PHP : les notions essentielles du cours

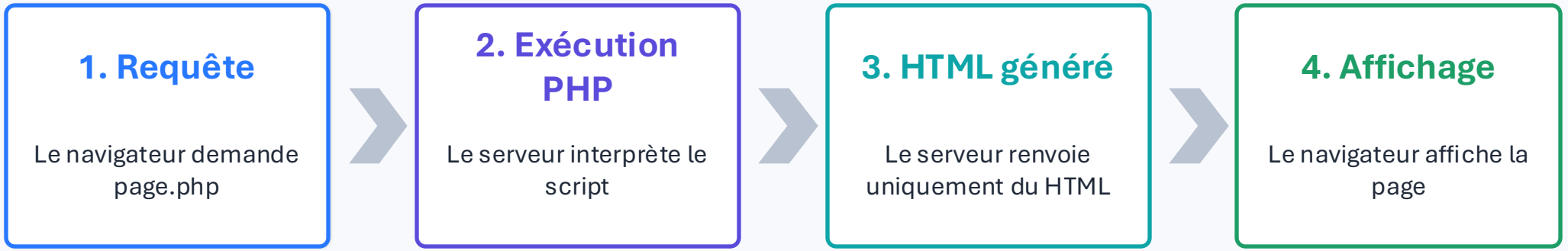
Comprendre le flux, écrire du PHP simple et traiter un formulaire sans erreur.

Variables • Conditions • Tableaux • Fonctions • GET / POST

Objectif : comprendre tout le chapitre

1 MINUTE

Le modèle mental à retenir



Le navigateur ne reçoit jamais le code PHP.

3 MINUTES

Écrire et afficher du contenu

```php
<?php
$nom = "Yoda";
$age = 400;
echo "Bonjour " . $nom;
?>
```

À retenir

- **$** — Une variable commence toujours par $
- **;** — Chaque instruction se termine par ;
- **.** — Le point concatène des chaînes
- **echo** — echo génère du texte ou du HTML

Astuce débogage : var_dump($variable); affiche la valeur et le type.

3 MINUTES

Décider et répéter

Conditions

if / elseif / else
Comparer : < <= > >=
== compare les valeurs
=== compare la valeur ET le type

```php
if ($note >= 10) {
  echo "Admis";
} else {
  echo "Ajourné";
}
```

Boucles

for : nombre d'itérations connu
while : répéter tant qu'une condition est vraie
foreach : parcourir un tableau

```php
foreach ($notes as $note) {
  echo $note;
}
```

3 MINUTES

Les tableaux : clé → valeur

```php
$etudiant = [
  "nom" => "Dupont",
  "note" => 15
];

echo $etudiant["nom"];
```



Parcours indispensable

```php
foreach ($etudiant as $cle => $valeur) {
  echo $cle . " : " . $valeur;
}
```

Fonctions utiles : count() • array_sum() • isset() • unset()

2 MINUTES

Les fonctions : isoler une responsabilité

```php
function moyenne(array $notes): float {
  return array_sum($notes) / count($notes);
}

$resultat = moyenne([12, 15, 18]);
```

Une fonction évite les répétitions et rend le code testable.

Entrées

Les paramètres

Traitement

Le bloc de la fonction

Sortie

La valeur de return

2 MINUTES

Formulaire : du champ au tableau PHP

```html
<form method="post">
  <input name="nom" type="text">
  <button type="submit">Envoyer</button>
</form>
```



```php
if (isset($_POST["nom"])) {
  $nom = $_POST["nom"];
}
```

Le lien essentiel :



Sans attribut name, aucune donnée exploitable n’est envoyée.

1 MINUTE

GET ou POST ?

GET

Données visibles dans l'URL
Pratique pour recherche, filtres et liens partageables
Récupération : $_GET

POST

Données envoyées dans la requête
Pratique pour formulaires et modifications
Récupération : $_POST

POST ne rend pas les données automatiquement sûres.

2 MINUTES

La règle d’or : vérifier puis valider

1

Existe ?

```
1 isset($_POST["age"])
```

2

Format correct ?

```
1 preg_match("/^\d+$/",
...)
```

3

Valeur cohérente ?

```
1 $age >= 0 && $age <=
120
```

Les données utilisateur ne sont jamais fiables par défaut.

1 MINUTE

Cookies et sessions : ne pas confondre

Cookie

Stocké chez le client
Peut durer longtemps
Peut être lu ou modifié par l'utilisateur
Ne pas y stocker d'information sensible

Session

Données stockées sur le serveur
Une session par utilisateur
À activer avec session_start()
Adaptée à l'état de connexion

setcookie() et session_start() doivent être appelés avant tout HTML.

2 MINUTES

Checklist avant de coder

1. Je sais quel HTML je veux produire.
2. Je choisis les bonnes variables et structures.
3. Je parcours les tableaux avec foreach.
4. Je découpe les traitements en fonctions.
5. **Je vérifie et valide chaque donnée reçue.**
6. Je teste souvent avec var_dump().

Si vous retenez une seule chose : PHP reçoit des données, les vérifie, puis génère du HTML.

2 MINUTES

Organiser le code avec require_once

```php
require_once "fonctions.php";
require_once "classes/Joueur.php";
```

Pourquoi découper ?

Séparer l'affichage, les fonctions et les classes.

Pourquoi require_once ?

Inclure une définition une seule fois et éviter les redéfinitions.

require : erreur fatale si le fichier manque • require_once : même principe, sans double inclusion

4 MINUTES

Fonctions : paramètres, types et portée

```php
function moyenne(array $notes): float {
  return array_sum($notes) / count($notes);
}

function saluer(string $nom = "élève"): void {
  echo "Bonjour " . $nom;
}
```

Paramètre par défaut

Utilisé lorsque l'appel ne fournit pas de valeur.

Typage

Le type attendu peut être indiqué pour les paramètres et le retour.

Portée

Une variable locale n'existe que dans la fonction. global est à éviter.

4 MINUTES

Objet et classe : le vocabulaire essentiel

CLASSE

Le modèle ou le plan

Définit :
- les attributs
- les méthodes
- les règles de visibilité

new

OBJETS / INSTANCES

$j1 : Joueur
nom = "Ada"
score = 10

$j2 : Joueur
nom = "Linus"
score = 25

Une classe peut produire plusieurs objets ayant le même comportement, mais des états différents.

5 MINUTES

Définir une classe et créer un objet

```php
class Joueur {
  public string $nom;
  private int $score;

  public function __construct(string $nom) {
    $this->nom = $nom;
    $this->score = 0;
  }

  public function gagner(int $points): void {
    $this->score += $points;
  }
}

$j1 = new Joueur("Ada");
$j1->gagner(10);
```

$this

L'objet courant à l'intérieur d'une méthode.

__construct

Initialise l'objet au moment du new.

->

Accède à un attribut ou appelle une méthode.

Un objet regroupe un état et les opérations autorisées sur cet état.

4 MINUTES

Encapsulation et visibilité

public

Accessible partout

protected

Classe et classes filles

private

Uniquement dans la classe

```php
private int $score;

public function getScore(): int {
  return $this->score;
}

public function gagner(int $points): void {
  if ($points > 0) {
    $this->score += $points;
  }
}
```

Pourquoi private ?

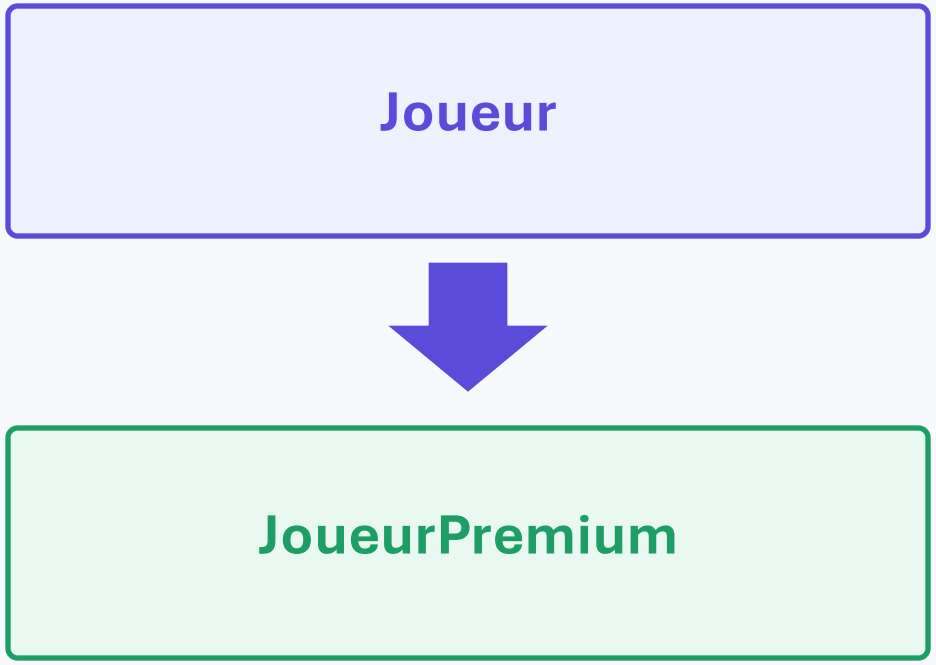
- empêcher une modification incohérente
- centraliser les règles dans les méthodes
- pouvoir changer l'implémentation sans casser les appels

4 MINUTES

Héritage : spécialiser une classe

```php
class JoueurPremium extends Joueur {
  private int $bonus;

  public function gagner(int $points): void {
    parent::gagner($points + $this->bonus);
  }
}
```



extends hérite • parent:: appelle le comportement parent • protected est accessible aux descendants

3 MINUTES

Expressions régulières et die

Expressions régulières

Décrire un format attendu
^ début • $ fin
\d chiffre • + une ou plusieurs fois
preg_match() teste
preg_replace() remplace

```php
if (!preg_match('/^\d+$/', $age)) {
  die("Âge invalide");
}
```

die()

Affiche éventuellement un message puis arrête immédiatement le script. Utile pour bloquer un traitement impossible, mais à utiliser avec un message clair.

2 MINUTES

Vue d’ensemble du chapitre

1. **Recevoir** — GET / POST
2. **Vérifier** — isset / regex
3. **Traiter** — conditions / boucles
4. **Structurer** — fonctions / objets
5. **Conserver** — cookies / session
6. **Produire** — HTML valide

Le point central du cours : transformer des données non fiables en une réponse HTML correcte, avec un code structuré.