Cours 04 - Formulaires HTML et validation

https://github.com/heig-vd-progserv1-course

Support de cours · Présentation (web) · Présentation (PDF)

L. Delafontaine, avec l'aide de GitHub Copilot.

Ce travail est sous licence CC BY-SA 4.0.

Retrouvez plus de détails dans le support de cours

Cette présentation est un résumé du support de cours. Pour plus de détails, consultez le support de cours.

Objectifs (1/2)

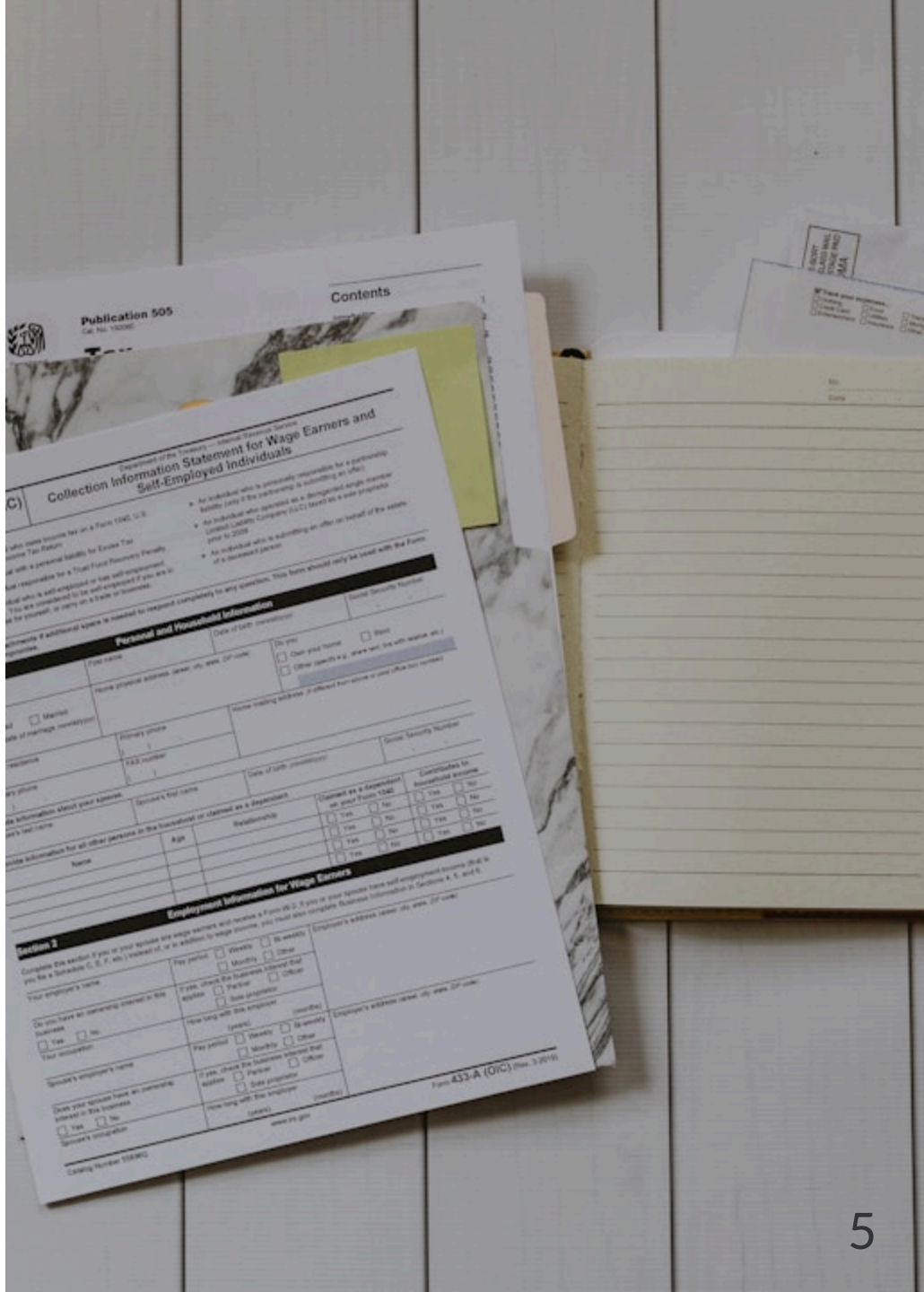
- Créer des formulaires HTML pour collecter des données utilisateur.
- Envoyer des données de formulaires au serveur à l'aide de PHP.
- Récupérer les données envoyées par le formulaire à l'aide de PHP.
- Expliquer la différence entre les méthodes `GET` et `POST`.

Objectifs (2/2)

- Valider les données saisies par l'utilisateur côté serveur et côté client.
- Afficher des messages d'erreur clairs en cas de validation échouée.
- Pré-remplir les champs de formulaire avec les valeurs précédemment saisies.

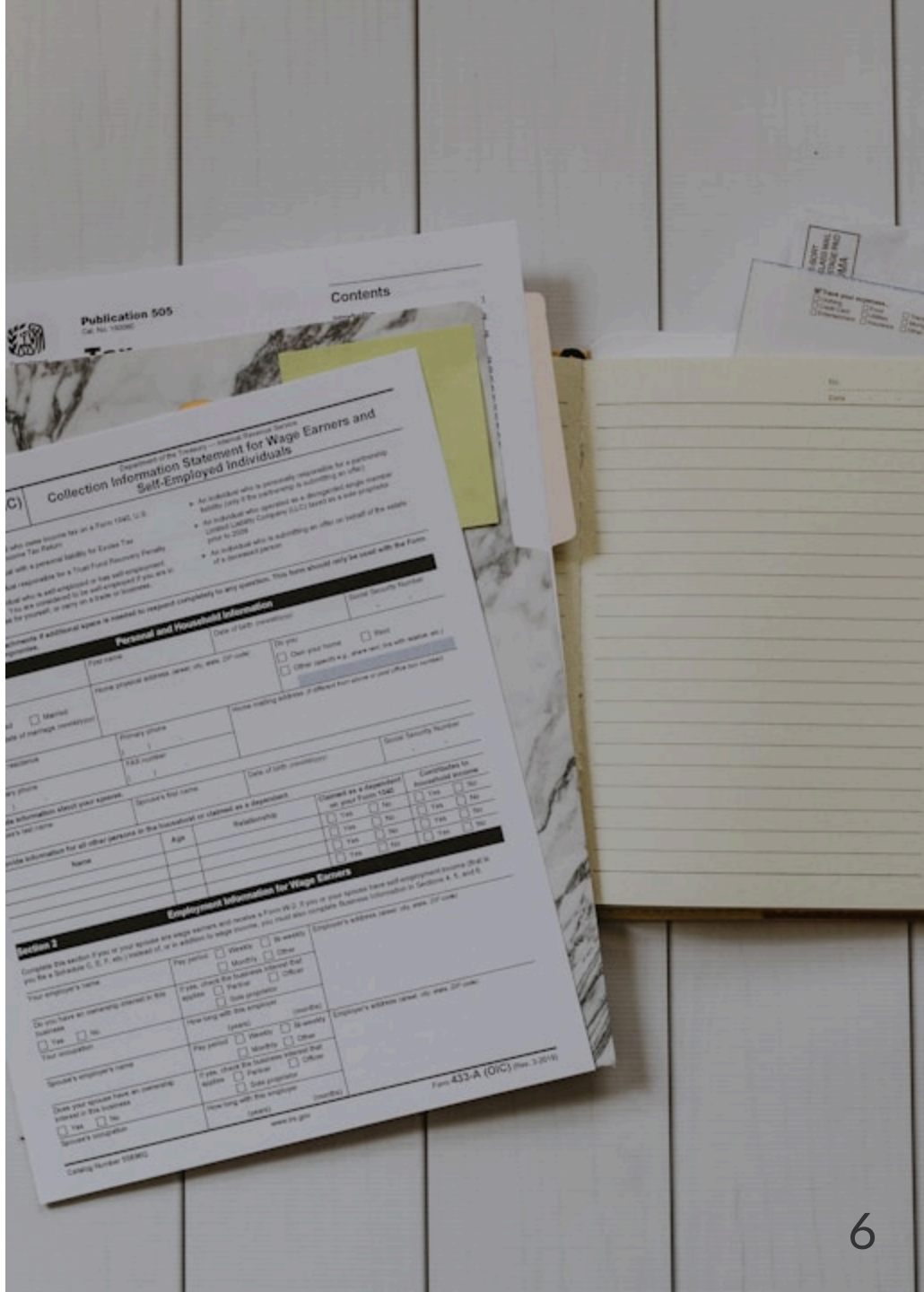
Formulaires HTML

- Permettent de collecter des données utilisateur.
- Sont composés de champs de saisie (input fields) et de boutons.
- Sont généralement utilisés pour des actions telles que l'inscription, la connexion, la recherche, etc.
- Rendent l'expérience utilisateur plus interactive et dynamique.



Structure d'un formulaire (1/2)

- Définit à l'aide de la balise `<form>`.
- Contient des éléments de formulaire tels que des champs de saisie, des boutons, etc.
- Peut inclure des attributs pour spécifier l'URL d'action, la méthode d'envoi, etc.

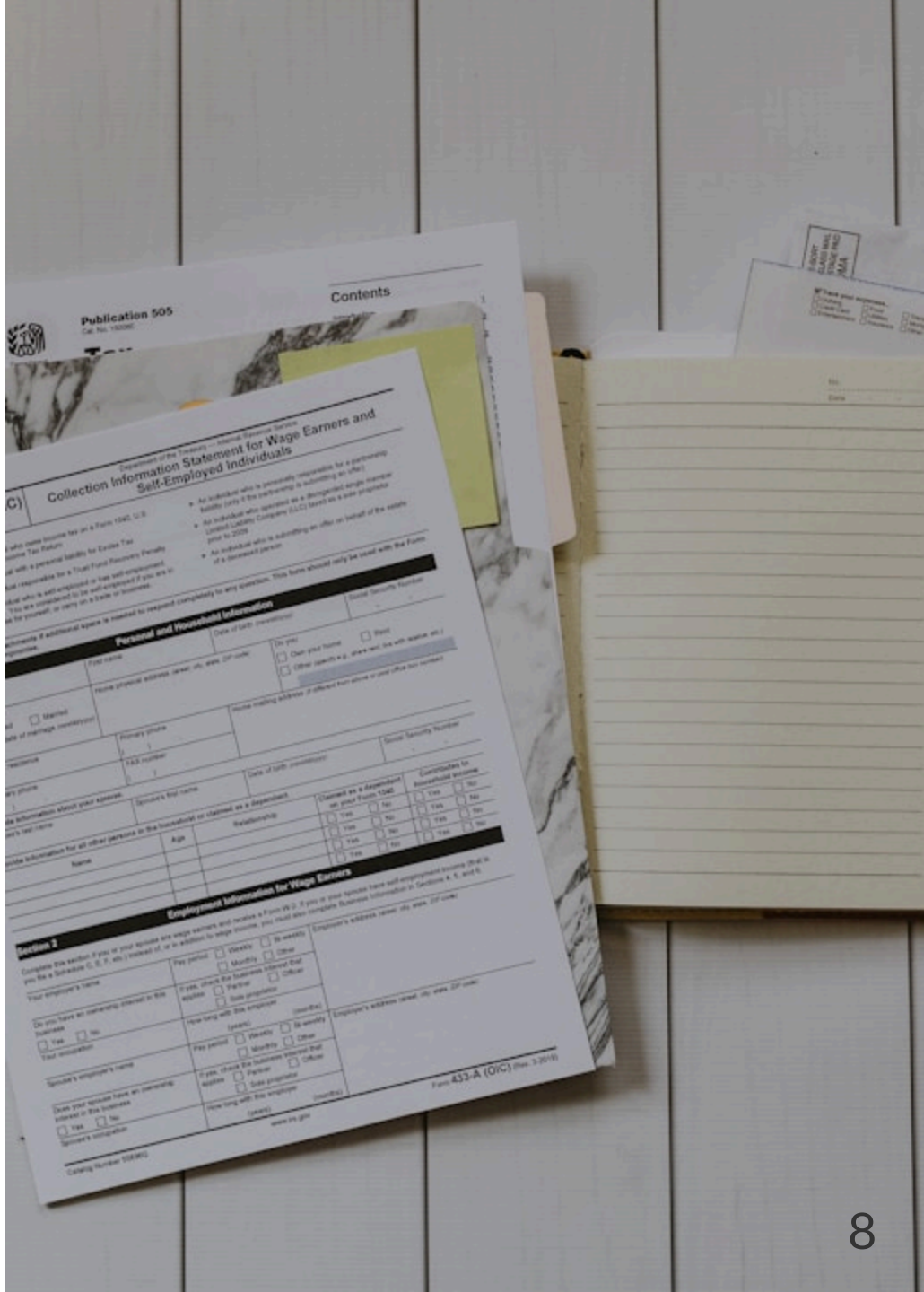


Structure d'un formulaire (2/2)

```
<form action="register.php" method="POST">
        <label for="username">Pseudo :</label><br />
        <input type="text" id="username" name="username" />

        <label for="password">Mot de passe :</label><br />
        <input type="password" id="password" name="password" />

        <br />

        <button type="submit">Envoyer</button>
</form>
```

Champs `<input>` (1/2)

- Champs de saisie de texte, mot de passe, e-mail, etc.
- Définis à l'aide de la balise `<input>`.
- Plusieurs types disponibles (entre autres) :
  - **`text`** : champ texte
  - **`password`** : champ mot de passe
  - **`email`** : champ e-mail
  - **`number`** : champ numérique

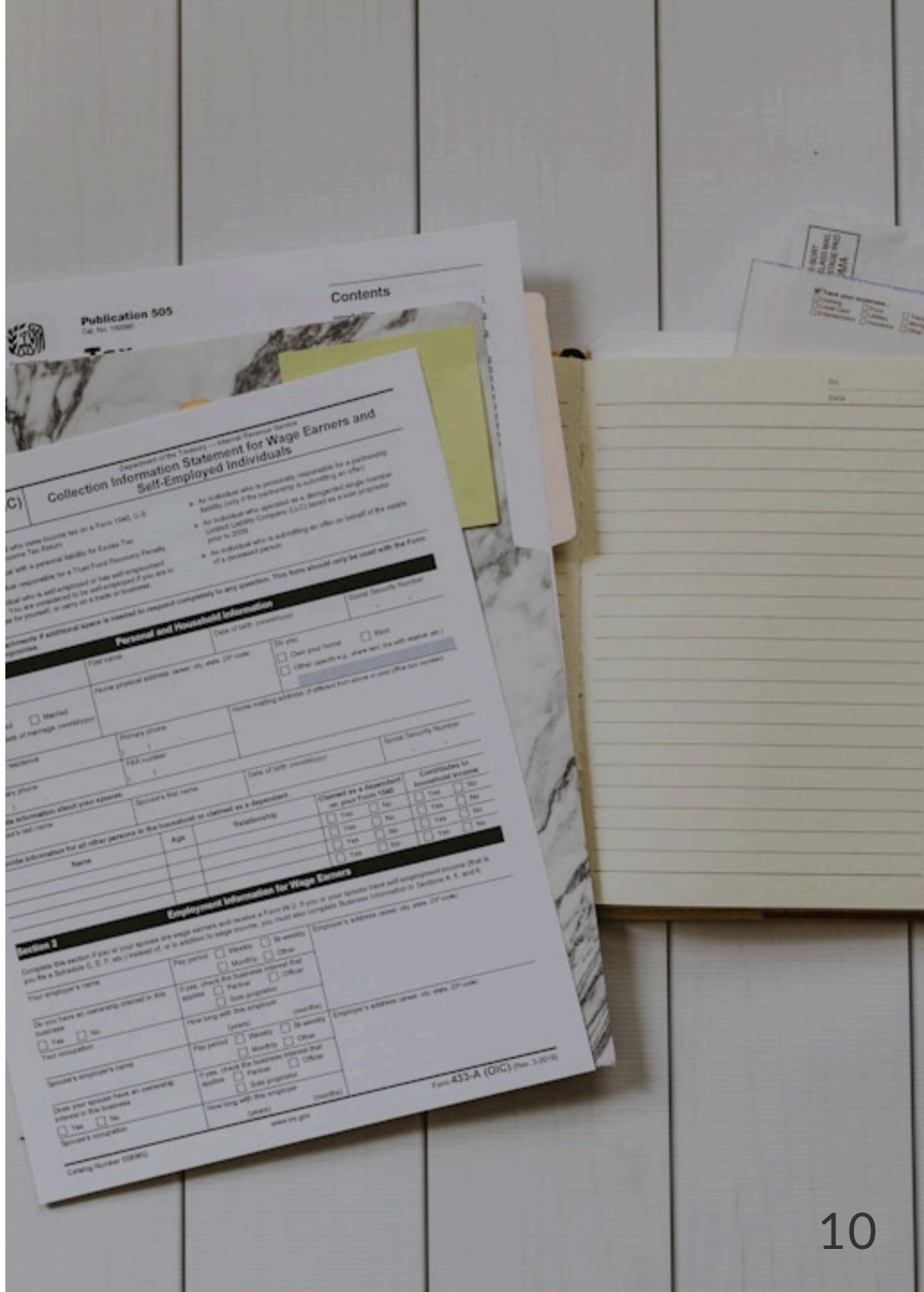


Champs `<input>` (2/2)

```html
<!-- Champ de texte -->
<input type="text" name="name" />

<!-- Champ de email -->
<input type="email" name="email" />

<!-- Champ de mot de passe -->
<input type="password" name="password" />

<!-- Champ de case à cocher -->
<input type="checkbox" name="subscribe" value="yes" />
```

Champ `<select>` (1/2)

- Liste déroulante permettant de sélectionner une option parmi plusieurs.
- Défini à l'aide de la balise `<select>`.
- Les options sont définies à l'aide de la balise `<option>`.

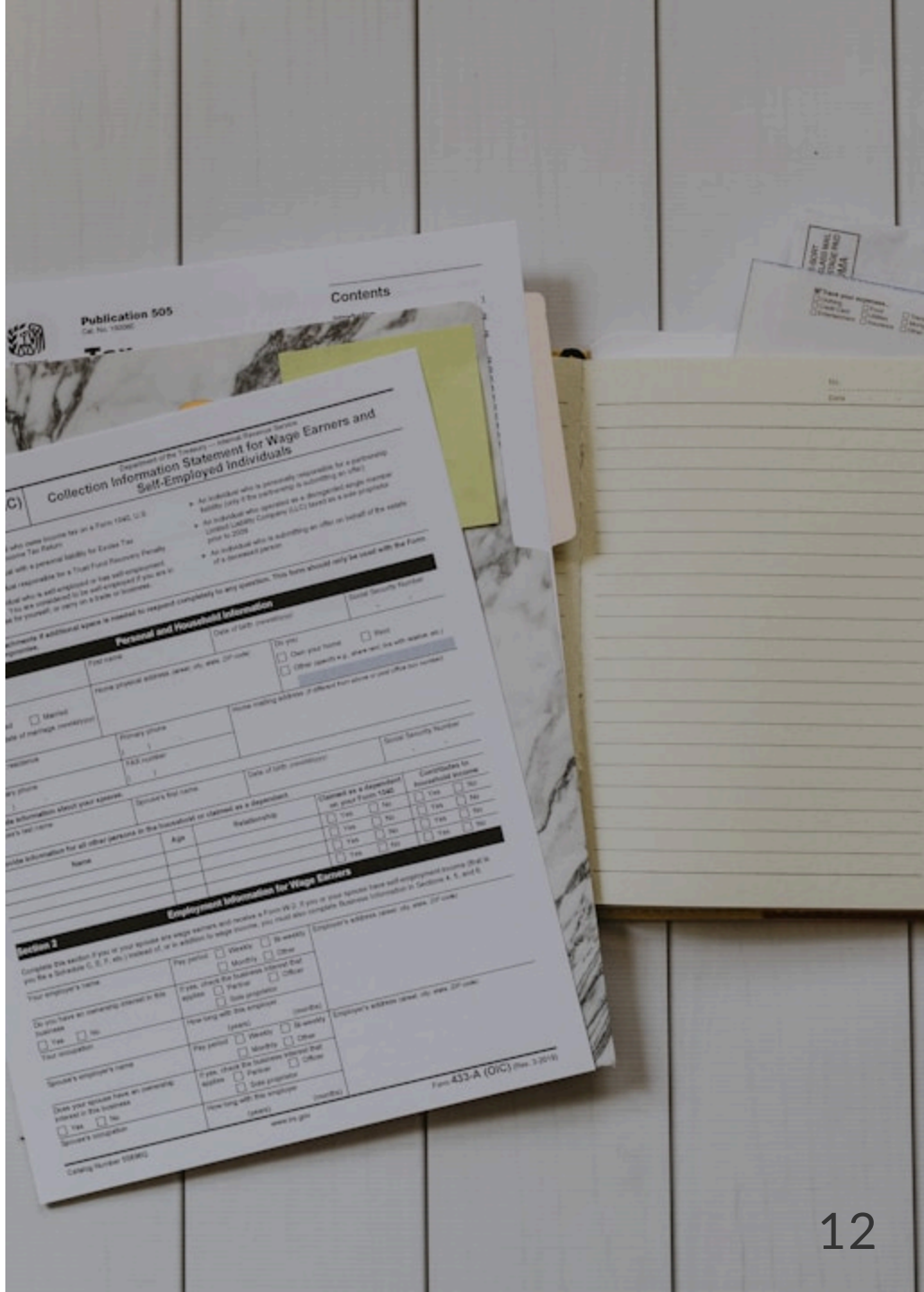


Champ `<select>` (2/2)

```
<select name="country">
        <option value="france">France</option>
        <option value="switzerland">Suisse</option>
        <option value="belgium">Belgique</option>
</select>
```

Champ `<textarea>` (1/2)

- Champ de saisie de texte multiligne.
- Utilisé pour des commentaires, des messages, etc.
- Défini à l'aide de la balise `<textarea>`.
- Peut inclure des attributs pour spécifier la taille, le nombre de lignes, etc.

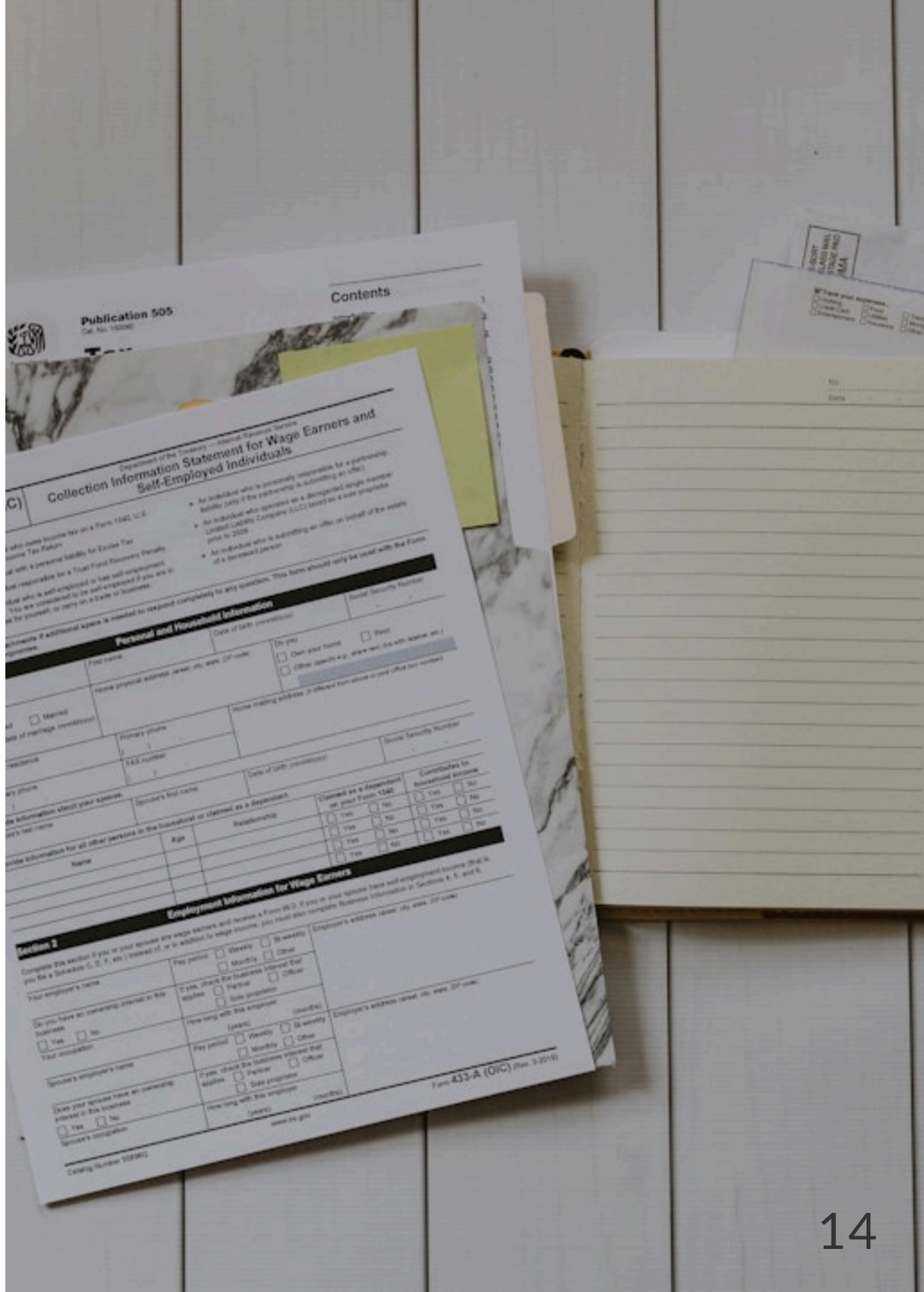


Champ `<textarea>` (2/2)

```
<textarea name="message" rows="4" cols="50">
    Écrivez votre message ici...
</textarea>
```

Champs `<button>` (1/2)

- Boutons pour soumettre ou réinitialiser le formulaire.
- Défini à l'aide de la balise `<button>`.
- Peut inclure des attributs pour spécifier le type de bouton (`submit` , `reset` , etc.).
- Peut contenir du texte ou des images.

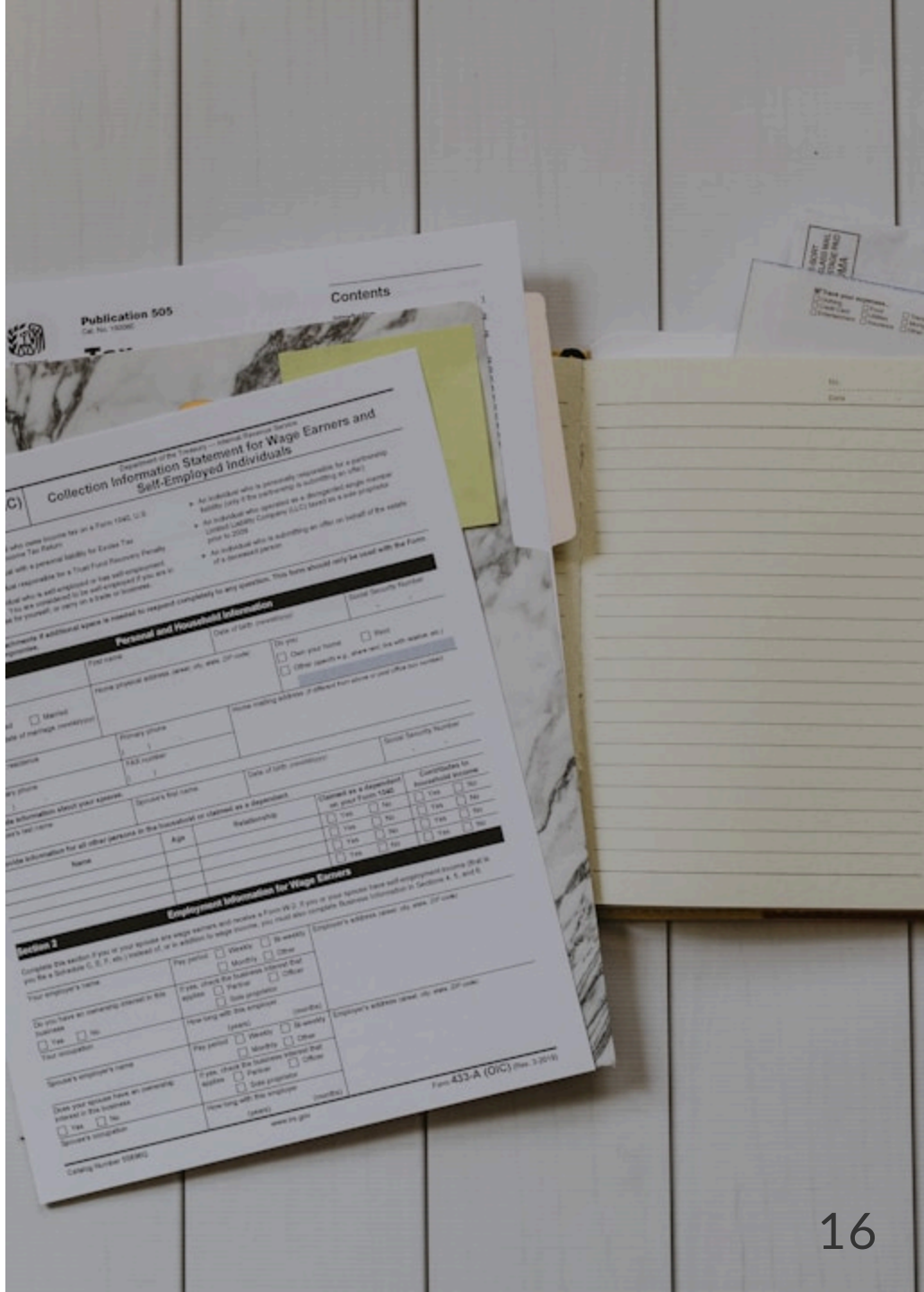


Champ `<button>` (2/2)

```html
<!-- Ce bouton soumet le formulaire -->
<button type="submit">Envoyer</button>

<!-- Ce bouton réinitialise le formulaire -->
<button type="reset">Réinitialiser</button>
```

Attributs

- Les attributs HTML permettent de personnaliser le comportement des éléments de formulaire.
- Des attributs courants sont :
  - `name` : nom du champ
  - `id` : identifiant unique
  - `value` : valeur par défaut
  - `placeholder` : aide à la saisie
  - `required` : champ obligatoire



Envoyer les données des formulaires

- L'envoi des données au serveur se fait grâce au bouton de soumission.
- Composé de deux parties :
  - **URL d'action** : où les données sont envoyées
  - **Méthode** : comment les données sont envoyées

URL d'action

- Définit la destination des données du formulaire grâce à l'attribut `action`.
- Peut être une page PHP, un script, une API, etc.
- Exemple : `action="register.php"`.

Méthodes `GET` et `POST` (1/4)

La méthode (`method`) du formulaire définit comment les données sont envoyées au serveur. Il existe deux méthodes principales :

`GET` : envoie les données dans l'URL (visible dans la barre d'adresse)

- Limité en taille
- Utilisé pour des requêtes non sensibles

`POST` : envoie les données dans le corps de la requête (invisible)

- Pas de limite de taille
- Utilisé pour des données sensibles
- Recommandé en tout temps

Méthodes `GET` et `POST` (2/4)

```html
<!-- La méthode peut être `GET` ou `POST` -->
<form action="login.php" method="">
        <label for="username">Pseudo : </label><br />
        <input type="text" id="username" name="username" value="xXBestOf1400Xx" />

        <label for="password">Mot de passe :</label><br />
        <input
                type="password"
                id="password"
                name="password"
                value="m0n-sup3r-m0t-de-p4asse"
        />

        <button type="submit">Envoyer</button>
</form>
```

Méthodes `GET` et `POST` (3/4)

`GET`

Les données sont ajoutées à l'URL de la requête, séparées par un `?` et des `&` :

```
http://localhost/login.php?username=xXBestOf1400Xx&password=m0n-sup3r-m0t-de-p4asse
```

Problème de sécurité...

Méthodes `GET` et `POST` (4/4)

`POST`

Les données sont envoyées dans le corps de la requête :

`http://localhost/login.php`

Les données ne sont plus visibles dans l'URL.

Problème de sécurité résolu !

Réceptionner les données des formulaires

- Lorsque le formulaire est soumis, le serveur reçoit les données.
- Ces données peuvent être traitées de différentes manières selon la technologie utilisée.
- En PHP, les données sont accessibles via les superglobales `$_GET` et `$_POST`.

Traitement des données à l'aide des superglobales PHP (1/3)

- Les superglobales sont des tableaux associatifs prédéfinis en PHP.
- Elles contiennent les données envoyées par le formulaire :
  - `$_GET` : contient les données envoyées par la méthode `GET`.
  - `$_POST` : contient les données envoyées par la méthode `POST`.
- Les données sont accessibles par le nom du champ défini dans le formulaire grâce à l'attribut `name` des champs.

Traitement des données à l'aide des superglobales PHP (2/3)

```
<!-- Gère l'affichage du formulaire -->
<!DOCTYPE html>
<html>

<head>
    <title>Authentification</title>
</head>

<body>
    <h1>Se connecter</h1>
```

```
<form action="login.php" method="POST">
    <label for="username">Pseudo :</label><br>
    <input type="text" id="username" name="username" />

    <br>

    <label for="password">Mot de passe :</label><br>
    <input type="password" id="password" name="password" />

    <br>

    <button type="submit">Envoyer</button>
</form>
```

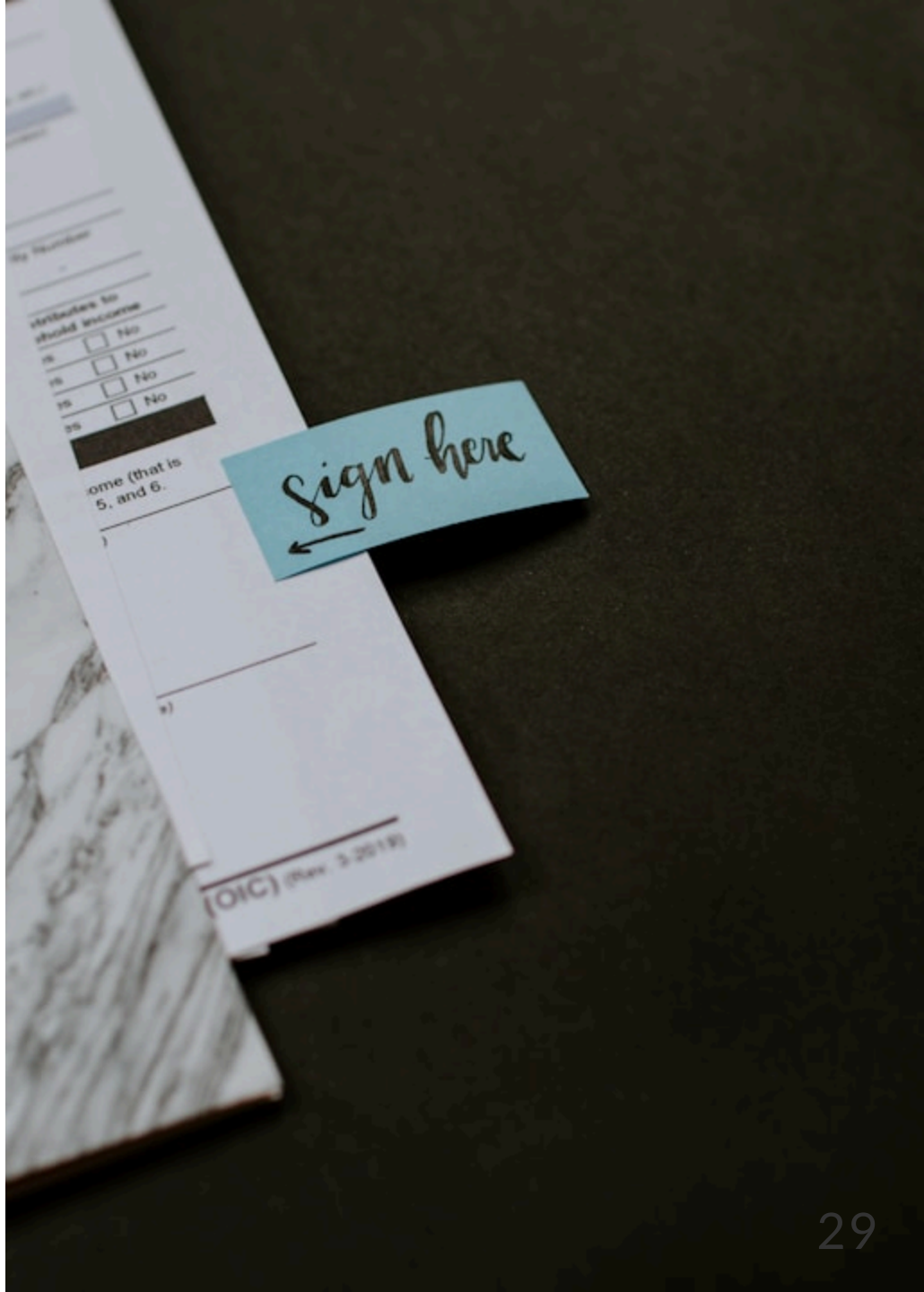
```
</body>

</html>

<?php
// Gère la soumission du formulaire
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST["username"];
    $password = $_POST["password"];

    echo "Le nom d'utilisateur est : " . $username . "<br>";
    echo "Le mot de passe est : " . $password . "<br>";
}
?>
```

Traitement des données à l'aide des superglobales PHP (3/3)

Démonstration

Sauvegarde des données saisies (1/3)

- Lors de la soumission du formulaire, les données sont perdues.
- Il est possible de sauver les valeurs précédemment saisies pour les réutiliser en cas d'erreurs.
- Cela permet de ne pas perdre les informations utiles à l'utilisateur.



Sauvegarde des données saisies (2/3)

```php
<?php
// Gère la soumission du formulaire
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST["username"];
    $password = $_POST["password"];
}
?>

<!-- Gère l'affichage du formulaire -->
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Authentification</title>
</head>

<body>
    <h1>Se connecter</h1>
    <form action="login.php" method="POST">
        <label for="username">Pseudo :</label><br>
        <input
            type="text"
            id="username"
            name="username"
            value="<?php echo isset($username) ? $username : ''; ?>" />

        <br>
```

```
        <label for="password">Mot de passe :</label><br>
        <input
            type="password"
            id="password"
            name="password" />

        <br>

        <button type="submit">Envoyer</button>
    </form>
```

```
    <?php
    // Gère la soumission du formulaire
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $username = $_POST["username"];
        $password = $_POST["password"];

        echo "Le nom d'utilisateur est : " . $username . "<br>";
        echo "Le mot de passe est : " . $password . "<br>";
    }
    ?>
</body>

</html>
```

Sauvegarde des données saisies (3/3)

Démonstration

Validation des formulaires

- Valider les données saisies permet de s'assurer qu'elles sont correctes et conformes aux attentes.
- La validation peut se faire à deux endroits :
  1. Côté serveur
  2. Côté client

Côté serveur (1/3)

La validation des formulaires peut inclure des vérifications telles que :

- Vérifier que les champs obligatoires sont remplis.
- Vérifier que les données saisies respectent un format spécifique (par exemple, une adresse e-mail valide ou longueur minimale, etc.).
- Etc.

Côté serveur (2/3)

```php
<?php
// Gère la soumission du formulaire
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST["username"];
    $password = $_POST["password"];

    // Par défaut, il n'y a pas d'erreurs
    $errors = [];

    // Validation des données
    if (empty($username)) {
        // On ajoute un message d'erreur au tableau
        array_push($errors, "Le pseudo est obligatoire.");
    }
```

```php
    if (strlen($username) < 2) {
        // On ajoute un message d'erreur au tableau
        array_push($errors, "Le pseudo doit contenir au moins 2 caractères.");
    }

    if (empty($password)) {
        // On ajoute un message d'erreur au tableau
        array_push($errors, "Le mot de passe est obligatoire.");
    }

    if (strlen($password) < 8) {
        // On ajoute un message d'erreur au tableau
        array_push($errors, "Le mot de passe doit contenir au moins 8 caractères.");
    }
}
?>
```

```
<!-- Gère l'affichage du formulaire -->
<!DOCTYPE html>
<html>

<head>
    <title>Authentification</title>
</head>

<body>
    <h1>Se connecter</h1>

    <form action="login.php" method="POST">
        <label for="username">Pseudo :</label><br>
```

```
        <input
            type="text"
            id="username"
            name="username"
            value="<?php echo isset($username) ? $username : ''; ?>" />

        <br>

        <label for="password">Mot de passe :</label><br>
        <input
            type="password"
            id="password"
            name="password" />

        <br>

        <button type="submit">Envoyer</button>
    </form>
```

```
    <?php
    // On affiche les données si le formulaire a été soumis
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        // S'il n'y a pas d'erreurs, on affiche les données
        if (empty($errors)) {
            echo "<p style='color: green;'>Le nom d'utilisateur est : $username</p>";
            echo "<p style='color: green;'>Le mot de passe est : $password</p>";
        } else {
            // S'il y a des erreurs, on les affiche
            foreach ($errors as $error) {
                echo "<p style='color: red;'>$error<p>";
            }
        }
    }
    ?>

</body>

</html>
```

Côté serveur (3/3)

Démonstration

Côté client (1/3)

- L'expérience utilisateur peut être améliorée en faisant une validation côté client également.
- Le navigateur va directement nous indiquer si un champ n'est pas conforme.
- 🚨 **Elle ne remplace pas la validation côté serveur**, car la validation cliente peut être contournée. 🚨

Côté client (2/3)

```php
<?php
// Gère la soumission du formulaire
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST["username"];
    $password = $_POST["password"];

    // Par défaut, il n'y a pas d'erreurs
    $errors = [];

    // Validation des données
    if (empty($username)) {
        // On ajoute un message d'erreur au tableau
        array_push($errors, "Le pseudo est obligatoire.");
    }
```

```
    if (strlen($username) < 2) {
        // On ajoute un message d'erreur au tableau
        array_push($errors, "Le pseudo doit contenir au moins 2 caractères.");
    }

    if (empty($password)) {
        // On ajoute un message d'erreur au tableau
        array_push($errors, "Le mot de passe est obligatoire.");
    }

    if (strlen($password) < 8) {
        // On ajoute un message d'erreur au tableau
        array_push($errors, "Le mot de passe doit contenir au moins 8 caractères.");
    }
}
?>
```

```
<!-- Gère l'affichage du formulaire -->
<!DOCTYPE html>
<html>

<head>
    <title>Authentification</title>
</head>

<body>
    <h1>Se connecter</h1>

    <form action="login.php" method="POST">
        <label for="username">Pseudo :</label><br>
```

```
        <input
            type="text"
            id="username"
            name="username"
            value="<?php echo isset($username) ? $username : ''; ?>"
            required
            minlength="2" />

        <br>

        <label for="password">Mot de passe :</label><br>
        <input
            type="password"
            id="password"
            name="password"
            required
            minlength="8" />

        <br>
```

```
        <button type="submit">Envoyer</button>
    </form>

    <?php if ($_SERVER["REQUEST_METHOD"] == "POST") { ?>
        <?php if (empty($errors)) { ?>
            <p style="color: green;">Le formulaire a été soumis avec succès !</p>
        <?php } else { ?>
            <p style="color: red;">Le formulaire contient des erreurs :</p>
            <ul>
                <?php foreach ($errors as $error) { ?>
                    <li><?php echo $error; ?></li>
                <?php } ?>
            </ul>
        <?php } ?>
    <?php } ?>
</body>

</html>
```

Côté client (3/3)

Démonstration

Conclusion

- Les formulaires permettent la saisie des données de l'utilisateur.
- Ils envoient les données au serveur qui est en charge de les traiter.
- La validation peut se faire du côté serveur ou du côté client (🚨).
- Les messages d'erreur et la sauvegarde des saisies améliorent l'expérience utilisateur.

Questions

Est-ce que vous avez des questions ?

À vous de jouer !

- (Re)lire le support de cours.
- Réaliser le mini-projet.
- Faire les exercices.
- Poser des questions si nécessaire.

Pour le mini-projet ou les exercices, n'hésitez pas à vous entraidez si vous avez des difficultés !

Sources (1/2)

- Illustration principale par Richard Jacobs sur Unsplash
- Illustration par Aline de Nadai sur Unsplash
- Illustration par Kelly Sikkema sur Unsplash
- Illustration par Anastasiia Nelen sur Unsplash
- Illustration par Shavr IK sur Unsplash
- Illustration par Kelly Sikkema sur Unsplash
- Illustration par Blake Wisz sur Unsplash
- Illustration par Taylor Vick sur Unsplash

Sources (2/2)

- Illustration par Nikita Kachanovsky sur Unsplash