3. Schermafdruk van resultaat (jpg of png)

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 (txt)
2. De gegenereerde HTML (.html)

3. De gegenereerde PHP (.php)
4. 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 het menu uit de code hierboven als voorbeeld gebruikt en waarbij je het volgende menu maakt:

image.png

> 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) en de responce van AI (.txt)
2. De gegenereerde knop (.html)
3. Een screenshot van de knop in de browser (.png, .jpg)

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 gerichter 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 (.txt)
2. De gegenereerde HTML-code (.html)
3. Een screenshot van het formulier (.jpg, .png)

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 gebeuert er in stap 3 en waarom moet ik een 'method=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 antwoord (.txt)
2. De uiteindelijke werkende HTML (.html)
3. De PHP (.php)
4. Screenshot van de werkende pagina (.jpg, .png)

7, Combineer technieken (integratieopdracht)

? Uitleg

Je kent nu zes *advanced prompt-technieken*. In de praktijk gebruik je ze vaak **samen**: je begint met een duidelijke context, kiest de juiste details, beperkt de output, vraagt een logische volgorde en laat de AI stap voor stap redeneren. In deze opdracht ga je laten zien dat je bewust kunt kiezen welke technieken je inzet — en waarom.

?? Opdracht

Je wil een AI vragen om een **kleine mini-webapplicatie** te maken waarin een gebruiker zijn **favoriete programmeertaal** kan kiezen en een korte uitleg krijgt over die taal.

Maak een prompt waarin je minstens **vier** van de volgende technieken toepast:

1. **Isolate the problem** – focus op de essentie van de taak.
2. **Provide lists in bullet points** – geef structuur aan je vraag.
3. **Provide the order** – geef aan in welke volgorde de AI moet werken.

4. **Few-shot prompting** – geef één of meer voorbeelden van input en gewenste output.
5. **Stel voorwaarden of beperkingen** – leg grenzen vast (zoals aantal regels, geen frameworks, enz.).
6. **Chain-of-thought prompting** – laat de AI eerst een plan maken en daarna pas uitvoeren.

Je prompt moet daarnaast voldoen aan de drie **basiskenmerken** van een goede prompt: **context**, **details** en **duidelijkheid**.

? Inleveren

1. Je volledige prompt met AI antwoord (.txt)
2. Een korte uitleg (minimaal 100 woorden) waarin je beantwoordt:
   - Aan welke **vier technieken** jouw prompt voldoet (.txt)
   - **Hoe** die technieken terugkomen in je prompt.
   - **Waarom** je juist deze technieken hebt gekozen.
   - Aan welke **basiskenmerken** (context, details, duidelijkheid) jouw prompt voldoet, en hoe dat in je tekst zichtbaar is.

8, Afsluitende Reflectie

? Uitleg

Je hebt nu geleerd hoe je met verschillende technieken betere prompts kunt schrijven en hoe je de AI kunt sturen met **context**, **details**, **duidelijkheid** en **doelgerichtheid**. In deze laatste opdracht denk je terug op wat je hebt geleerd — *niet over de inhoud van de code*, maar over je eigen denkproces.

?? Opdracht

Beantwoord de onderstaande vragen in je eigen woorden. Gebruik **geen AI-hulp** voor het schrijven van de tekst. De opdracht is bedoeld om te laten zien wat jij zélf hebt geleerd en ervaren.

> **Belangrijk:** De tekst wordt beoordeeld op echtheid en reflectievermogen. AI-gegenereerde teksten klinken vaak te algemeen of te perfect. Gebruik daarom concrete voorbeelden uit jouw eigen opdrachten en benoem specifieke situaties of keuzes.

Vragen

1. Welke **twee prompt-technieken** vond jij het meest nuttig, en waarom?
2. Beschrijf een moment waarop je eerst een slechte prompt maakte, maar daarna verbeterde. Wat heb je precies aangepast?
3. Welke **basiskenmerken** (context, details, duidelijkheid, doelgerichtheid, vorm, toon) vind jij het lastigst toe te passen? Leg uit waarom.
4. Hoe heeft het leren van deze technieken jouw manier van denken of werken met AI veranderd?
5. Wat zou je in een volgende opdracht anders aanpakken als je opnieuw met een AI werkt?

? Inleveren

1. Beantwoord de vragen één voor één en nummer de antwoorden. Lever PDF in.
2. De tekst bevat minimaal **200 woorden** en maximaal **400 woorden**.

> **Tip:** Vermijd vage zinnen als “Ik heb veel geleerd over prompts.” Schrijf concreet, bijvoorbeeld: “Bij opdracht 4 gebruikte ik eerst te veel code in mijn prompt. Toen ik het beperkte tot alleen het formulier, werkte het beter.”

JS 2 - (DOM1)

1 Introductievideo

datasource

? Leerdoelen

- Je weet wat de DOM is.
- Je weet dat je met JavaScript het DOM kan lezen en kan aanpassen.

? Uitleg

Bekijk de volgende video

https://www.roc.ovh/link/902#bkmrk-6-formulieren-in-html

https://www.youtube.com/embed/NO5kUNxGIu0

Code

Gebruik dit als start-code.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Simple Page</title>
  <style>body{font-family:Arial,sans-serif;background-color:#f0f4f8;padding:60px;text-
align:center}.container{background-color:#fff;padding:40px;border-radius:10px;box-shadow:0 4px
12px rgba(0,0,0,0.1);max-width:600px;margin:auto}h1{color:#2c3e50}p{color:#555;font-
size:1.1em}</style>
</head>
```

```
<body>

<div class="container">
  <h1 id="header">Embrace the Simplicity</h1>
  <p>Sometimes, all you need is a quiet page, a clear message, and a bit of whitespace. Let your thoughts breathe and focus on what truly matters.</p>
</div>

</body>
</html>
<script>

</script>
```

?? Opdracht

Bekijk de video daarin worden dingen voorgendaan, die jij ook moet toepassen.

1. Neem de start-code over in het bestand `dom1.html` en
2. Op regel 18 plaats code waarmee je de `backgroundColor` van de body grijs (grey) maakt.
3. Plaats vervolgens JavaScript code om tekst van de header met id "header" aan te passen naar:
   "Hallo slimme student".
4. In de video wordt dit beiden voorgedaan!

Je ziet aan het eind dus:

image.png

? Inleveren

- De aangepaste code (`dom1.html` bestand).

2 Elementen ophalen en aanpassen

? Leerdoelen

- 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 `dom2.html`.
2. Begin met de start-code (hieronder)
3. Als je op de knop klikt, moet de tekst in de paragraaf veranderen naar iets anders (bijv. “Hallo wereld!”).

Start code

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Voorbeeld</title>
</head>
<body>

<p id="mijnParagraaf">Dit is de originele tekst.</p>
<button onclick="verander()">Verander</button>
```

```
<script>
  function verander() {
    // plaats hier de code die de inhoud van de pragraaf veranderd.
  }
</script>

</body>
</html>
```

? 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 `dom2.html` in.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever deze mee in.

3 Meerdere elementen aanpakken

? Leerdoelen

- 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>

<button onclick="verander()">Verander</button>

<script>
  function verander() {
      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 `dom3.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 `geselecteerd` krijgen.
5. Als je weer op de knop drukt, dan moet de <p> weer terug veranderen (tip: gebuik **toggle**!).
6. Gebruik de voorbeeld code en pas die zelf aan.

Voorbeeld (gebruik zelf andere items)

image.png	image.png

? Reflectie

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

? Inleveren

- Lever het bestand `dom3.html` in.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

4 Interactie met knoppen en events

? Leerdoelen

- 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.”

```
<button id="klikMij">Klik mij (addEventListener)</button>

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

Dit doet hetzelde als:

```
<button onclick="toonMelding()">Klik mij (via functie)</button>

<script>
  function toonMelding() {
    alert("Je klikte op de knop!");
  }
</script>
```

Waarom zou je eventlisteners gebruiken?

Denk eerst zelf na en klik dan open!

Voordelen:

Scheidt structuur (HTML) van **gedrag (JS)** → netter.

Je kunt **meerdere functies** aan één event koppelen.

Flexibeler: makkelijk dynamisch events toevoegen of verwijderen.

Werkt ook bij elementen die pas later op de pagina verschijnen (bijv. via JavaScript geladen).

Nadelen:

Iets meer code nodig.

Niet altijd direct zichtbaar in de HTML wat er gebeurt (iets minder beginner-vriendelijk).

Andere events

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";
  });
});
```

Wat doet dit? Denk eerst na en controleer dan door hier te klikken.

Elk vak met de class "*kleurvak*" krijgt een gele achtergrond kleur als je er met de muis overheen gaat.

?? Opdracht – Events in actie

1. Maak een nieuw HTML-bestand `dom4.html`.
2. Voeg 5 divjes toe met de class `kleurvak`, elk met een vaste afmeting en een andere begin-kleur.
3. Als je met de muis over een vakje gaat, verandert de achtergrondkleur in geel.
4. Als je erop klikt, moet de tekst in het vakje veranderen naar “Geklikt!”.

Gebruik zowel `mouseover` als `click` events.

Gebruik dit als basis voor de CSS:

```
<style>
  .kleurvak {
    width: 100px;
    height: 100px;
    display: inline-block;
    margin: 10px;
    text-align: center;
    line-height: 100px;
    background-color: lightblue;
    font-weight: bold;
    cursor: pointer;
  }
</style>
```

Resultaat

https://youtu.be/deDyTevLTgk

https://www.youtube.com/embed/deDyTevLTgk

? 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 `dom4.html` in.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

5 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>
```

- Op regel **7** wordt de waarde van het HTML elemenet met het id `tekstvlak` gelezen en in de variabel `invoer` gezet.
- Op regel **8** wordt er een nieuwe paragraaf gemaakt.
- Op regel **9** wordt het nieuwe element gevuld met tekst.
- Op regel **10** wordt het nieuwe element toegevoegd aan de de `div` met id *resultaat*

?? Opdracht – Invoer toevoegen

1. Maak een bestand `dom5.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.
5. Bonus
   Als je het leuk vindt, maak je het invoerveld na het klikken automatisch weer leeg.
   Gebruik daaarvoor `input.value = ""` om het veld te legen.

? Reflectie

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

? Inleveren

- Lever je bestand `dom5.html` in.

- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

6 Elementen 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 `click`), kun je met `event.target` achterhalen welk element er 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>
```

Er wordt een HTML lijst gemaakt en er wordt op elke element van de lijst een click event geplaatst. Dit event verwijderd zichzelf.

?? Opdracht – Klik en verwijder

1. Maak een bestand `dom6.html`.
2. Voeg een lijst toe (bijv.) 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()` binnen je event handler.

? Reflectie

- Wat is `event.target` en waar gebruik je het voor?
- Bedenk en beschrijf een nuttige toepassing waarbij je meet `event.target` iets wil aanpassen of verwijderen.

? Inleveren

- Lever je bestand `dom6.html` in.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

7 Herhaling – Interactieve favorietenlijst

We gaan een aantal onderdelen die we in de opdrachten hebben behandeld nu combineren en we gaan onze eigen interacteive lijst maken.

Wat gaan we maken?

https://www.youtube.com/embed/CxEQ1B4m-I0

? Leerdoelen

- Je kunt elementen aanmaken, toevoegen en verwijderen met JavaScript.
- Je kunt reageren op events zoals klikken en rechtermuisklikken.
- Je kunt meerdere elementen selecteren en aanpassen.

? Uitleg

In deze opdracht herhaal je alles wat je geleerd hebt in de vorige opdrachten. Je maakt een interactieve lijst waarin je dingen kunt toevoegen, markeren en verwijderen.

?? Opdracht – Favorietenlijst

1. Maak een bestand `dom7.html`.
2. Voeg een **invoerveld** toe waarin de gebruiker een favoriete hobby, snack of film kan typen.
3. Voeg een **knop** toe met de tekst `Voeg toe`.
4. Onder de knop komt een lijst (<ul>) waarin elke keer een **nieuw item** verschijnt als de gebruiker op de knop klikt.
5. Wanneer je op een item klikt, **verandert** de **tekstkleur** of opmaak (gebruik `classList.toggle()`).
6. Wanneer je met de **rechtermuisknop** op een item klikt, moet dat item **verdwijnen** (gebruik `event.preventDefault()`).
7. Boven de lijst staat een **teller** met hoeveel items er in de lijst staan. Deze teller werkt automatisch.

? Reflectie

- Wat heb je allemaal moeten combineren uit de vorige lessen?
- Waarom is `event.preventDefault()` nodig bij het gebruik van de rechtermuisknop?
- Wat is het voordeel van `classList.toggle()` ten opzichte van `add()` en `remove()`?
- Waarom moet je elementen ophalen met `getElementById` of `querySelector` nadat de HTML geladen is?

? Inleveren

- Lever je bestand `dom7.html` in.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

8 Debuggen met de console

? Leerdoelen

- Je weet wat de `console` is in de browser.
- Je kunt fouten (errors) in JavaScript herkennen in de console.
- Je kunt `console.log()` gebruiken om waarden te controleren.

? Uitleg

De console is een hulpmiddel in je browser waarmee je kunt zien wat er “onder de motorkap” van je code gebeurt. Je opent de console met **F12** of via **Rechtermuisknop → Inspecteren → Console**.

Als er iets misgaat, verschijnt er een foutmelding (bijvoorbeeld in rood). Vaak vertelt die precies waar de fout zit — bijvoorbeeld in welke regel of welk bestand.

Voorbeeld

```
<script>
  const naam = "Max";
  console.log("De naam is:", naam);

  // Fout voorbeeld
  console.log(voornaam); // ReferenceError: voornaam is not defined
</script>
```

De foutmelding vertelt je dat de variabele `voornaam` niet bestaat. Zo kun je fouten snel vinden en oplossen.

?? Opdracht – Debuggen met de console

1. Maak een nieuw bestand `dom8.html`.
2. Kopieer onderstaande code:

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Debug oefening</title>
</head>
<body>
  <h1>Welkom op mijn pagina</h1>
  <p id="tekst">De tekst wordt hieronder aangepast...</p>

  <script>
    const tekstElement = document.getElementById("tekts");
    tekstElement.textContent = "Nieuwe tekst";
    console.log("Code is uitgevoerd!");
  </script>
</body>
</html>
```

3. Open je pagina in de browser.
4. Gebruik de **console (F12)** en zoek de foutmelding.
5. Noteer wat er misgaat en op welke regel de fout verschijnt.
6. Corrigeer de fout en controleer of de tekst nu wel verandert.
7. Gebruik daarna `console.log()` om te controleren of je variabele correct gevuld is.

? Reflectie

- Wat betekent de foutmelding die je kreeg?
- Hoe hielp de console jou om de fout te vinden?
- Wat is het verschil tussen een foutmelding (error) en een melding via `console.log()`?

? Inleveren

- Lever het bestand `dom8.html` in (met de fout opgelost).
- Lever een kort reflectieverslag (.txt of .pdf) waarin je de drie reflectievragen beantwoordt.

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.

Java Script Challenge

DOM Challenge – Bouw jouw eigen mini-app

datasource

? 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 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.

? Eisen

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

- Een of meer **invoervelden** + **knoppen**
- Een **lijst** of reeks elementen die via JavaScript groeit
- Een invoerveld met een **search** functie waarbij als je die gebruikt alleen de gezochte items worden getoond.
- Na een search moet je ook weer terug gaan en **alle** elementen kunnen laten zien.
- **Event-handling** (bijv. click, submit)
- Bij een delete van een item, vraag je om eerst om **bevestiging**: "Weet je zeker dat je item 'voorbeel' wilt verwijderen?"
- Je maakt tenminste gebruik van `querySelector`, `addEventListener`, en `innerHTML`, `classList.toggle()` en `remove()`

Bij het project **Boekenkast** moet je daarnaast ook werken met `localStorage`.

? Optioneel

- Als je alles in één pagina maakt, dan kan je er eventueel vrij eenvoudig een app voor je telefoon van maken.

?? Opdracht – Kies en bouw jouw mini-app

1. Maak een bestand met een duidelijke naam, bijv. `dom-project.html`.
2. Kies welk 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 (DOM)
6. Stijl je pagina met CSS zodat deze overzichtelijk en prettig in gebruik 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!
- **Je laat het resultaat aan de docent zien en je kan de code uitleggen.**

Kennis Check Blok 5

Kennis Check Blok 5

datasource

Heb je deze vragen doorgenomen?

?? Opdracht

Maak de kennis-check.

? Inleveren

Aan het einde van de kennis-check ontvang je een certificaat. Maak een schermafdruk en lever deze in.