

Blok 4 - PHP

- [PHP Intro](#)
- [Prompt Engineering 1](#)
- [PHP-1](#)
- [PHP Challenge](#)

PHP Intro

1 Front-End en Back-End

Leerdoelen

- Je weet wat frontend en backend zijn.
- Je kunt het verschil uitleggen in je eigen woorden.
- Je kunt bij een website aanwijzen wat frontend is en wat backend is.

Wat is frontend?

- **De voorkant** van een website: alles wat de gebruiker ziet.
- Wordt gemaakt met HTML, CSS, JavaScript.
- Denk aan knoppen, kleuren, tekst, afbeeldingen.

☐ Vergelijking: de menukaart en bediening van een restaurant.

Wat is backend?

- **De achterkant** van een website: alles wat op de server gebeurt.
- Zorgt voor berekeningen, het opslaan van gegevens, inloggen, bestellingen verwerken.
- Wordt vaak gemaakt met talen zoals PHP, Python, Node.js.

☐ Vergelijking: de keuken van een restaurant.

Client en server

De computer is geen restaurant, maar je kunt het er wel mee vergelijken: de webbrowser waar de bestellingen worden opgenomen en de webserver (ergens in de cloud) is de keuken.

image-1655279215130.png

Bestandsoorten

We kennen al HTML en CSS, later gaan we ons verdiepen in Javascript en PHP.

Bestandssoorten	Functie	Waar
HTML	Basis opmaak van een webpagina	Front-end / browser
CSS	Detail opmaak van een webpagina	Front-end / browser
JavaScript	Interactie programmeren in de browser	Front-end / browser
PHP	Interactie programmeren op de server	Back-end / server

Opdracht 1

Geef van elk van de onderstaande situatie wat het is:

front-end, back-end of beiden?

Licht elk antwoord in eigen woorden toe.

	Situatie	Front-end, back-end, beiden	Toelichtng
1	Je klikt op een knop om een filmpje te starten.		
2	Je stuurt een contactformulier in en krijgt "Bedankt voor je bericht!".		
3	Een webwinkel controleert of er nog voorraad is.		
4	Je ziet de kleuren en lettertypes van een website veranderen.		
5	Een systeem herkent of je het juiste wachtwoord hebt ingevuld.		
6	Je krijgt je resultaten na een online quiz.		

Opdracht 2

Beantwoord de volgende vragen in eigen woorden

- Wat wist je nog niet?
- Waarom denk je dat er **zowel frontend als backend** nodig is?
- Wat lijkt jou leuker om te maken?

Inleveren

Open het Word document [Opdracht Front-end backend.docx](#) en beantwoord de vragen uit opdracht 1 en opdracht 2. Bewaar het bestand als PDF en lever dat in.

- PDF document

2 Installeren XAMPP

Wij hebben (nog) geen webserver in de cloud. In plaats daarvan gaan we een soort 'nep-server' gebruiken en hiervoor gaan we XAMPP installeren.

Download de software

Google of download de juiste versie van [apachefriends](#)

Voer de installatie uit

Je hebt alleen **Apache** en **MySQL** nodig, de andere opties kan je beter niet installeren.

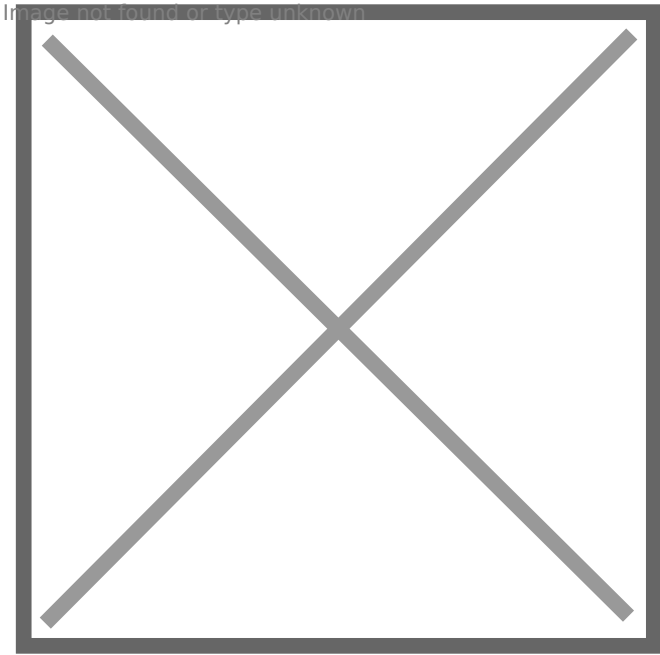
Als er wordt gevraagd om *php.exe aan het path toe te voegen* dan kies je ja/yes. Alle andere opties kun je laten staan zoals ze *default* staan.

Opstarten XAMPP

Als het goed is, is er tijdens de installatie een short-cut gemaakt die heet *XAMPP Control*.

Deze shortcut voert `C:\xampp\xampp-control.exe` uit.

Druk op de eerste twee knopjes start naast Apache en MySQL. Hiermee start je de webserver en de database op. Als alles goed is dan zie je het volgende.



XAMPP Instellingen

We gaan een paar dingen anders instellen.

Allereerst gaan we de database als Windows-service laten draaien dat verhoogt de stabiliteit.

1 Stop XAMPP

Stop XAMPP en als dat niet lukt, restart je machine of gebruik de Windows task manager om XAMPP te stoppen.

2 Properties xampp-controll.exe

Ga via de file explorer naar c:\xampp\ en zoek naar xampp-controll.exe, selecteer met rechtermuis propertjes (of eigenschappen).

3 Compatibily 'Run this program as an administrator'

Selecteer onder het tabje Compatibily 'Run this program as an administrator'

4 OK en klaar.

[image-1671310391146.png](#)

5 Start XAMPP

Sluit XAMPP en start XAMPP opnieuw op.

6 Groen vinkje

Druk op het rode kruisje links naast MySQL. Het kruisje wordt een groen vinkje.

image-1687096321945.png

7 index.php weg

Tenslotte, ga naar c:\xampp\htdocs\ en gooi het bestand index.html weg.

Waarom dat is dat zien we later als we met XAMPP gaan werken.

Opdracht

Voer alle 7 stappen uit.

Inleveren

1. Screenshot van XAMPP Control panel met (minimaal) één groen vinkje en Apache en MySQL gestart.
Er zijn geen rood gekleurde foutmeldingen te zien.

3 Installatie Visual Studio Code

In deze les leer je hoe je Visual Studio Code installeert, waarom we een code editor gebruiken, en hoe je PHP-bestanden opent vanuit de `htdocs`-map.

☐☐ Wat gaan we doen?

- We installeren Visual Studio Code (VSC)
- We leggen uit waarom we een goede editor nodig hebben
- We openen een PHP-bestand in de `htdocs`-map

☐☐ Waarom Visual Studio Code?

- VSC is een gratis en krachtige code editor

- Hij maakt je code overzichtelijk met kleuren, inspringing en suggesties
- Hij helpt je fouten sneller te vinden en biedt slimme aanvulling
- Het is de standaard editor voor veel professionele programmeurs

Zonder goede editor werk je in Kladblok of iets simpels – dat is lastig lezen, foutgevoelig en traag werken.

☐☐ Installatie

1. Ga naar de officiële website: <https://code.visualstudio.com/>
2. Klik op **Download for Windows**
3. Start het installatieprogramma en kies de standaardinstellingen (volgende, volgende...)
4. Vink aan: "Add to PATH" en "Open with Code" in *contextmenu* als dat wordt gevraagd

☐☐ Eerste keer openen

Open Visual Studio Code. Je ziet een startscherm. Je hoeft je nergens aan te melden.

☐☐ Openen van een PHP-bestand

1. Ga in VSC naar **File > Open Folder**
2. Navigeer naar de map `htdocs` op jouw computer (bijv. `C:\xampp\htdocs`)
3. Klik op **Select Folder**
4. In de linkerbalk maak je nu een nieuw bestand en die noem je `test.php`
5. Klik op een bestand om het te openen en te bewerken

Tip: Klik met rechts op het bestand en kies "Reveal in File Explorer" als je het ook in Verkenner wil zien.

☐☐ Extra VSC Tips

- Installeer de **PHP Intelephense**-extensie voor slimme PHP-hulp

- Gebruik **Ctrl + S** om op te slaan
- Als je veel bestanden hebt, gebruik dan **Ctrl + P** om snel te zoeken

Opdracht

Zet dit in het bestand `test.php`

```
<?php  
  
echo "TEST";
```

Sla op en ga nu met de browser naar <https://localhost/test.php>

Wat zie je?

☐☐ Reflectie

Niet inleveren maar probeer dit voor je zelf te beantwoorden als voorbereiding op de Kennis-Check.

- Waarom gebruiken we Visual Studio Code en niet Kladblok?
- Wat heb je geleerd over het openen van PHP-bestanden?

☐☐ Inleveren

Maak een screenshot van:

1. Visual Studio Code met een geopend bestand uit `htdocs`

4 *Jouw eerste PHP-pagina*

In deze les ga je jouw allereerste PHP-pagina maken. Je leert hoe je PHP-code schrijft en uitvoert, en hoe je PHP en HTML samen kunt gebruiken.

☐☐ Wat gaan we doen?

- Je maakt een PHP-bestand aan in de map `htdocs`
- Je schrijft je eerste `echo`-opdracht in PHP
- Je bekijkt het resultaat via `http://localhost`

i Uitleg

- **PHP** is een programmeertaal voor de backend
- PHP draait op de server en geeft HTML terug aan de browser
- Een PHP-bestand eindigt op `.php`
- Met `echo` toon je iets op het scherm

Voorbeeld 1: eenvoudige tekst

```
<?php
echo "Hallo wereld!";
?>
```

Toont de tekst "Hallo wereld!" in je browser.

Voorbeeld 2: HTML + PHP

```
<!DOCTYPE html>
<html>
<body>
  <h1>Welkom</h1>
  <p><?php echo "Dit is gegenereerd door PHP!"; ?></p>
</body>
</html>
```

Hierin zie je dat je PHP kunt combineren met gewone HTML.

Opdracht 3A

Maak een bestand `intro.php` in de map `htdocs` en zet daarin:

```
<html>
<body>
```

```
<h2>Over mij</h2>
<p>
  <?php
    echo "Hallo! Ik ben [jouw naam] en dit is mijn eerste PHP-pagina.";
  ?>
</p>
</body>
</html>
```

- Vervang [jouw naam] door je eigen naam
- Open in de browser via: `http://localhost/intro.php`

Opdracht 3B – Extra

Laat meer zinnen zien met meerdere `echo`-regels en gebruik `
` om af te breken:

```
<?php
echo "Ik hou van programmeren.<br>";
echo "Mijn favoriete kleur is blauw.<br>";
echo "PHP is best leuk!";
?>
```

Tip: combineer dit in een klein "dagboekje over jezelf".

Afsluiting

- Wat gebeurt er als je een PHP-bestand opent door erop te dubbelklikken in Verkenner?
- Stel je maakt een folder in htdocs en zet in deze folder een .php bestand, hoe kan je dat dan via localhost in de browser openen?

Inleveren

1. Screenshot van de pagina in je browser met jouw naam zichtbaar

5 Variabelen en Strings in PHP

In deze les leer je hoe je gegevens kunt opslaan in variabelen, hoe je met tekst (strings) werkt, en hoe je deze informatie kunt tonen op je website met PHP.

☐☐ Wat gaan we doen?

- We leren wat een variabele is
- We maken een paar variabelen aan in PHP
- We tonen informatie op het scherm met `echo`
- We combineren tekst met variabelen (string concatenatie)

☐☐ Wat is een variabele?

- Een variabele is een soort doosje waarin je informatie kunt stoppen
- Je geeft het doosje een naam, zoals `$naam` of `$leeftijd`
- Je kunt de waarde later gebruiken of aanpassen

Voorbeeld

```
<?php
$naam = "Fatima";
$leeftijd = 16;
echo "Hallo, mijn naam is " . $naam . " en ik ben " . $leeftijd . " jaar oud.";
?>
```

Resultaat: Hallo, mijn naam is Fatima en ik ben 16 jaar oud.

☐☐ Opdracht – Toon jouw info

Maak een bestand `opdracht5.php` in je `htdocs`-map.

Vul dit in met jouw eigen gegevens:

```
<?php
$naam = "Jouw naam hier";
$leeftijd = 15;
$school = "ROC Amstelland";
```

```
echo "<h1>Over mij</h1>";  
echo "<p>Ik ben " . $naam . " en ik ben " . $leeftijd . " jaar oud.</p>";  
echo "<p>Ik zit op " . $school . ".</p>";  
?>
```

- Vervang de waarden door je eigen gegevens
- Open het bestand via `http://localhost/opdracht5.php`

Opdracht – Extra uitdaging

- Voeg een variabele toe voor je favoriete hobby
- Toon dit op een derde regel met:

```
Mijn hobby is muziek maken.
```

i Uitleg

- In PHP begint een variabele altijd met een `$`-teken
- Tekst noemen we een **string** (tussen aanhalingstekens)
- Als je tekst en variabelen combineert, gebruik je `.` (punt) om aan elkaar te plakken

Reflectie

- Wat is het verschil tussen een tekst en een getal in PHP?
- Waarom is het handig om gegevens op te slaan in variabelen?
- Kun je bedenken hoe je variabelen later zou gebruiken in een formulier of website?

Inleveren

1. Lever je het bestand `opdracht.php5` in.
Je tekst moet minimaal 3 variabelen gebruiken

6 Strings en Getallen in PHP

In deze les leer je het verschil tussen tekst (strings) en getallen (integers) in PHP. Je leert ook wat er gebeurt als je probeert te rekenen met tekst of tekst toevoegt aan een getal.

☐☐ Wat gaan we doen?

- We leggen het verschil uit tussen strings en getallen
- We oefenen met optellen en tekst plakken
- We maken een mini-rekenmachine

☐☐ Wat is het verschil?

- **String** = tekst, tussen aanhalingstekens: "Hallo"
- **Getal** = nummer, zonder aanhalingstekens: 5
- Als je twee strings met `.` (punt) combineert, dan “plak” je tekst aan elkaar
- Als je twee getallen met `+` optelt, dan krijg je de som

Voorbeeld

```
<?php
$tekst1 = "Hallo";
$tekst2 = "wereld";
$getal1 = 10;
$getal2 = 5;

echo $tekst1 . " " . $tekst2; // Hallo wereld
echo "<br>";
echo $getal1 + $getal2;      // 15
?>
```

☐☐ Opdracht – Reken met getallen

Maak een nieuw bestand aan in je `htdocs`-map: `rekenen.php`

Schrijf een script dat dit doet:

- Maakt twee variabelen met getallen
- Toont het resultaat van **optellen**, **aftrekken**, **vermenigvuldigen** en **delen**.

Voorbeeldcode

```
<?php
$prijs = 7.50;
$aantal = 3;
$totaal = $prijs * $aantal;

echo "<p>Prijs per stuk: €" . $prijs . "</p>";
echo "<p>Aantal: " . $aantal . "</p>";
echo "<p>Totaalprijs: €" . $totaal . "</p>";
?>
```

☐☐ Let op met aanhalingstekens!

Als je een getal tussen aanhalingstekens zet, wordt het een **string** en kun je er niet goed mee rekenen:

```
$getal1 = "10"; // string
$getal2 = 5; // integer
echo $getal1 + $getal2; // dit werkt, maar PHP zet de string stilletjes om naar een getal
```

Tip: Probeer getallen **zonder aanhalingstekens** te houden als je ermee rekent!

☐☐ Reflectie

- Wat gebeurt er als je probeert te rekenen met een string zoals "hallo"?
- Waarom is het handig dat PHP automatisch strings soms omzet naar getallen?
- Wat zou er gebeuren als je \$prijs een tekst maakt in plaats van een getal?

☐☐ Inleveren

1. Lever een screenshot in van jouw `rekenen.php` in de browser
Je code moet verschillende berekeningen laten zien.

7 Leeftijd check met formulier

In deze les leer je hoe je een formulier maakt dat gegevens doorstuurt naar een PHP-script, en hoe je met if-statements reageert op die gegevens.

Leerdoelen

- Je maakt een formulier dat gegevens verzendt naar een PHP-bestand
- Je kunt een `if` gebruiken om iets te controleren
- Je reageert in de browser op wat de gebruiker heeft ingevuld

Formulier maken

Maak een bestand `formulier.html` aan in je `htdocs`-map. Zet hierin de volgende code:

```
<!DOCTYPE html>
<html>
<body>

  <h2>Hoe oud ben jij?</h2>

  <form action="leeftijd.php" method="get">
    <label for="leeftijd">Voer je leeftijd in:</label><br>
    <input type="number" name="leeftijd" id="leeftijd"><br><br>
    <input type="submit" value="Verstuur">
  </form>

</body>
</html>
```

Uitleg

Dit is een eenvoudig HTML-formulier dat de leeftijd vraag en die deze waarde dan opstuurd naar het bestand leeftijd.php.

□□ leeftijd.php

Maak nu een bestand `leeftijd.php` en voeg dit toe:

```
<?php
$leeftijd = $_POST["leeftijd"];

echo "<h2>Jouw leeftijd is: " . $leeftijd . "</h2>";

if ($leeftijd >= 18) {
    echo "Je bent volwassen.";
} else {
    echo "Je bent nog geen 18.";
}
?>
```

Uitleg

Op regel 2 wordt de \$leeftijd uit de formulier variable gehaald.

Regel 6 t/m 10 laten zien hoe je een conditiemet een **if-then-else** in PHP maakt.

□□ Opdracht – Zelf uitbreiden

- Voeg een extra if-statement toe die controleert of iemand jonger is dan 12:

```
if ($leeftijd < 12) {
    echo "<br>Je bent nog een kind.";
}
```

- Voeg ook een if-statement toe die een compliment geeft bij een mooie leeftijd (bijvoorbeeld 16 of 21)
- **Bedenk zelf een leuke uitbreiding** of verander het programma. De code moet wel:
 - minimaal **3 if-statements** bevatten en minimaal **1 if-then-else**.
 - **Test je code**

Tip

- Gebruik `$_GET["veldnaam"]` om iets op te halen uit een formulier
- Gebruik `if`, `else` of `elseif` om verschillende reacties te geven

Reflectie

- Kun je bedenken hoe je dit formulier zou kunnen gebruiken voor een login of een quiz?

Inleveren

1. Lever een screenshot in van `leeftijd.php` met een ingevuld formulier en meerdere testleeftijden
Je code moet minstens 3 `if`-statements bevatten waarvan minimaal 1 `if-then-else`.

8 Kortingscalculator met formulier

In deze les maak je een eigen kortingscalculator. Je vult een bedrag en een kortingspercentage in, en de PHP-code berekent de prijs na korting.

Leerdoelen

- Je gebruikt een formulier met meerdere invoervelden
- Je rekt in PHP met getallen die uit een formulier komen
- Je maakt zelf een praktische webpagina

Stap 1 – Pas je formulier aan

Pas het formulier uit `formulier.html` aan zodat het er zo uitziet:

```
<!DOCTYPE html>
<html>
<body>
```

```
<h2>Kortingscalculator</h2>

<form action="korting.php" method="get">
  <label for="bedrag">Voer het bedrag in:</label><br>
  <input type="number" name="bedrag" step="0.01" required><br><br>

  <label for="korting">Korting (%):</label><br>
  <input type="number" name="korting" step="1" required><br><br>

  <input type="submit" value="Bereken korting">
</form>

</body>
</html>
```

☐☐ Stap 2 – Maak korting.php

Maak een nieuw bestand aan in je `htdocs`-map met de naam `korting.php` en plak daarin de volgende code:

```
<?php
$bedrag = $_POST["bedrag"];
$korting = $_POST["korting"];

$kortingBedrag = $bedrag * ($korting / 100);
$nieuwBedrag = $bedrag - $kortingBedrag;

echo "<h2>Resultaat</h2>";
echo "<p>Oorspronkelijk bedrag: €" . $bedrag . "</p>";
echo "<p>Korting: " . $korting . "%</p>";
echo "<p>Korting in euro: €" . round($kortingBedrag, 2) . "</p>";
echo "<p>Bedrag na korting: €" . round($nieuwBedrag, 2) . "</p>";
?>
```

☐☐ Uitleg berekening

- Voorbeeld: je voert in €100 en 20% korting

- PHP rekent uit: `100 * (20 / 100) = 20`
- Nieuwe prijs = `100 - 20 = 80`

Opdracht 1

- Laat een extra bericht zien met `if`:
 - Bij een korting van 50% of meer: `"Wat een superdeal!"`
 - Bij minder dan 10%: `"Dat is maar een klein beetje korting."`

Opdracht 2

Maak een .txt. of PDF en geef antwoord op de volgende vragen.

Vraag 1

In het formulier staat: `<input type="number" name="bedrag" step="0.01" required>

`
Wat doet `step="0.01"`; wat gebeurt er als je `step="1"` gebruikt?

Vraag 2

In het formulier staat: `<input type="number" name="bedrag" step="0.01" required>

`
Wat doet `required`; wat gebeurt er als je dit weglaat?

Vraag 3

In het formulier staat: `<input type="number" name="bedrag" step="0.01" required>

`
Verander `name="bedrag"` naar `name="prijs"`. Test of de berekening nog werkt.

Leg uit wat je moet aanpassen in de code `korting.php` om de berekening weer te laten werken na deze aanpassingen.

Reflectie

- Wat gebeurt er als je 0% korting invoert?
- En als je 100% korting invoert?
- Zou je dit op een echte webshop kunnen gebruiken?

Inleveren

1. Het php bestand korting.php
2. PDF of txt-bestand met de antwoorden op de drie vragen in eigen woorden.

Prompt Engineering 1

Introductie

Prompt Engineering is alsof je een buitenstaander die niets van je weet iets wilt vragen. Omdat te doen moet je dus duidelijk en goed communiceren.

image.png
image not found or type unknown

We gaan leren hoe we goede prompt kunnen schrijven.

Voor een **goede prompt** moet je rekening houden met de volgende zaken:

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

Al deze punten worden stuk-voor-stuk behandeld. De eerste drie zijn **rood** omdat *élke* prompt altijd moet voldoen aan deze kenmerken.

Maar laten we eerst even kijken wat nu eigenlijk *prompting* precies is.

Wat is prompting?

Prompting betekent dat je iets vraagt aan een AI, zoals ChatGPT, op een manier waardoor je een goed antwoord krijgt. Je *geeft een opdracht of stelt een vraag*, en dat noemen we een **prompt**.

Je kunt het vergelijken met hoe je een klasgenoot iets vraagt:

1. Als je gewoon zegt: "Doe dit," snapt die ander misschien niet precies wat je bedoelt.

2. Maar als je zegt: "Kun je me helpen met deze code, want ik snap het niet?", is dat al een stukje beter.
3. Maar het kan nog beter: "Kun je me helpen met deze code want ik krijg een foutmelding 'unknown variable on line 23', ik snap niet wat de foutmelding betekent? "".
4. En zelfs dit kan nog beter want je vergeet een belangrijk detail, zie jij welke detail?

Kort gezegd: prompting is het slim stellen van een vraag of opdracht aan AI, zodat je krijgt wat je nodig hebt.

In de laatste zin staat niets over in welke omgeving en met welke programmeertaal je werkt. Ook zou je kunnen overwegen om de code of een stuk daarvan mee te sturen.

Opdracht 1

Verbeter de prompt die hierboven bij punt 4 staat.

1. Context

We hebben geleerd dat een goede prompt detail en context heeft.

Context is een soort van omgeving.

Als je vraagt om code aan passen dan moet je dus aangeven wat de 'omgeving' is, in dit geval PHP en als het je bepaalde frameworks gebruikt of andere zaken die van belang zijn dan noem je die ook.

- **Context** betekent: de omgeving waarin iets zich afspeelt.
- Bij programmeren is dat bijvoorbeeld: de taal (zoals PHP), het framework (zoals Laravel), of iets anders dat belangrijk is om te weten.
- Een AI snapt je vraag beter als jij eerst vertelt wat de **omgeving/context** is.

Opdracht 2

Situatie

Je hebt de volgende PHP-code gekregen.

Deze code toont de huidige datum:

```
<?php  
echo date("Y-m-d");  
?>
```

Maar jij wil dat de datum verschijnt in het **Nederlands**, zoals: 9 mei 2025

De AI moet je helpen om de code aan te passen.

Jouw taak:

1. **Schrijf een prompt aan ChatGPT** waarin je vraagt om deze code aan te passen zodat de datum in het Nederlands verschijnt.
2. Let erop dat je **duidelijk aangeeft wat de context is**:
 - Je werkt met **PHP**
 - De code moet draaien op een gewone server (geen framework zoals Laravel)
 - Je wil de **datum in het Nederlands**

Inleveren

1. Screenshot van de prompt en het antwoord dat je hebt gekregen.
2. Screenshot van het resultaat in je browser.

2. Detail/Specifiek

Doel

Je leert hoe belangrijk **detail en specificiteit** zijn in een goede prompt. Hoe specifieker je bent, hoe beter de AI je kan helpen met het schrijven of verbeteren van PHP-code.

Theorie

- Een vage prompt zoals “*Maak een contactformulier*” geeft een vaag resultaat.

- Welke velden moeten erin?
 - Wat moet er met de ingevulde gegevens gebeuren?
 - Moet het formulier controle uitvoeren?
 - Moet er een succesbericht komen?
- Een **specifieke prompt** vertelt precies wat je wil.

Een goede prompt bevat concrete dingen en bevat geen (of zo min mogelijk) 'vage' [termen](#).

Test je prompt

Als je over een prompt vragen kan stellen dan is die niet 100% concreet.

Voorbeeld prompt:

Maak de banner iets breder en verander het letter type zodat het groter is.

Wat is 'iets breder' en wat is 'groter'?

Beter zou zijn:

Maak de banner 5% breder en maak het font 2px groter.

Opdracht 3

Situatie

Je wil een **contactformulier in PHP** laten maken door ChatGPT.

Jouw taak

1. **Schrijf een prompt aan ChatGPT** waarin je duidelijk vraagt om een contactformulier in PHP.

Je moet in je prompt minimaal deze 4 dingen **specifiek vermelden**:

- Welke velden het formulier moet hebben (bv. naam, e-mail, bericht)
- Wat er met de data moet gebeuren (bijv. opslaan, versturen, tonen)
- Of er foutcontrole moet zijn (bijv. verplicht invullen, geldig e-mailadres)
- Of er een succesmelding moet komen na het verzenden

Inleveren

1. Screenshot van de prompt en het antwoord dat je hebt gekregen.
2. Screenshot van het resultaat in je browser.

3. Duidelijkheid

Duidelijk betekent dat je prompt voldoende detail bevat. Hierdoor is het duidelijk wat je precies bedoeld.

Een **duidelijke prompt** zegt precies:

1. *Wat* je wil
2. *Waarvoor* je het wil
3. Wat de beperkingen zijn
4. En eventueel *hoe* je het eruit wil laten zien

Een **voorbeeld** van de punten zijn:

1. Je wilt een website
2. Je wilt een website waarin de gebruiker zijn naam en adres moet opgeven.
3. Alle velden moeten op één web pagina passen en bestaat uit HTML en CSS.
De velden zijn allemaal verplicht. Zodra de submit button wordt ingedrukt wordt gecontroleerd of alle velden zijn ingevuld. Als dat niet het geval is dan volgt er een aanwijzing/waarschuwing.
4. Het moet er eenvoudig en professioneel uitzien, Gebruik 'Tailwind' als CSS framework.

Opdracht 4

Situatie

Je wil een AI vragen om een klein PHP-script te maken waarmee je een getal controleert: is het **even of oneven**?

Hieronder zie je een voorbeeld van een **vage prompt**. Lees hem goed.

“Schrijf iets in PHP dat met getallen werkt.”

Je gaat nu zelf een **duidelijke prompt** schrijven waarin je vraagt om een PHP-script dat:

- Één getal controleert
- Zegt of het getal even of oneven is
- De gebruiker het getal zelf laat invoeren via een HTML-formulier

Inleveren

1. Screenshot van de prompt en het antwoord dat je hebt gekregen.
2. Screenshot van het resultaat in je browser.

4. Doelgericht

Theorie

- AI is niet helderziend. Als je alleen zegt “Leg dit uit”, weet de AI niet hoe uitgebreid, voor wie of in welke vorm.
 - Wat wil je *precies* weten of krijgen?
 - Wil je code, uitleg, een stappenplan, een voorbeeld, een tabel, enz.?
 - Voor wie is het bedoeld? (Bijv. jezelf, een beginner, een klasgenoot?)Een **doelgerichte prompt** maakt duidelijk:

Opdracht

Situatie

Je wil begrijpen hoe een **while-loop** werkt in PHP, en je wil dat de AI het aan je uitlegt.

Deel 1: Vergelijk twee prompts

Prompt A (niet doelgericht):

“Wat is een while-loop?”

Prompt B (wel doelgericht):

“Leg uit wat een while-loop in PHP doet. Geef een voorbeeld met code en uitleg in simpele taal, zodat ik het kan gebruiken in een quiz voor klasgenoten.”

□ Wat is het verschil tussen A en B?

Deel 2: Schrijf je eigen doelgerichte prompt

Bedenk nu een onderwerp in PHP dat jij lastig vindt (bijvoorbeeld: arrays, forms, functies, POST vs GET...).

Schrijf een doelgerichte prompt waarbij je duidelijk maakt **wat je wil, in welke vorm, en voor wie het bedoeld is**

Inleveren

1. Deel 1, leg in je eigen woorden uit wat het verschil is tussen prompt A en prompt B.
2. Deel 2, schrijf je eigen doelgerichte prompt (zie beschrijving deel 2).

5. Vorm

□ Theorie (kort samengevat)

- Soms weet de AI wél wat je bedoelt, maar krijg je het antwoord in een **onhandige vorm**:
 - Alleen tekst terwijl jij voorbeeldcode wilde.
 - Een lang verhaal terwijl jij liever een stappenplan had.
 - Code zonder uitleg erbij.
- Daarom is het slim om in je prompt te zeggen **hoe je het antwoord wil zien**:
 - Als lijst, tabel, voorbeeldcode, uitleg in korte zinnen, enz.

Opdracht

Deel 1: Slechte prompt (voorbeeld)

“Leg uit hoe je een formulier maakt met PHP.”

Wat is hier onduidelijk over de **vorm** van het antwoord?

Deel 2: Schrijf zelf een betere prompt

Jij gaat nu een betere prompt schrijven waarin je **duidelijk zegt in welke vorm** je het antwoord wil krijgen. Kies een vorm, zoals:

- Stap-voor-stap uitleg
- PHP-code met uitleg onder elke regel
- Een tabel met onderdelen van een formulier
- Een combinatie van uitleg en voorbeeld

☐ Vergeet niet te vermelden dat je werkt met **PHP**, en dat het voor een **beginner** is.

Inleveren

....

6. Toon

☐ Theorie

- De **toon** is *hoe iets klinkt of overkomt*: serieus, grappig, formeel, simpel, kinderlijk, informeel, enz.
- Als je geen toon aangeeft, kiest de AI zelf. Soms is dat te zakelijk, te moeilijk of juist te speels.
- In een goede prompt zeg je dus **hoe de AI zich moet gedragen**.

Opdracht

Situatie:

Je wil dat ChatGPT uitlegt wat een **functie in PHP** is.

Je gaat dezelfde uitleg in **verschillende tonen** laten geven.

Stap 1: Schrijf drie verschillende prompts

Gebruik hetzelfde onderwerp ("Wat is een functie in PHP?") en verander **alleen de toon**.

1. **Formeel en technisch** (voor iemand met programmeerervaring):
2. **Grappig en speels** (alsof je het uitlegt aan een kind van 12):
3. **Duidelijk en vriendelijk** (voor een klasgenoot die net begint met PHP):

Inleveren

1. Welke toon vond jij het **meest duidelijk** voor een beginner?
2. Welke toon past het best bij jouw klasgroep?
3. Welke toon past het best bij jouw klasgroep?
4. Wat gebeurt er als je géén toon vraagt in je prompt?

Samengevat

Een goede prompt.

1. **Context** – Geef achtergrondinformatie, zoals *voor wie het is, wat het doel is, of waar het over moet gaan (welke programmeertaal?)*.
2. **Detail** – Hoe specifieker je bent, hoe beter het antwoord past bij wat je zoekt.
3. **Duidelijkheid** – Gebruik duidelijke en begrijpelijke taal. Vermijd vage of dubbelzinnige zinnen.

4. **Doelgerichtheid** – Geef aan *wat je precies wil* (bijvoorbeeld: een uitleg, een lijst, een verhaaltje, een vergelijking, enz.).
5. **Vorm** – Soms helpt het als je zegt *hoe* het antwoord eruit moet zien (bijvoorbeeld: “maak er een tabel van”, “gebruik korte zinnen”, “schrijf het op het niveau van een brugklasser”).
6. **Toon/Stem** – Wil je dat het grappig is? Serieus? Zakelijk? Kinderlijk? Dat kun je ook zeggen in je prompt.

Voorbeeld prompt

“Leg uit wat een if-statement is in PHP. Geef een simpel voorbeeld met uitleg in makkelijke taal. Ik ben 14 en net begonnen met PHP, dus graag zonder moeilijke woorden. Laat ook zien wat er gebeurt als de voorwaarde niet waar is.”

Opdracht 3

Maak een prompt waarin je aan AI vraagt om een programma in HTML-pagina te maken die er zo goed mogelijk lijkt op deze website template.

image.png
image not found or type unknown

Probeer dit in één tekst prompt te doen (gebruik dus géén plaatje) en denk aan alle 6 eigenschappen voor een goede prompt.

Inleveren

1. Prompt die je hebt gemaakt (schermafdruck).
2. Resultant (schermafdruck van de webpage).

PHP-1

1 Gegevens doorsturen met GET en POST

Leerdoelen

- Je weet wat `GET` en `POST` zijn.
- Je kunt gegevens doorsturen van de ene pagina naar de andere.
- Je kunt de gegevens gebruiken in een tweede PHP-bestand.

Uitleg

Bij een formulier kies je of je `GET` of `POST` gebruikt om de ingevulde gegevens naar de server te sturen:

Methode	Kenmerk	Voorbeeld
<code>GET</code>	Toont gegevens in de URL	<code>pagina.php?naam=Ali</code>
<code>POST</code>	Stuurt gegevens "verborgen" via de browser	Geen zichtbare URL-parameters

Opdracht 1 – formulier.html

Maak een bestand aan met de naam `formulier.html` en zet hier de volgende code in:

```
<!DOCTYPE html>
<html>
<body>

  <h2>Wat is je naam?</h2>
```

```
<form action="begroeting.php" method="get">
  <label for="naam">Naam:</label><br>
  <input type="text" id="naam" name="naam"><br><br>
  <input type="submit" value="Verstuur">
</form>

</body>
</html>
```

Check, `method="get"` op regel 7

Opdracht 2 – begroeting.php

Maak een nieuw bestand met de naam `begroeting.php` en zet hierin:

```
<?php
$naam = $_GET["naam"];
echo "<h1>Hallo $naam!</h1>";
?>
```

Check of alles werkt.

Als dat zo is verander dan regel 7 in het formulier naar:

```
<form action="begroeting2.php" method="post">
```

Pas nu de code in `begroeting.php` aan, zodat het formulier goed werkt.

Reflectie

- Wat is het verschil tussen POST en GET?
- Wat doet `action="begroeting2.php"` in het formulier?
- Wanneer zou je liever `POST` gebruiken dan `GET`? Leg uit!

Inleveren

- Beantwoord de drie vragen uit de reflectie en lever die in (.txt of .pdf bestand).

2 Include en Require

Leerdoelen

- Je weet wat `include` en `require` doen in PHP.
- Je kunt een header of footer inladen met `include`.
- Je herkent het verschil tussen `include` en `require`.

Uitleg

Vaak gebruik je op meerdere pagina's dezelfde stukjes HTML, zoals een menu of een footer. Je kunt dat opslaan in een apart bestand en invoegen met `include` of `require`.

- `include "bestand.php"`: probeert het bestand in te laden. Als dat niet lukt, gaat de pagina wel gewoon verder.
- `require "bestand.php"`: probeert het bestand in te laden. Als dat niet lukt, stopt de pagina met een fout.

Voorbeeld:

Stel, je maakt een bestand `header.php` met daarin een simpel menu:

```
<!-- header.php -->
<header>
  <h1>Mijn Website</h1>
  <nav>
    <a href="index.php">Home</a> |
    <a href="info.php">Info</a>
  </nav>
</header>
```

En dan gebruik je `include` in een andere pagina:

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

```
<h2>Welkom op de homepagina</h2>
```

```
<p>Dit is de inhoud van index.php</p>
```

Opdracht – Header en footer maken

1. Maak een bestand `header.php` met een kop en menu zoals hierboven.
2. Maak een bestand `footer.php` met daarin bijvoorbeeld:
`<footer>© 2025 Mijn Website</footer>`
3. Maak een bestand `index.php` met bovenin een `include("header.php")` en onderin een `include("footer.php")`.

Reflectie

- Waarom is het handig om met `include` te werken?
- Wanneer zou je liever `require` gebruiken?
- Wat zou er gebeuren als je honderd pagina's hebt zonder `include` te gebruiken? In welk geval zou dat onhandig zijn?

Inleveren

- Beantwoord de drie vragen in eigen woorden uit de reflectie en lever die in (.txt of .pdf bestand).

3 Arrays en Loops

Leerdoelen

- Je weet wat een array is in PHP.
- Je kunt een array maken met meerdere waarden.
- Je kunt een `foreach`-loop gebruiken om alle items te tonen.

Uitleg

Een **array** is een soort lijstje waarin je meerdere dingen kunt bewaren, zoals hobby's of namen.

Een array is een variabele die meer dan één item (waarde) kan bevatten.

Je gebruikt een `foreach`-loop om elk item uit de array één voor één te gebruiken.

Voorbeeld:

```
<?php
$hobbies = ["voetbal", "lezen", "gamen"];

foreach ($hobbies as $hobby) {
    echo "<p>Mijn hobby is: $hobby</p>";
}
?>
```

Resultaat:

Mijn hobby is: voetbal

Mijn hobby is: lezen

Mijn hobby is: gamen

Opdracht – Eigen array

Maak een bestand `lijst.php` en vul dit met een array van 5 dingen die jij leuk vindt (hobby's, favoriete eten, games...).

- Gebruik een `foreach`-loop om ze allemaal op het scherm te tonen.
- Geef elk item weer in een eigen paragraaf (`<p>`).

Uitleg

Er is ook een ander soort loop; een `for`-loop.

Gebruik een `for`-loop in plaats van `foreach`:

```
<?php
$games = ["Minecraft", "FIFA", "Fortnite", "Roblox"];
for ($i = 0; $i < count($games); $i++) {
```

```
echo "<p>" . $i . " Favoriete game: " . $games[$i] . "</p>";  
}  
?>
```

De waarde van `$games[0]` is dus "Minecraft", `$games[1]` = "FIFA", etc, etc.

Het eerste element in een array heeft een index 0!

Opdracht - deel 2

Plaats in de PHP-code (`lijst.php`) een eigen voorbeeld waarbij je een `for-loop` gebruikt.

Reflectie

- Wat is het voordeel van een array ten opzichte van losse variabelen?
- Wat is het verschil tussen `for` en `foreach`?
- Wanneer zou je liever een `for`-loop gebruiken?

Inleveren

- Lever een screenshot in van jouw `lijst.php` met minimaal 5 items in een array, getoond in de browser.
- Laat zien dat je zowel `foreach` als `for` gebruikt hebt.

4 Quiz met arrays en loops

Leerdoelen

- Je kunt een quiz maken met meerdere vragen in een formulier.
- Je kunt de juiste antwoorden opslaan in een array.
- Je kunt een loop gebruiken om automatisch te controleren hoeveel vragen goed zijn.

Uitleg

Met een array kun je makkelijk meerdere juiste antwoorden opslaan. Met een loop kun je daar automatisch doorheen lopen om te controleren hoeveel antwoorden goed zijn ingevuld.

Opdracht 1 – quiz.html

Maak een nieuw bestand `quiz.html` met dit formulier:

```
<!DOCTYPE html>
<html>
<body>

  <h2>Mini-quiz</h2>

  <form action="uitslag.php" method="get">

    <p>1. Wat is de hoofdstad van Nederland?</p>
    <input type="radio" name="vraag1" value="A"> A. Rotterdam<br>
    <input type="radio" name="vraag1" value="B"> B. Amsterdam<br>
    <input type="radio" name="vraag1" value="C"> C. Utrecht<br>

    <p>2. Wat betekent HTML?</p>
    <input type="radio" name="vraag2" value="A"> A. Hyperlink Text Markup Language<br>
    <input type="radio" name="vraag2" value="B"> B. HyperText Markup Language<br>
    <input type="radio" name="vraag2" value="C"> C. Home Tool Markup Language<br>

    <p>3. Wat doet PHP?</p>
    <input type="radio" name="vraag3" value="A"> A. Styling toevoegen aan websites<br>
    <input type="radio" name="vraag3" value="B"> B. Informatie opslaan en berekenen op de server<br>
    <input type="radio" name="vraag3" value="C"> C. Tekst zichtbaar maken in Word<br>

    <br><input type="submit" value="Verzend antwoorden">

  </form>

</body>
</html>
```

Opdracht 2 – uitslag.php met array en loop

Maak een nieuw bestand `uitslag.php` en gebruik deze code:

```
<?php
$goed = 0;

// juiste antwoorden in een array
$juisteAntwoorden = [
    "vraag1" => "B",
    "vraag2" => "B",
    "vraag3" => "B"
];

$goed = 0;

if (isset($_GET['vraag1']) && $_GET['vraag1'] == 'a') {
    $goed++;
}
if (isset($_GET['vraag2']) && $_GET['vraag2'] == 'c') {
    $goed++;
}
if (isset($_GET['vraag3']) && $_GET['vraag3'] == 'b') {
    $goed++;
}

echo "<h2>Je hebt $goed van de 3 vragen goed!</h2>";
?>
```

Opdracht

Bereid de quiz uit met 2 zelf bedachte vragen.

Zet de code in `uitslag.php` om, waarbij je de score berekend met behulp van een loop.

Vraag AI om hulp en vraag AI om uit te leggen hoe de loop werkt.

☐☐ Reflectie

- In het HTML-formulier staan telkens drie `<input>` elementen met dezelfde naam. Leg uit waarom dat zo is.
- Wat is het voordeel van een array gebruiken voor de juiste antwoorden?
- Wat gebeurt er als je geen antwoord op een vraag invult?
- Wat doet `isset()` op regel 13, 16 en 19?
- Hoe zou je de quiz uitbreiden naar 5 of 10 vragen met zo min mogelijk extra code?

☐☐ Inleveren

- `quiz.html` en `uitslag.php`

5 Functies in PHP

☐☐ Leerdoelen

- Je weet wat een functie is in PHP.
- Je kunt zelf een functie schrijven en aanroepen.
- Je kunt gegevens meegeven aan een functie (parameters).

☐☐ Uitleg

Een functie is een blokje code dat je een naam geeft en later opnieuw kunt gebruiken. Zo hoeft je niet telkens dezelfde code opnieuw te schrijven.

Je kunt informatie aan een functie meegeven (dat noem je een **parameter**) en je kunt iets teruggeven (dat noem je een **return**).

Voorbeeld:

```
<?php
function begroet($naam) {
```

```
echo "Hallo, $naam!<br>";  
}  
  
begroet("Fatima");  
begroet("Ali");  
?>
```

☐ Opdracht 1 – Maak je eigen begroetfunctie

1. Maak een nieuw bestand `functie.php`
2. Maak daarin een functie `begroet` die één parameter accepteert: `$naam`
3. Laat de functie een boodschap tonen zoals “Hoi, [naam], welkom terug!”
4. Roep de functie minstens 3 keer aan met verschillende namen.

☐ Opdracht 2 – Maak een rekentool met functie

1. Maak een functie `kortingBerekenen` die twee getallen als parameter gebruikt: `$bedrag` en `$percentage`
2. Laat de functie het bedrag na korting `returnen`
3. Laat het resultaat op het scherm zien met `echo`

Voorbeeldcode:

```
<?php  
function kortingBerekenen($bedrag, $korting) {  
    $nieuwBedrag = $bedrag - ($bedrag * ($korting / 100));  
    return $nieuwBedrag;  
}  
  
echo "<p>Je betaalt nu: €" . kortingBerekenen(100, 25) . "</p>";  
?>
```

☐☐ Reflectie

- Waarom is het handig om functies te gebruiken?
- Wat gebeurt er als je geen parameter meegeeft aan een functie die er wel één verwacht?
- Waar zou je functies nog meer voor kunnen gebruiken?

☐☐ Inleveren

- `functie.php` met ten minste 1 begroetfunctie én 1 rekentool.
Je moet ten minste 2 functies gebruiken met parameters, waarvan 1 ook een `return` gebruikt.

6 Datum en Tijd in PHP

☐☐ Leerdoelen

- Je weet hoe je de huidige datum en tijd kunt tonen met PHP.
- Je kunt berekenen hoeveel dagen of jaren geleden iets was.
- Je kunt de leeftijd berekenen op basis van een geboortedatum.

☐☐ Uitleg

PHP heeft functies om met datum en tijd te werken. De belangrijkste is `date()`, waarmee je de huidige tijd kunt weergeven in verschillende formaten.

- `date("Y")` → jaar (bijv. 2025)
- `date("d-m-Y")` → dag-maand-jaar (bijv. 04-06-2025)
- `date("H:i")` → tijd in uren:minuten (bijv. 14:36)

Voorbeeld:

```
<?php
echo "Vandaag is het: " . date("d-m-Y") . "<br>";
echo "Het is nu: " . date("H:i") . " uur.";
?>
```

Opdracht 1 – Toon de datum en tijd

1. Maak een bestand `datum.php`
2. Laat daarin zien:
 - Welke dag het vandaag is
 - De datum in formaat `dd-mm-yyyy`
 - De tijd in uren en minuten

Opdracht 2 – Leeftijd berekenen

Bereken iemands leeftijd op basis van geboortedatum:

```
<?php
$geboortedatum = "2007-03-15";
$geboortedag = new DateTime($geboortedatum);
$vandaag = new DateTime();

$verschil = $vandaag->diff($geboortedag);
echo "Je bent " . $verschil->y . " jaar oud.";
?>
```

Uitleg: PHP kan automatisch het aantal jaren berekenen tussen twee datums met `diff()`.

Reflectie

- Wat is het verschil tussen `date()` en `new DateTime()`?
- Waarom moet je bij het berekenen van leeftijd een `diff()` gebruiken?
- Wat zou er gebeuren als je geboortedatum in een verkeerd formaat schrijft?

□□ Inleveren

- `datum.php` met de juiste datum, tijd en berekende leeftijd.
Je gebruikt minstens één functie met datum en één met tijd, en `diff()` voor de leeftijd.

7 Sessies en Inloggen met PHP

□□ Leerdoelen

- Je weet wat een sessie is in PHP.
- Je kunt een eenvoudige inlogpagina maken.
- Je kunt een gebruiker herkennen op een andere pagina.

□□ Uitleg

Een **sessie** in PHP is een manier om informatie te onthouden zolang de gebruiker je website bezoekt. Bijvoorbeeld of iemand is ingelogd of wat zijn/haar gebruikersnaam is.

Je start een sessie met:

```
<?php
session_start();
?>
```

□□ Opdracht 1 – inloggen.html

Maak een bestand `inloggen.html` met een formulier waarin je gebruikersnaam invoert:

```
<!DOCTYPE html>
<html>
<body>

<h2>Inloggen</h2>
```

```
<form action="login.php" method="post">
  <label for="naam">Naam:</label><br>
  <input type="text" name="naam" id="naam" required><br><br>
  <input type="submit" value="Inloggen">
</form>

</body>
</html>
```

Opdracht 2 – login.php

Maak een bestand `login.php` dat de naam opslaat in een sessie en doorstuurt:

```
<?php
session_start();
$_SESSION["gebruiker"] = $_POST["naam"];
header("Location: welkom.php");
exit;
?>
```

Opdracht 3 – welkom.php

Maak een bestand `welkom.php` dat de gebruiker begroet:

```
<?php
session_start();
if (isset($_SESSION["gebruiker"])) {
    echo "<h1>Welkom, " . $_SESSION["gebruiker"] . "!</h1>";
    echo '<p><a href="uitloggen.php">Uitloggen</a></p>';
} else {
    echo "<p>Je bent niet ingelogd.</p>";
}
?>
```

Opdracht 4 – uitloggen.php

```
<?php
session_start();
session_destroy();
header("Location: inloggen.html");
exit;
?>
```

☐☐ Reflectie

- Waarom moet je altijd `session_start()` gebruiken bovenaan?
- Wat gebeurt er als je probeert `welkom.php` te openen zonder in te loggen?
- Wat zou je kunnen uitbreiden, bijvoorbeeld met wachtwoordcontrole?

☐☐ Inleveren

- Screenshots van `inloggen.html`, `welkom.php` met jouw naam, en `uitloggen.php` na het uitloggen.
- Je moet gebruik maken van sessies en de naam van de gebruiker correct kunnen tonen op meerdere pagina's.

8 *Inloggen met wachtwoordcontrole*

☐☐ Leerdoelen

- Je maakt een formulier met gebruikersnaam én wachtwoord.
- Je controleert de invoer in PHP.
- Je begrijpt waarom `$_GET` niet veilig is voor wachtwoorden.
- Je leert werken met een associatieve array.

Uitleg

Een loginformulier stuurt gebruikersnaam en wachtwoord naar PHP. In deze les gebruiken we eerst `$_GET` om te laten zien waarom dat niet veilig is – je ziet het wachtwoord in de URL.

Opdracht 1 – login.html

Maak een bestand `login.html`:

```
<!DOCTYPE html>
<html>
<body>

  <h2>Loginformulier</h2>

  <form action="controle.php" method="get">
    <label>Gebruikersnaam:</label><br>
    <input type="text" name="gebruiker" required><br><br>

    <label>Wachtwoord:</label><br>
    <input type="password" name="wachtwoord" required><br><br>

    <input type="submit" value="Log in">
  </form>

</body>
</html>
```

Opdracht 2 – controle.php

Maak een bestand `controle.php` waarin je controleert of de gebruiker mag inloggen:

```
<?php
$gebruiker = $_GET["gebruiker"];
$wachtwoord = $_GET["wachtwoord"];

if ($gebruiker == "admin" && $wachtwoord == "geheim123") {
  echo "<h2>Welkom, $gebruiker!</h2>";
}
```

```
} else {  
    echo "<p>Foutieve inloggegevens. Probeer opnieuw.</p>";  
}  
?>
```

Let op:

Als je dit formulier verstuurt, zie je het wachtwoord in de URL. Dat is niet veilig!

☐☐ Opdracht 3 – Verbeter met POST

- Pas het formulier aan zodat het `method="post"` gebruikt
- Pas `controle.php` aan zodat het `$_POST` gebruikt
- Test: zie je het wachtwoord nog in de URL?

☐☐ Uitleg – Associatieve array

Tot nu toe heb je gewerkt met lijsten zoals:

```
$hobby's = ["voetbal", "gamen", "lezen"];
```

Dit is een **indexed array**: de computer onthoudt zelf de volgorde (index 0, 1, 2).

Een **associatieve array** heeft zelfgekozen namen als index (zogenaamde "keys"):

```
$gebruikers = [  
    "admin" => "geheim123",  
    "student" => "welkom01"  
];
```

Je kunt dan bijvoorbeeld zeggen:

```
echo $gebruikers["admin"]; // toont: geheim123
```

Heel handig voor wachtwoorden of gebruikerslijsten!

☐☐ Extra opdracht – Meerdere gebruikers

Breid `controle.php` uit met een associatieve array van toegestane gebruikers en wachtwoorden:

```

<?php
$gebruikers = [
    "admin" => "geheim123",
    "student" => "welkom01",
    "docent" => "phprules"
];

$gebruiker = $_POST["gebruiker"];
$wachtwoord = $_POST["wachtwoord"];

if (isset($gebruikers[$gebruiker]) && $gebruikers[$gebruiker] == $wachtwoord) {
    echo "<h2>Welkom, $gebruiker!</h2>";
} else {
    echo "<p>Inloggen mislukt.</p>";
}
?>

```

Voeg zelf nog twee gebruikers toe: één voor jouw zelf (dus je eigen voornaam) en één voor een klasgenoot.

☐☐ Reflectie

- Wat is het verschil tussen een indexed array en een associatieve array?
- Waarom is het veiliger om `$_POST` te gebruiken voor wachtwoorden?
- Wat zou je doen om inloggen met een wachtwoord nog veiliger te maken?

☐☐ Inleveren

- controle.php
- Een .txt. of .pdf bestand met de antwoorden op de drie reflectievragen.

PHP Challenge

PHP Challenge - Mini-website bouwen

Doelen

- Je laat zien wat je geleerd hebt over PHP-backend programmeren.
- Je maakt een mini-website met meerdere pagina's, functies, formulieren en sessies.
- Je werkt zelfstandig aan een afgerond eindproject.

Opdracht

Kies één van onderstaande opties of bedenk in overleg je eigen variant:

Optie 1 – Persoonlijke website

- Inlogpagina met naam via sessie
- Een pagina “Over mij” met gegevens in variabelen
- Hobby's in een array en met een `foreach` getoond
- Een formulier waarin iemand je een bericht kan sturen

Optie 2 – Quiz met score

- Meerdere vragen opgeslagen in een array
- Gebruik een `foreach` om antwoorden te controleren
- Gebruik een functie om de score te berekenen
- Gebruik sessie om naam van deelnemer te onthouden

☐☐ Optie 3 – Simpele webshop of rekentool

- Formulier waarin je een product kiest en korting toepast
- Gebruik een functie om korting te berekenen
- Toon de datum en tijd van aankoop
- Maak het overzichtelijk met includes (header/footer)

☐☐ Eisen

Jouw project moet aan de volgende eisen voldoen:

- Minstens 3 aparte pagina's (.php)
- Gebruik van `include()` of `require()` voor menu of footer
- Minstens 1 formulier die met `POST` of `GET` gegevens verwerkt
- Gebruik van een array en loop (`for` of `foreach`)
- Gebruik van minstens 1 zelfgemaakte functie
- Gebruik van sessies (`$_SESSION`)

Gebruik AI om je te helpen, maar begrijp wat er gebeurt!

☐☐ Reflectie

- Wat ging goed in dit project?
- Wat vond je lastig?
- Wat zou je toevoegen als je meer tijd had?

☐☐ Inleveren

- Voeg een kort reflectiedocument (.pdf of .txt) toe waarin je de vragen hierboven beantwoordt
- Laat een docent zien dat alles op jouw computer werkt en dat je aan alle eisen voldoet.