

# TP Connexion

---

Event, addEventListener

## Event List:

### Click

OnClick : clic de souris sur un objet

Ondblclick : double-clic de la souris sur un objet

### Mouse

Onmousedown : un bouton de la souris est enfoncé

Onmousemove : la souris est déplacée

Onmouseout : la souris est déplacée d'un élément

Onmouseover : la souris est déplacée sur un élément

Onmouseup : le bouton de la souris est relâché

### Focus

Onblur : l'élément perd le focus

Onfocus : l'élément obtient le focus

Onselect : le texte est sélectionné

### Formulaire

Onerror : Une erreur s'est produite lors du chargement d'un document ou d'une image

OnChange : change le contenu

Onreset : le bouton de réinitialisation est cliqué

Onsubmit : bouton de soumission est cliqué

### Clavier

Onkeydown : La touche d'un clavier est enfoncée

Onkeypress : La touche d'un clavier est enfoncée

Onkeyup : La touche d'un clavier est relâchée

## Windows

Onload : une page ou une image est chargée

Onunload : quitter la page

Onresize : la fenêtre ou le cadre est redimensionné

# Attacher un événement

soit la fonction `doSomething` qui peut être remplacé par n'importe quelle fonction :

```
function doSomething() {  
    console.log("Bouton cliqué !");  
}
```

Deux façons d'attacher cette fonction à un événement.

## Utilisation de onclick

html

```
<button id="myButton" onclick="doSomething()">Cliquez-moi</button>
```

js

```
<script>  
function doSomething() {  
    console.log("Bouton cliqué !");  
}  
</script>
```

## Ajouter un écouteur d'événement

html

```
<button id="myButton">Cliquez-moi</button>
```

js

```
<script>  
function doSomething() {  
    console.log("Bouton cliqué !");  
}  
document.getElementById("myButton").addEventListener("click", doSomething);  
</script>
```

ou encore ( fonction anonyme )

```
<script>  
document.getElementById("myButton").addEventListener("click", function() {
```

```
        console.log("Bouton cliqué !");
    });
</script>
```

ou encore ( fonction fléchée )

```
<script>
document.getElementById("myButton").addEventListener("click", () => {
    console.log("Bouton cliqué !");
});
</script>
```

Comme la fonction `foreach` la fonction `addEventListener` utilise une callback

## Avantages de `addEventListener`

1. **Séparation des préoccupations** : `addEventListener` permet de séparer le HTML du JavaScript, ce qui améliore la lisibilité et la maintenabilité du code.
2. **Plusieurs gestionnaires d'événements** : Vous pouvez attacher plusieurs fonctions au même événement sur un élément.
3. **Capture et bouillonnement** : `addEventListener` permet de contrôler la phase de l'événement (capture ou bouillonnement).
4. **Suppression facile des événements** : Vous pouvez facilement supprimer des gestionnaires d'événements avec `removeEventListener`.
5. **Pas de conflits de nommage** : Évite les conflits potentiels avec d'autres attributs ou propriétés nommés `on...`

Bref cette méthode est largement **préférée** au `onclick`.

## Exemple

```
<button id="myButton">Cliquez-moi</button>

<script>
function handlerClick1() {
    console.log("Premier gestionnaire");
}
```

```
}

function handleClick2() {
    console.log("Deuxième gestionnaire");
}

const button = document.getElementById("myButton");

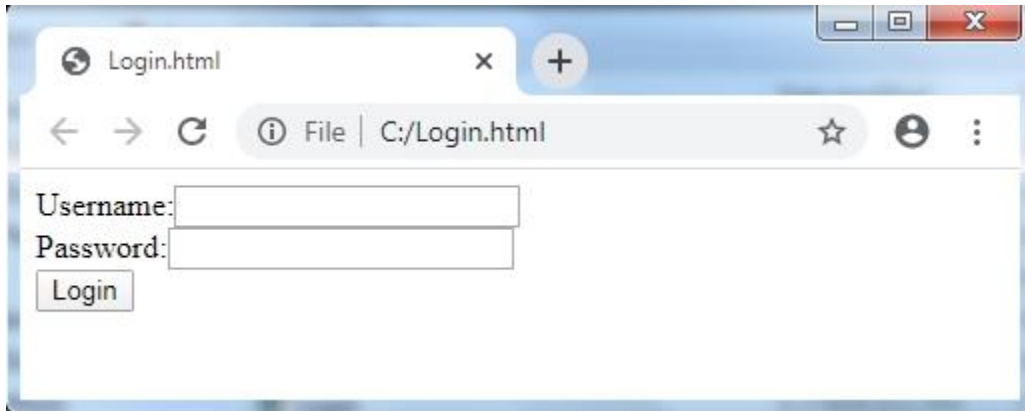
// Ajouter deux gestionnaires d'événements
button.addEventListener("click", handleClick1);
button.addEventListener("click", handleClick2);

// Plus tard, si nécessaire, on peut supprimer un gestionnaire spécifique
// button.removeEventListener("click", handlerClick1);
</script>
</body>
</html>
```

Remarque : “handle” est souvent utilisé comme nom pour la gestion des événements (devient une convention ?)

For example, an `onClick` event running a `handleClick` function.

# Page Web de Connexion




1. Créez un fichier: **Login.html**

```
<p>Username: <input type="text" id="username" value="" /></p>
<p>Password: <input type="password" id="pwd" value="" /></p>
<input type="button" value="Login" onclick="checkLogin()" />
<script type="text/javascript">
  function checkLogin() {
    const usernameObj = document.getElementById("username");
    const pwdObj = document.getElementById("pwd");
    if (usernameObj.value == "") {
      alert("Please input username !");
      return;
    }
    if (pwdObj.value == "") {
      alert("Please input password !");
      return;
    }
    alert("login successfull !");
  }
</script>
```

Exercice :

- Transformer la fonction pour avoir qu'un seul return
- Remplacer le **onclick** par un **addEventListener**

# exercice d'inscription

 Ajouter une section inscription ( en dessous )

Username:

Password:


Login

Username:

Password:

Confirm :

Inscription

 code à compléter

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Connexion et Inscription</title>
  <style>
    .error { color: red; }
    .success { color: green; }
    .weak { color: red; }
    .medium { color: orange; }
    .strong { color: green; }
  </style>
</head>
<body>
  <h2>Connexion</h2>
  <p>Username: <input type="text" id="username" value="" /></p>
  <p>Password: <input type="password" id="pwd" value="" /></p>
  <input type="button" value="Login" onclick="checkLogin()" />

  <h2>Inscription</h2>
  <p>Username: <input type="text" id="newUsername" />
  <span id="usernameStatus"></span></p>
  <p>Password: <input type="password" id="newPwd" />
```

```

<span id="passwordStrength"></span></p>
<p>Confirm : <input type="password" id="confirmPwd" />
<span id="passwordMatch"></span></p>
<input type="button" value="S'inscrire" id="registerButton" />

<h2>Liste des Utilisateurs</h2>
<ul id="userList"></ul>

<script>
// constantes
const loginButton = _____;

// Fonctions
// TODO: Implémenter les fonctions de l'exercice

// Initialisation
function init() {
    // Ajoutez ici tous les event listeners nécessaires
    loginButton.addEventListener("click", checkLogin);

}

// Appelez la fonction d'initialisation quand la page est chargée
window.addEventListener("load", init);
</script>
</body>
</html>

```



## Instructions

1- Implémenter la fonction `checkUsername()` :

- Utiliser `addEventListener` pour détecter quand l'utilisateur a fini de taper.
- Simuler une vérification de disponibilité du nom d'utilisateur.
- Afficher le résultat dans l'élément avec l'id `usernameStatus`.

2- Implémenter la fonction `checkPasswordStrength()` :

- Utiliser `addEventListener` pour vérifier la force du mot de passe en temps réel.
- Définir des critères pour un mot de passe faible, moyen et fort.
- Afficher le résultat dans l'élément avec l'id `passwordStrength`.

3- Implémenter la fonction `checkPasswordMatch()` :

- Utiliser `addEventListener` pour vérifier si les mots de passe correspondent.
- Afficher le résultat dans l'élément avec l'id `passwords Match`.

4- Implémenter la fonction `register()` :

- Vérifier si tous les champs sont correctement remplis.
- Si oui, ajouter l'utilisateur à la liste et l'afficher dans l'élément avec l'id `userList`.
- Sinon, afficher des messages d'erreur appropriés.

5- Compléter la fonction `init()` :

- Ajouter tous les event listeners nécessaires pour les différentes fonctions.

Affichage possible :

Affichage simple des messages :

---

## Inscription

Username:  disponible

Password:  Force Moyenne

Confirm :

Affichage simple de la liste des utilisateurs

---

## Liste des Utilisateurs

- Régis

# Prévention de la soumission

Dans cet exercice, nous allons modifier le formulaire d'inscription HTML pour le rendre standard avec un type de bouton "submit".

Ajouter les formulaires dans des balises `<form>` et modifier le type du bouton

```
<input type="submit" value="S'inscrire" id="registerButton" />
```

Observer le changement.

Par défaut, lorsqu'un formulaire est soumis, la page se recharge et les données sont envoyées au serveur, ce qui empêche toute validation côté client ou affichage de messages d'erreur.

Utilisez `event.preventDefault()` pour empêcher le comportement par défaut du formulaire.

## Bonne pratique

Une bonne pratique consiste à mettre un événement directement sur le formulaire.

```
const form = document.getElementById('registrationForm');

form.addEventListener('submit', function(event) {
  event.preventDefault();
  // Vérifier la validation du formulaire
});
```

Ceci permet aussi de valider le formulaire grâce au bouton entrer. et pas uniquement avec le click du bouton.

Peut-on hacher le mot de passe en js avant d'envoyer le formulaire ?