

Blok 5 - PHP en JS (DOM)

- [PHP 2](#)
- [Database 1](#)
- [JS - \(DOM\)](#)
- [JS - \(DOM\) - Extra](#)
- [Java Script Challenge](#)

PHP 2

1 Foutafhandeling en Basis Debugging in PHP

Leerdoelen

- Je weet wat de verschillende typen PHP-fouten zijn (`Errors`, `Warnings`, `Notices`).
- Je kunt foutmeldingen in PHP interpreteren.
- Je kunt foutweergave instellen met `error_reporting()` en `ini_set()`.
- Je kunt basisprincipes van foutafhandeling toepassen (zoals `isset()`, `empty()`, `file_exists()`).
- Je gebruikt debuggingtechnieken zoals `echo`, `print_r()` en `var_dump()`.

Uitleg

Soorten fouten in PHP

- **Parse error:** fout in de code (bijvoorbeeld vergeten ; of haakje).
- **Fatal error:** code probeert iets wat niet kan, bijvoorbeeld een functie die niet bestaat.
- **Warning:** foutmelding, maar de code gaat door.
- **Notice:** melding van iets wat waarschijnlijk fout is (bijv. niet bestaande variabele).

⚙️ Foutmeldingen tonen of verbergen

```
<?php
// Toon alle fouten
ini_set("display_errors", 1);
error_reporting(E_ALL);
```

```
?>
```

⚠ Op een live website wil je foutmeldingen verbergen voor gebruikers. Gebruik dan:

```
ini_set("display_errors", 0);
```

☐☐ Veelgebruikte foutafhandelingstechnieken

- `isset($variabele)` – controleert of een variabele bestaat.
- `empty($variabele)` – controleert of een variabele leeg is.
- `file_exists("bestand.txt")` – controleert of een bestand bestaat.
- `die("Foutmelding")` of `exit()` – stopt het script bij een ernstige fout.

☐☐ Debuggen zonder debugger

- `echo` – handig om waardes snel te tonen.
- `print_r($array)` – toont de inhoud van een array of object.
- `var_dump($variabele)` – toont type + waarde (ook handig bij fouten met getallen).

☐☐ Opdracht 1 – Debugging oefenen

1. Maak een PHP-bestand genaamd `korting.php` met deze functie:

```
<?php
function berekenKorting($prijs, $korting) {
    return $prijs - $prijs * $korting / 100;
}
echo berekenKorting(100); // Fout! Tweede parameter ontbreekt
?>
```

2. Voeg foutmeldingen toe via `error_reporting()` zodat je de waarschuwing ziet.
3. Gebruik `isset()` en `empty()` om de fout af te vangen.
4. Test wat er gebeurt als je een variabele gebruikt die niet bestaat.
5. Gebruik `var_dump()` en `echo` om te zien wat je fout doet.

☐☐ Reflectie

- Wat is het verschil tussen een warning en een fatal error?
- Waarom is het handig om fouten wel te tonen in de ontwikkelfase, maar niet op een live website?
- Welke debuggingtechniek vond je het meest bruikbaar?

☐☐ Inleveren

- Lever het bestand `korting.php` in.
- Lever een korte uitleg in (.txt of .pdf) waarin je aangeeft:
 - Welke fout(en) je hebt gevonden
 - Welke techniek je gebruikte om het op te lossen
 - Wat je ervan geleerd hebt

2 Bestanden en Loggen

☐☐ Leerdoelen

- Je kunt bestanden aanmaken, lezen, schrijven en toevoegen met PHP.
- Je weet wanneer het handig is om gegevens in bestanden op te slaan in plaats van een database te gebruiken.
- Je begrijpt wat logbestanden zijn en waarom ze nuttig zijn.
- Je kunt een eenvoudig logsysteem implementeren dat fouten of gebeurtenissen opslaat.

☐☐ Uitleg

☐☐ Waarom bestanden gebruiken?

Bestanden kunnen handig zijn voor eenvoudige opslag zoals instellingen, bezoekerslogs of tijdelijke data. In kleine projecten is dit vaak eenvoudiger dan een database.

☐☐ Bestandsfuncties in PHP

- `file_put_contents("bestand.txt", "tekst")` – schrijft tekst naar bestand (overschrijft).
- `file_get_contents("bestand.txt")` – leest hele bestand in één keer.
- `fopen()` + `fwrite()` – uitgebreidere controle (bijv. toevoegen).
- `file_exists("bestand.txt")` – controleert of bestand bestaat.
- `fclose()` – sluit een bestand (nodig bij `fopen()`).

☐☐ Loggen: fouten en gebeurtenissen bijhouden

Een logbestand houdt bij wat er gebeurt in een script. Bijvoorbeeld foutmeldingen of bezoekersactiviteit.

Voorbeeld – logregel opslaan:

```
<?php
$melding = "[" . date("Y-m-d H:i:s") . "] Foutmelding\n";
file_put_contents("error.log", $melding, FILE_APPEND);
?>
```

`FILE_APPEND` zorgt dat de nieuwe regel onderaan toegevoegd wordt in plaats van het bestand te overschrijven.

☐☐ Opdracht 1 – bezoekersteller.php

1. Maak een nieuw bestand `bezoekersteller.php`.
2. Laat het script een teller bijhouden in `teller.txt`:
 - Bestaat het bestand nog niet? Begin bij 1.
 - Zo niet? Lees het getal in, verhoog met 1, en schrijf het terug.
3. Toon het aantal bezoeken op het scherm.

☐☐ Opdracht 2 – gastenboek.php (uitbreiding)

1. Maak een script waarin een gebruiker een bericht kan achterlaten via een formulier.
2. Sla elk bericht op in `gastenboek.txt` met datum/tijd.

3. Toon de laatste 5 berichten boven het formulier.
4. Gebruik `htmlspecialchars()` om invoer veilig weer te geven.

☐☐ Reflectie

- Wat zijn de voor- en nadelen van gegevens opslaan in een bestand ten opzichte van een database?
- Wat gebeurt er als je `file_put_contents()` gebruikt zonder `FILE_APPEND`?
- Waarom is loggen belangrijk, zelfs in kleine projecten?

☐☐ Inleveren

- Lever je bestanden `bezoekersteller.php` en/of `gastenboek.php` in.
- Lever een reflectieverslag in (.txt of .pdf).

3 Geavanceerde Functies, Abstractie en Modulaire Code

☐☐ Doelstellingen

- Je begrijpt waarom functies handig zijn (herbruikbaarheid, leesbaarheid, structuur).
- Je kunt functies gebruiken om complexe taken te verbergen (abstractie).
- Je snapt het verschil tussen globale en lokale variabelen (scope).
- Je kunt functies opslaan in aparte bestanden en deze inladen met `include()` of `require()`.

☐☐ Uitleg

☐☐ Waarom functies gebruiken?

- Je hoeft een stuk code maar één keer te schrijven.

- Je kunt de code op meerdere plekken gebruiken.
- Je maakt je code overzichtelijker en makkelijker te begrijpen.

☐ Abstractie: de details verbergen

Een functie kan iets ingewikkelds doen, zonder dat je telkens opnieuw de details hoeft te schrijven.

```
function berekenKorting($bedrag, $percentage) {  
    return $bedrag - ($bedrag * $percentage / 100);  
}  
  
echo berekenKorting(100, 10); // → 90
```

Je hoeft niet telkens opnieuw de hele berekening te typen. De functie “verbergt” die logica.

☐ Variable scope (bereik van een variabele)

- Variabelen binnen een functie zijn **lokaal**: ze bestaan alleen daarbinnen.
- Variabelen buiten functies zijn **globaal**.
- Je kunt met `global` een globale variabele binnen een functie gebruiken, maar dat is niet altijd wenselijk.

```
$x = 5;  
  
function testScope() {  
    $x = 10;  
    echo $x; // Toont 10, NIET 5  
}  
  
testScope();  
echo $x; // Toont nog steeds 5
```

☐ Modulaire code met `include()`

Je kunt functies in een apart bestand zetten, bijvoorbeeld `utils.php`, en die inladen in andere bestanden.

```
// Bestand: utils.php  
function toonWelkomstbericht($naam) {  
    echo "<p>Welkom, $naam!</p>";  
}
```

```
// Bestand: index.php
include("utils.php");
toonWelkomstbericht("Ali");
```

Je kunt ook meerdere functies in één bestand zetten, zoals:

- `validation.php`: functies voor formuliercontrole
- `format.php`: functies voor tekst- of getalopmaak

Opdracht – utils.php gebruiken

1. Maak een nieuw bestand `utils.php` en schrijf daarin minstens 3 functies:
 - `formatteerBedrag($bedrag)` → toont bijv. "€ 12,50"
 - `valideerInvoer($waarde)` → controleert of de waarde niet leeg is
 - `toonWelkomstbericht($naam)` → toont HTML met de naam
2. Maak een ander bestand `test_utils.php` waarin je `include("utils.php")` gebruikt en de functies test.
3. Gebruik `echo`, `print_r()` en `var_dump()` om het resultaat van de functies te tonen.

Reflectie

- Wat zijn de voordelen van functies in aparte bestanden bewaren?
- Welke functie vond je het handigst om te schrijven en waarom?
- Wat heb je geleerd over het verschil tussen globale en lokale variabelen?

Inleveren

- Lever de bestanden `utils.php` en `test_utils.php` in.
- Lever een reflectiedocument in (.txt of .pdf) waarin je uitlegt wat je hebt gedaan en geleerd.

4 Form Validatie en Beveiliging

Doelstellingen

- Je kunt formulierinvoer controleren op geldigheid.
- Je kent de risico's van onveilige invoer (zoals XSS).
- Je kunt invoer ontsnappen met `htmlspecialchars()`.
- Je gebruikt functies als `isset()`, `empty()` en `trim()` om invoer te valideren.

Uitleg

Waarom valideren?

Gebruikers maken fouten. Hackers doen het expres. Daarom controleer je altijd of een formulier goed is ingevuld voordat je ermee werkt.

Veelvoorkomende gevaren

- **XSS (Cross-Site Scripting):** iemand vult HTML of JavaScript in een formulier in, dat op jouw pagina wordt getoond.
- **Lege velden:** de gebruiker vergeet iets in te vullen.
- **Verkeerde types:** een tekst waar een getal moet staan.

Handige functies voor validatie

- `isset($_POST["naam"])` – is het veld verzonden?
- `empty($_POST["naam"])` – is het veld leeg?
- `trim()` – verwijdert spaties voor en na de invoer.
- `htmlspecialchars()` – maakt HTML onschadelijk.

```
<?php
if ($_SERVER["REQUEST_METHOD"] === "POST") {
    $naam = trim($_POST["naam"]);
    if (empty($naam)) {
        echo "Vul je naam in.";
    } else {
        echo "Welkom, " . htmlspecialchars($naam);
    }
}
```

```
}  
?>
```

☐☐ Simpel formulier

```
<form method="post">  
  <input type="text" name="naam" placeholder="Je naam">  
  <input type="submit" value="Verstuur">  
</form>
```

☐☐ Opdracht 1 – formulier_validatie.php

1. Maak een PHP-bestand met een formulier waarin je de gebruiker vraagt om:
 - Naam
 - Leeftijd
 - Bericht
2. Controleer of alle velden zijn ingevuld. Toon een foutmelding als iets ontbreekt.
3. Gebruik `htmlspecialchars()` om het bericht veilig weer te geven.
4. Gebruik `is_numeric()` om te controleren of de leeftijd een getal is.

Extra (optioneel)

- Maak je foutmeldingen visueel opvallend met HTML (bijv. ``).

☐☐ Reflectie

- Waarom is het belangrijk om input van gebruikers altijd te controleren?
- Wat zou er kunnen gebeuren als je `htmlspecialchars()` niet gebruikt?
- Welke fouten kwamen er tijdens het testen naar boven?

☐☐ Inleveren

- Lever je bestand `formulier_validatie.php` in (.php).

- Lever een reflectiedocument in (.txt of .pdf) met jouw antwoorden.

5 Superglobals en Associatieve Arrays

Leerdoelen

- Je begrijpt wat superglobals zijn in PHP (`$_GET`, `$_POST`, `$_SESSION`, `$_FILES`).
- Je weet wat een associatieve array is en hoe je ermee werkt.
- Je kunt formulierdata opslaan en verwerken met behulp van superglobals.

Uitleg

Wat zijn superglobals?

Superglobals zijn ingebouwde variabelen in PHP die overal beschikbaar zijn. Je gebruikt ze bijvoorbeeld om gegevens uit een formulier op te halen.

- `$_POST` – bevat formulierdata die via POST is verzonden
- `$_GET` – bevat data uit de URL (bijv. `?pagina=contact`)
- `$_SESSION` – bevat gegevens die je wilt onthouden tussen pagina's
- `$_FILES` – bevat geüploade bestanden

Associatieve arrays

Een associatieve array is een array met 'sleutels' in plaats van indexnummers:

```
$persoon = [  
    "naam" => "Ali",  
    "leeftijd" => 19,  
    "email" => "ali@example.com"  
];  
  
echo $persoon["naam"]; // Toont: Ali
```

Formulieren leveren ook associatieve arrays op via `$_POST`:

```
echo $_POST["naam"];
```

☐☐ Opdracht 1 – formulier_superglobals.php

1. Maak een formulier met de volgende velden:

- Naam
- E-mail
- Bericht

2. Verwerk de invoer met `$_POST` en sla de gegevens op in een associatieve array:

```
$data = [  
    "naam" => $_POST["naam"],  
    "email" => $_POST["email"],  
    "bericht" => $_POST["bericht"]  
];
```

3. Toon de waarden netjes op het scherm met `htmlspecialchars()`.

4. Gebruik eventueel `print_r($data)` om de hele array te tonen voor debugging.

☐☐ Reflectie

- Wat is het verschil tussen een normale array en een associatieve array?
- Waarom is `$_POST` eigenlijk een associatieve array?
- Wanneer zou je liever `$_GET` gebruiken in plaats van `$_POST`?

☐☐ Inleveren

- Lever je bestand `formulier_superglobals.php` in (.php).
- Lever een reflectiedocument in (.txt of .pdf) met je antwoorden.

6 Sessies en Cookies

Doelstellingen

- Je begrijpt wat sessies en cookies zijn.
- Je kunt sessies gebruiken om gegevens tijdelijk op te slaan voor een gebruiker.
- Je kunt cookies aanmaken en uitlezen met PHP.
- Je kunt de juiste techniek kiezen om gebruikersgegevens te bewaren.

Uitleg

Wat is een sessie?

Een sessie is een manier om informatie te onthouden zolang een gebruiker actief is op de website. Bijvoorbeeld: je logt in en blijft ingelogd op alle pagina's.

Een sessie start je met:

```
<?php
session_start();
$_SESSION["naam"] = "Ali";
echo $_SESSION["naam"];
?>
```

Je kunt sessiegegevens gebruiken zolang de browser open is (of tot je ze verwijdert met `session_destroy()`).

Wat is een cookie?

Een cookie wordt opgeslagen in de browser van de gebruiker, meestal voor een langere tijd. Handig om voorkeuren of gebruikersgegevens te onthouden tussen bezoeken.

```
// Cookie instellen
setcookie("taal", "NL", time() + 3600); // 1 uur geldig

// Cookie uitlezen
echo $_COOKIE["taal"];
```

☐ Verschillen tussen sessies en cookies

Kenmerk	Sessie	Cookie
Opslaglocatie	Server	Browser van gebruiker
Levensduur	Tijdelijk (tot de browser sluit of je het wist)	In te stellen (bijv. 1 uur, 30 dagen)
Toepassing	Ingelogde gebruiker, winkelmandje	Voorkeuren, laatst bezochte pagina

☐ Opdracht 1 – sessie_en_cookie.php

1. Maak een pagina waar de gebruiker zijn naam kan invoeren via een formulier.
2. Na verzenden:
 - Sla de naam op in een **sessie** én in een **cookie** (1 uur).
 - Toon op de pagina: “Welkom terug, [naam]!” als de gebruiker opnieuw langskomt.
3. Laat ook zien wat er gebeurt als de gebruiker het formulier overslaat.
4. Gebruik `session_start()` bovenaan het script.

Extra (optioneel)

- Voeg een link toe die de sessie wist (`session_destroy()`).

☐ Reflectie

- Wanneer kies je voor een sessie? En wanneer voor een cookie?
- Wat gebeurt er als je `session_start()` vergeet?
- Waar moet je op letten bij het gebruik van cookies met privacy?

☐ Inleveren

- Lever het bestand `sessie_en_cookie.php` in (.php).
- Lever een reflectiedocument in (.txt of .pdf) met je antwoorden.

7 State en Bestandgebaseerde 'Database'

Leerdoelen

- Je begrijpt het concept van 'state' in webapplicaties.
- Je kunt gebruikersgegevens opslaan in een tekstbestand.
- Je kunt meerdere records opslaan als gestructureerde regels (bijv. JSON of CSV).
- Je kunt gegevens uit zo'n bestand inlezen en tonen aan de gebruiker.

Uitleg

Wat is 'state'?

Een webpagina 'vergeet' wat er net gebeurd is zodra je hem ververs. Daarom moet je zelf bijhouden wat de toestand (state) is van je applicatie.

Je kunt dat doen met sessies, cookies of door gegevens op te slaan in een bestand of database.

Bestanden als 'mini-database'

In plaats van een echte database zoals MySQL, kun je voor simpele toepassingen gegevens bewaren in een tekstbestand.

Bijvoorbeeld: als iemand een bericht achterlaat in een formulier, voeg je dat toe aan `data.txt`.

Structuur

- Elk bericht komt op een eigen regel
- Of je gebruikt een JSON-array, met meerdere objecten erin
- Of je gebruikt CSV (waardes gescheiden met komma's)

```
// Voeg toe aan tekstbestand
$bericht = htmlspecialchars($_POST["bericht"]);
```

```
file_put_contents("data.txt", $bericht . "\n", FILE_APPEND);
```

```
// Lees het bestand
```

```
$inhoud = file("data.txt");
```

```
foreach ($inhoud as $regel) {
```

```
    echo "<p>" . trim($regel) . "</p>";
```

```
}
```

Opdracht 1 – gastenboek_met_bestand.php

1. Maak een formulier waarin iemand een naam en een bericht kan achterlaten.
2. Als de gebruiker op "Verstuur" klikt:
 - Sla het bericht op in een bestand `gastenboek.txt`.
 - Voeg ook datum/tijd toe met `date()`.
3. Toon de laatste 5 berichten onder het formulier.
4. Zorg dat de HTML veilig blijft via `htmlspecialchars()`.

Extra (optioneel)

- Laat de berichten in omgekeerde volgorde zien (nieuwste bovenaan).
- Sla de data op in JSON-formaat i.p.v. tekstregels.

Reflectie

- Wat zijn de voordelen van een tekstbestand boven een database?
- Wanneer loop je tegen de beperkingen aan?
- Hoe zou je dit uitbreiden zodat iemand ook berichten kan verwijderen?

Inleveren

- Lever je bestand `gastenboek_met_bestand.php` in (.php).

- Lever het bestand `gastenboek.txt` mee met minimaal 3 testberichten.
- Lever een reflectie in (.txt of .pdf).

8 Inleiding tot PHP Include-logica en Templatebestanden

Doelstellingen

- Je begrijpt het nut van het opdelen van HTML/PHP-bestanden in herbruikbare componenten.
- Je kunt `include` en `require` gebruiken om logica en opmaak te splitsen.
- Je kunt een eenvoudige template-structuur bouwen (header, content, footer).

Uitleg

Waarom opdelen in componenten?

Als je meerdere pagina's hebt met dezelfde header of footer, is het onhandig om die steeds te kopiëren. Daarom gebruik je `include()` of `require()` om ze in te laden.

```
<?php include 'header.php'; ?>

<h1>Welkom op de homepage</h1>

<?php include 'footer.php'; ?>
```

Verschil tussen include en require

- `include()`: Laadt het bestand in. Als het bestand niet bestaat, gaat de rest van de pagina gewoon door.
- `require()`: Laadt ook het bestand in, maar als het niet bestaat, stopt de pagina met een foutmelding.

Voorbeeldstructuur

- `header.php` – HTML <head>, navigatie
- `footer.php` – Copyright info, afsluitende HTML
- `index.php` – Hoofdpagina die alles samenvoegt

☐☐ Opdracht 1 – templatestructuur

1. Maak drie bestanden:
 - `header.php`: bevat een <header> met de titel van je site en een eenvoudige navigatie (bijv. naar home en contact).
 - `footer.php`: bevat een <footer> met een copyrightregel.
 - `index.php`: bevat de pagina-inhoud en gebruikt `include()` om de header en footer in te laden.
2. Zorg dat alles er netjes uitziet (je mag CSS toevoegen).
3. Test wat er gebeurt als je `include("footer.php")` verandert in `require("footer.php")` en het bestand bestaat niet.

☐☐ Reflectie

- Waarom is het handig om componenten zoals header/footer apart te houden?
- Wat is het risico als je alles in één bestand zou houden?
- Wanneer zou je `require()` verkiezen boven `include()`?

☐☐ Inleveren

- Lever `index.php`, `header.php` en `footer.php` in.
- Lever een reflectiedocument in (.txt of .pdf) met je antwoorden.

9 Validatie en Veiligheid bij Formulieren

Doelstellingen

- Je begrijpt waarom inputvalidatie belangrijk is.
- Je kunt formulierinvoer controleren met PHP (server-side).
- Je kent de risico's van onveilige invoer (bijv. XSS, SQL-injectie).
- Je kunt input veilig maken met functies zoals `htmlspecialchars()` en `filter_var()`.

Uitleg

Waarom valideren?

Gebruikers kunnen fouten maken (of expres verkeerde dingen invullen). Als je invoer niet controleert, kan dat leiden tot bugs of zelfs beveiligingsproblemen.

Soorten validatie

- **Client-side:** Met HTML of JavaScript (bijv. `required`, `type="email"`).
- **Server-side:** Met PHP (altijd nodig, want je kunt client-side validatie omzeilen).

Veiligheid: wat kan er misgaan?

- **XSS (Cross-Site Scripting):** Als je niet ontsnapte HTML toont, kan iemand scripts injecteren.
- **SQL-injectie:** Als je invoer rechtstreeks in een query zet (komt later aan bod).

Voorbeeld inputcontrole

```
<?php
$naam = "";
$fout = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["naam"])) {
        $fout = "Naam is verplicht!";
    } else {
        $naam = htmlspecialchars($_POST["naam"]);
    }
}
```

```
}  
}  
?>  
  
<form method="post">  
  Naam: <input type="text" name="naam">  
  <input type="submit" value="Verstuur">  
</form>  
  
<?php  
if ($fout) {  
  echo "<p style='color:red;'>$fout</p>";  
} else if ($naam) {  
  echo "<p>Welkom $naam</p>";  
}  
?>
```

☐☐ Opdracht 1 – formuliercontrole

1. Maak een formulier met de volgende velden:
 - Naam (verplicht)
 - Email (verplicht, geldig e-mailadres)
 - Bericht (optioneel, max. 200 tekens)
2. Controleer de invoer met PHP:
 - Laat foutmeldingen zien als iets ontbreekt of fout is.
 - Gebruik `filter_var()` om het e-mailadres te controleren.
 - Gebruik `htmlspecialchars()` om veilige output te tonen.
3. Toon een nette samenvatting van de ingevulde gegevens als alles goed is.

Extra (optioneel)

- Voeg visuele validatie toe met CSS-klassen (bijv. rode rand bij fout).
- Voeg een resetknop toe aan het formulier.

☐☐ Reflectie

- Wat gebeurt er als je geen server-side validatie gebruikt?
- Wat is het verschil tussen validatie en het veilig maken van input?
- Hoe zou je dit formulier uitbreiden voor een login- of registratiepagina?

Inleveren

- Lever je formulierpagina in als `formulier_validatie.php`.
- Lever een reflectiedocument in (.txt of .pdf).

Database 1

1 Wat is een database?

Leerdoelen

- Je weet wat een database is en waarvoor die gebruikt wordt.
- Je kunt gegevens uit de echte wereld omzetten naar tabellen en kolommen.
- Je begrijpt het verschil tussen ruwe gegevens en een gestructureerd model.

Uitleg

In het dagelijks leven slaan organisaties gegevens op: over klanten, producten, studenten, boeken, etc. Een **database** is een digitale plek waar zulke gegevens netjes georganiseerd worden bewaard. Je verdeelt de informatie over verschillende tabellen, waarbij elke tabel over één onderwerp gaat, zoals:

- **Studenten:** naam, klas, geboortedatum
- **Docenten:** naam, vak, afdeling
- **Opleidingen:** naam, niveau, duur

Een database lijkt een beetje op een Excel-bestand, maar is veel krachtiger en beter gestructureerd. Je wilt geen dubbele gegevens en alles moet logisch met elkaar verbonden zijn.

Voorbeeld: Dit is géén goede database:

Naam	Klas	Opleiding
-----	-----	-----
Fatima	SD1A	Software Developer
Ali	SD1A	Software Developer
Robin	SD2B	Software Developer
Jente	SD1A	Software Developer
Steven	SD1A	Recht & HR

→ Hier zie je dat de opleiding meerdere keren herhaald wordt. Dat is zonde en foutgevoelig.

☐☐ Wat zou beter zijn?

Je zou in plaats daarvan een aparte tabel 'Studenten' maken en een aparte tabel 'Opleidingen'. Studenten krijgen dan een *verwijzing* naar hun opleiding (dat komt in de volgende lessen aan bod).

☐☐ Opdracht 1 – Gegevens analyseren

1. Bekijk onderstaande lijst met gegevens:

Naam: Esra
Klas: SD1A
Opleiding: Software Developer
Docent: De Jong
Vak: Webontwikkeling
Lesdag: Woensdag

2. Welke verschillende **onderwerpen** zie je hierin? Probeer per onderwerp de eigenschappen op te schrijven.
 - **Student:** naam, klas
 - ... (jij vult aan)
3. Zet je antwoorden in een tabelvorm: welke tabellen zou je nodig hebben? Welke kolommen zouden erin staan?

☐☐ Reflectie

- Waarom is het vastleggen van gegevens op meerdere plaatsen foutgevoelig? Beschrijf een situatie waarin dat fout kan gaan.
- Waarom is het niet handig om alle informatie in één grote tabel te zetten?
- Wat denk je dat het voordeel is van losse tabellen met verbindingen?

☐☐ Inleveren

- Lever je tabellenindeling in (.txt, .pdf of screenshot).

- Voeg je antwoorden toe op de reflectievragen.

2 Entiteiten en Attributen

Leerdoelen

- Je weet wat een entiteit is en wat een attribuut is.
- Je kunt entiteiten en hun attributen herkennen in een realistisch scenario.
- Je kunt een eerste versie van een ERD tekenen met entiteiten en attributen.

Uitleg

Entiteit

In de vorige opdracht heb je '**onderwerpen**' en '**eigenschappen**' bepaald, weet je het nog? Een onderwerp was 'student' en een eigenschap was 'naam' en 'klas'.

Bij het ontwerpen van een database heet een **onderwerp** een '**entiteit**', en een **eigenschap** is een '**attribuut**'

Een **entiteit** is een "**ding**", "**persoon**" of "**gebeurtenis**" in de echte wereld waarover je gegevens wilt opslaan.

Bijvoorbeeld:

- Student -> persoon
- Cursus -> gebeurtenis
- Docent -> persoon
- Opleiding -> gebeurtenis
- Laptop -> ding

Attribuut

Een **attribuut** is een eigenschap van een entiteit. Bijvoorbeeld:

- De entiteit **Student** heeft de attributen: **studentnummer**, **voornaam**, **achternaam**, **telefoonnummer**.
- De entiteit **Cursus** heeft de attributen: **naam**, **duur**, **startdatum**.

ERD

Een entiteit met de attributen zet je in een bepaald formaat in een database ontwerp. Dat heet een ERD.

Op de eerste regel staat de **naam van de entiteit** en in de middelste kolom zet je alle **attributen**.

Voorbeeld:

image.png
Image not found or type unknown

In een ERD (Entity Relationship Diagram) teken je entiteiten als rechthoeken en attributen als ovale of gelabelde velden ernaast.

In een ERD heb je **drie kolommen** de eerste voor de **keys**, de tweede voor de **attribuutnamen** en de derde voor de **datatypes**.

Opdracht 1 – Entiteiten en attributen

- Je werkt voor een evenementenbureau. Zij willen bijhouden:
 - Welke klanten boekingen doen
 - Welke evenementen er zijn
 - Welke locaties beschikbaar zijn
- Maak een lijstje van minstens 3 entiteiten uit deze situatie. Bijvoorbeeld:
 - **Klant**
 - ...
 - ...
- Geef per entiteit minstens 3 bijpassende attributen. Bijv.:
 - Klant → voornaam, e-mailadres, ...
 - Locatie → naam,

4. Teken jouw eerste ERD in [Lucidchart](#).

Lucichart

Registreer je voor Lucichart en maak een gratis account.

Kies toevoegen library

image.png and or type unknown

Zoek naar ERD en selecteer "Entity Relationship"

image.png and or type unknown

Gebruik vervolgens dit figuur om een entiteit te maken.

image.png and or type unknown

☐☐ Reflectie

- Hoe weet je of iets een entiteit was of gewoon een attribuut?
- Heb je misschien dingen dubbel in verschillende entiteiten? Kun je iets beter loskoppelen?

☐☐ Inleveren

- Maak je ERD in Lucichart en maak een screenshot.

3 Primary Keys

☐☐ Leerdoelen

- Je weet wat een **primary key** is.
- Je begrijpt waarom een primary key verplicht is in elke tabel.
- Je kunt per entiteit een geschikte primary key kiezen.

Uitleg

Wat is een Primary Key (PK)?

Een primary key is een uniek gegeven waarmee je één rij uit een tabel kunt identificeren. Elke tabel in een database **moet** een primary key hebben.

Regel: Elke entiteit heeft precies één PK. Heb je meerdere opties? Kies er één. Heb je geen goede kandidaat? Gebruik dan een kunstmatige sleutel, zoals een `id`.

Voorbeelden:

- `Student` → `student_nr`
- `Auto` → `kenteken`
- `Evenement` → `evenement_id`
- `Klant` → `email_adres`

Een primary key moet:

- Uniek zijn
- Nooit leeg zijn
- Vast blijven (mag niet wijzigen)

Als je later tabellen met elkaar verbindt, gebruik je de PK om een verbinding te maken van de ene entiteit naar de andere entiteit.

Voorbeeld van een ERD met PK

image.png
image not found or type unknown

Opdracht 1 – Kies je primary keys

1. Gebruik het ERD van de vorige opdracht met de entiteiten: `evenement`, `klant` en `locatie`.
2. Voeg aan elke entiteit een geschikte primary key toe. Kies daarbij een bestaand attribuut of voeg zelf een sleutel toe.
3. Zet in de eerste kolom van je ERD de letters PK om aan te geven dat deze regel de PK bevat.

4. Controleer: is jouw PK echt uniek en onveranderlijk?

☐☐ Reflectie

- Welke van je gekozen PK's is natuurlijk (bestaand gegeven), en welke kunstmatig (gegenereerd ID)?
- Wat zou er misgaan als je geen PK kiest?

☐☐ Inleveren

- Lever een screenshot van je ERD in met in de eerste kolom de PK's (primary keys).

4 1:N-relaties en Foreign Keys

☐☐ Leerdoelen

- Je weet wat een 1:N-relatie is in een database.
- Je begrijpt wat een foreign key (FK) is en waarvoor die dient.
- Je kunt zelf een 1:N-relatie modelleren met een foreign key.

☐☐ Uitleg

☐☐ Wat is een 1:N-relatie?

Bij een **één-op-veel-relatie** (1:N) hoort bij één rij in de ene tabel, meerdere rijen in de andere tabel.

Bijvoorbeeld:

- 1 Klant → **veel** Boeking(en)
- 1 Docent → **veel** Cursussen

De **“meer”-kant** krijgt de foreign key (FK). De FK is een kopie van de primary key (PK) van de andere tabel.

☐ De regel **Meer = harkje = FK**

- Teken het harkje aan de kant waar “veel” is
- In die tabel voeg je de foreign key toe
- Voorbeeld: `boekingen` krijgt `klant_id` als FK

☐ Voorbeeld

Een klant kan veel boekingen hebben (andersom kan niet!). Dus het harkje komt aan de kant van de boeking.

Bij elk harkje hoort een **FK** die verwijst naar de **PK** van de entiteit waarmee die is gekoppeld. In dit geval dus klantnummer.

image.png and or type unknown

☐ Het `klantnummer` in boeking is een *foreign key* die verwijst naar klant.

☐ Opdracht 1 – Relaties en foreign keys

1. Gebruik het ERD van de vorige opdracht met de entiteiten: `evenement`, `klant` en `locatie`.
2. Bepaal de veel kant van klant en evenement (evenement zou een concert kunnen zijn).
3. Bepaal de veel kant van evenement en locatie
4. Teken de entiteiten en de relaties. Zet de harkjes aan de juiste kant.
5. Voeg foreign keys toe (FK's) en plaats in de eerste kolom een FK bij elke Foreign key.

☐ Inleveren

- Lever je een screenshot van de ERD's inclusief foreign keys en erelaties in.

5 Datatypes en Validatie

☐ Doelen

- Je kent de meest gebruikte datatypes in een database.
- Je kunt passende datatypes kiezen voor attributen in je ERD.
- Je weet waarom validatie belangrijk is bij het kiezen van datatypes.

Uitleg

Wat is een datatype?

Een datatype bepaalt welk soort informatie je in een kolom/tabel opslaat. Het zorgt ervoor dat je database weet hoe de data eruitziet en wat ermee mag gebeuren.

Niet alle data-typen zijn even snel als je ze in een database gebruikt. Als je één van de volgende veelgebruikte datatypes gebruikt dan zit je meestal goed. Wil je andere gebruiken zoek dan goed uit wat de nadelen zijn.

Veelgebruikte datatypes:

Datatype	Gebruik	Voorbeeld
INT	Voor hele getallen (bijv. ID's, aantallen)	5, 142
VARCHAR(50)	Voor tekst tot X karakters	'Jan', 'email@example.com'
DATE	Voor datums zonder tijd	'2025-06-06'
DATETIME	Voor datum én tijd	'2025-06-06 14:30:00'
DECIMAL(6,2)	Voor getallen met komma (bijv. prijs)	12.99, 9999.00

Voorbeeld

image.png
image not found or type unknown

Tips voor goede keuzes:

- Let op lengte bij VARCHAR: hoe langer, hoe trager, maar te kort is ook niet goed.
- Een telefoonnummer is een VARCHAR() en geen INT, waarom?
- Stel je wil een bedrag van maximaal 9999,99 opslaan dan gebruik je float(6,2).

Opdracht 1 – Pas je datatypes aan

1. Gebruik het ERD van de vorige opdracht met de entiteiten: `evenement`, `klant` en `locatie`.
2. Voeg de volgende attributen toe:
 - `evenement` -> `toegangsprijs`, `aanvangt datum` en `tijd`
3. Kies voor elk attribuut een passend datatype uit de tabel hierboven.
4. Schrijf de datatypes in de derde kolom van de ERD's..
5. Gebruik minimaal 3 verschillende soorten datatypes; je mag er zelf attributen bij bedenken.

Inleveren

- Lever je bijgewerkte ERD in met daarin duidelijk per attribuut het datatype genoteerd en de FK.

6 Case – Modelleer een realistisch scenario

Leerdoelen

- Je kunt zelfstandig entiteiten en relaties herkennen in een realistisch scenario.
- Je past de 5 stappen toe om een ERD te maken.
- Je maakt een logisch en technisch correct ERD met PK's, FK's en datatypes.

Herhaling, de 5 basisregels

1. Een **entiteit** is een persoon, ding of gebeurtenis. Een getal of bedrag (bijvoorbeeld gewicht) is nooit een entiteit, maar altijd een attribuut (=eigenschap) van een entiteit.
2. Elke entiteit heeft precies één **PK (primary key)**. De primary key maakt de entiteit uniek (bijvoorbeeld kenteken van een auto).
3. Entiteiten hebben de volgende **relaties** 1:1, 1:N, N:1 of N:M.

- 1:1 relaties bestaan bijna niet, als ze voorkomen dan kun je de relaties samenvoegen.
 - 1:N en N:1 is eigenlijk hetzelfde en komen het meest voor.
4. Een **1:N** relatie verbind je met een lijntje met een 'harkje'. Het **lijntje** staat aan de 1-kant en het '**harkje**' staat aan de meer-kant.
 5. Bij elk 'harkje' hoort precies één FK. De FK verwijst naar de PK van de table waarmee deze is verbonden.

☐☐ Uitleg

Tot nu toe heb je geleerd wat entiteiten, attributen, PK's, FK's, datatypes en 1:N-relaties zijn. Nu pas je alles toe op een echte situatie.

☐☐ Scenario: Fietsenmaker Snelle Jelle

Fietsenmaker Snelle Jelle wil na een **reparatiebeurt** zijn klanten per SMS of Whatsapp op de hoogte stellen dat de reparatie klaar is. In dit bericht wil hij ook vertellen hoe hoog de reparatiekosten zijn.

Omdat de veel **klanten** meer dan één **fiets** hebben, wil hij van de fietsen ook wat kenmerken vastleggen. Hij wil het merk, model, type en kleur kunnen vastleggen.

Van elke reparatiebeurt wil hij verder vastleggen wanneer het onderhoud plaatsvond, hoe lang de reparatie duurde, wat er is uitgevoerd en de prijs.

☐☐ Je hebt de volgende entiteiten

- **reparatie:**
- **klant:**
- **fiets:**

☐☐ Opdracht – Maak het ERD

1. Bepaal per entiteit eerst alle attributen, lees daarvoor goed het scenario door!
2. Teken de drie entiteiten in Lucichart
3. Zet alle attributen in de entiteiten

4. Bepaal de Primary Key (PK).
5. Bepaal de data-types.
6. Bedenk wat de relaties zijn en teken die met het hartje aan de goede kant.
7. Bepaal de Foreign Key (FK)

□□ Inleveren

- Lever een screenshot in van je ERD gemaakt in Lucichart. Zorg dat alles goed leesbaar is.

7 *Meerdere relaties en N:N*

□□ Leerdoelen

- Je begrijpt wat een N:N-relatie is en wanneer die voorkomt.
- Je kunt een N:N-relatie correct omzetten naar aparte tabellen met FK's.
- Je kunt meerdere relaties per entiteit modelleren in een ERD.

□□ Uitleg


□□ Wat is een N:N-relatie?

Een **veel-op-veel** (N:N) relatie komt voor wanneer meerdere records uit entiteit A gekoppeld kunnen zijn aan meerdere records uit entiteit B.

Voorbeeld: Studenten kunnen zich inschrijven voor meerdere vakken. Elk vak kan meerdere studenten hebben.

□□ Hoe modelleer je dit?

Je maakt een extra tabel tussen de twee entiteiten. Deze bevat alleen de foreign keys van beide kanten.

 and or type unknown

De tussentabel (ook koppeltabel genoemd) bevat meestal ook extra informatie, zoals de inschrijfdatum of het cijfer. Deze extra informatie gaat over de combinatie student-vak. Een student heeft geen cijfer, een vak heeft geen cijfer, maar de combinatie student-vak heeft wel een cijfer.

☐☐ Meerdere relaties op één entiteit?

Soms is een entiteit aan meerdere andere entiteiten verbonden.

Voorbeeld: Een **medewerker** werkt in een afdeling, maar ook aan meerdere projecten.

- 1 medewerker ↔ 1 afdeling → 1:N
- 1 medewerker ↔ meerdere projecten ↔ N:N

Gebruik verschillende relaties als het logisch is dat een entiteit meerdere rollen vervult.

☐☐ Opdracht – N:N-model

1. Maak een ERD voor dit scenario:
Een muziekschool organiseert **lessen**. **Leerlingen** kunnen zich inschrijven op meerdere lessen. Elke **les** wordt gevolgd door meerdere leerlingen.
2. Bedenk zelf voor elke entiteit minimaal 4 attributen.
3. Modelleer de juiste entiteiten, attributen, PK's, FK's en datatypes.
4. Voeg een tussentabel toe om de N:N-relatie correct te verwerken.

☐☐ Reflectie

- Waarom is een tussentabel nodig bij N:N?
- Welke extra informatie kun je in de tussentabel kwijt?
- Waar moet je op letten bij het toevoegen van meerdere relaties in je ERD?

☐☐ Inleveren

- Lever een screenshot in van je ERD gemaakt in Lucichart. Zorg dat alles goed leesbaar is.
- Beantwoord de reflectievragen (pdf of txt bestand)

8 Bibliotheek

□□ Scenario: Bibliotheek

In een bibliotheek wil men bijhouden welke klanten welk boek van welke periode tot periode hebben geleend.

Elke klant kan meerdere boeken gelijktijdig lenen. Verder wil men de klant een whatsapp kunnen sturen twee dagen voor het verstrijken van de inleverdatum.

- Maak een databaseontwerp (ERD).

□□ Inleveren

- Lever een screenshot in van je ERD gemaakt in Lucichart. Zorg dat alles goed leesbaar is.

--

JS - (DOM)

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 achtergrondkleur 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. ``) 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.

7 *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 verversed dan. 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 + "!";  
});
```

```
</script>
```

☐☐ 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.

8 Eindopdracht – Interactieve DOM-app

Leerdoelen

- Je past alle basisvaardigheden toe rondom DOM-manipulatie.
- Je werkt met invoervelden, lijsten, events en classList.
- Je maakt een kleine interactieve webapp met JavaScript.

Uitleg

In deze les ontwerp je zelf een kleine interactieve DOM-app. Kies zelf of je een **todo-lijst**, een **stemtool**, een **chatbox**, of een **quiz** maakt. Je past de technieken toe uit lessen 1 t/m 7.

Mogelijke voorbeelden:

- **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.

Gebruik minimaal deze onderdelen:

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

Opdracht – Kies en bouw jouw mini-app

1. Maak een bestand `dom8.html`.

2. Bedenk welk klein DOM-projectje je maakt (todo, quiz, poll of iets anders).
3. Werk stap voor stap: begin met HTML, voeg daarna JavaScript toe.
4. Gebruik de technieken uit de vorige lessen.
5. Stijl het geheel met CSS zodat het overzichtelijk is.

☐☐ Reflectie

- Wat vond je het makkelijkst om te doen in dit project? En het lastigst?
- Welke technieken heb je gebruikt? Noem er minstens drie.
- Wat zou je nog willen leren over JavaScript?

☐☐ Inleveren

- Lever je `dom8.html` bestand in, samen met je eigen CSS-bestand (indien apart).
- Lever je reflectie in als `.txt` of `.pdf`.
- Eventueel: voeg screenshots toe als je project moeilijk te testen is.

JS - (DOM) - Extra

▣ Gegevens bewaren met *localStorage*

▣ Doelstellingen

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

▣ Uitleg

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

Je gebruikt het bijvoorbeeld zo:

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

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

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

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

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

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

Opdracht – Opslaan wat je invult

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

Tips:

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

Reflectie

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

Inleveren

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

Gegevens ophalen met `fetch()`

Doelstellingen

- 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 ``.
5. Bonus: laat de gegevens pas zien als je op een knop “Laad gebruikers” hebt geklikt.

Extra uitdaging:

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

☐☐ Reflectie

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

☐☐ Inleveren

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

☐☐ *Lijsten filteren op basis van invoer*

☐☐ Leerdoelen

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

☐☐ Uitleg

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

Voorbeeld – een zoekveld dat een lijst filtert:

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

```
<ul id="dierenlijst">
```

```
<li>Hond</li>
```

```
<li>Kat</li>
```

```
<li>Papegaai</li>
```

```
<li>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 – Interactieve Boekenkast

🎯 Doelstellingen

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

📖 Uitleg

Je werkt bij een uitgeverij. Ze willen een interactieve webpagina waarop bezoekers hun favoriete boeken kunnen opslaan, bekijken en beheren. Jij gaat een kleine webapp bouwen waarin gebruikers boeken kunnen toevoegen, bekijken, afvinken als gelezen, verwijderen en zoeken.

Mogelijkheden van je webapp:

- Boeken toevoegen (titel + schrijver)
- Boeken afvinken als gelezen (class toggle)
- Boeken verwijderen (event.target.remove())
- Boeken zoeken (live filter)
- Boeken opslaan in `localStorage` (blijft bewaard bij herladen)

📖 Opdracht – Bouw je eigen boekenkast

1. Maak een nieuw HTML-bestand met de naam `boekenkast.html`.
2. Voeg een invoerformulier toe met velden voor **titel** en **schrijver**, en een knop “Voeg toe”.
3. Laat de boeken verschijnen in een lijst. Elke rij bevat:
 - De titel en schrijver
 - Een knop “✓ gelezen” (toggle class `gelezen`)
 - Een knop “✕ verwijder”
4. Gebruik `localStorage` om de lijst te bewaren en opnieuw te laden bij het openen van de pagina.
5. Voeg bovenaan een zoekveld toe waarmee de gebruiker live kan filteren op titel of schrijver.

Bonus: voeg een knop toe om alles te wissen (met `localStorage.clear()`).

📅 Planning maken

Maak vóórdat je begint een planning. Zet hierin:

- Welke onderdelen je gaat bouwen (bijv. HTML-formulier, toevoegen, verwijderen...)
- In welke volgorde je daaraan werkt
- Een schatting van hoeveel tijd je per onderdeel nodig hebt

Voorbeeld

Onderdeel	Tijd	Status
----- ----- -----		
HTML structuur	20m	
JS: boeken toevoegen	30m	
....	
.....	
etc. etc.		

📅 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 `boekenkast.html` in (zet de `<style>` sectie en je code de `<code>` sectie).
- Lever je **planning** in (.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.
- Lever je **reflectie** in als .txt of .pdf.
- Bonus: voeg een screenshot toe van je werkende webapp.