Interaction homme machine sur le web

Introduction

Qu'est ce que l'IHM ?

L'IHM est l'interface entre l'homme et la machine. Elle permet à l'utilisateur d'interagir avec la machine. Elle est composée de deux parties : l'interface et l'interaction. C'est donc autant l'interface graphique (comme les boutons, les menus, les fenêtres, etc.) que les interactions avec l'utilisateur (comme le clic sur un bouton, le déplacement de la souris, etc.).

Qu'est ce que l'IHM sur le web ?

Le web est un ensemble de pages web reliées entre elles par des liens hypertextes. Il est accessible via un navigateur web. L'IHM sur le web est donc l'interface entre l'utilisateur et le site web. On va par exemple retrouver des boutons, des menus, des formulaires, etc. L'interaction avec l'utilisateur se fait via la souris, le clavier, etc.

Rappels sur les pages web

Une page web est composée de deux parties :

- le code HTML qui permet de structurer la page web
- le code CSS qui permet de mettre en forme la page web

Mais il est aussi possible d'ajouter du code **JavaScript** pour rendre la page web dynamique.

JavaScript

Le JavaScript est un langage de programmation qui permet d'ajouter des fonctionnalités à une page web. Il est exécuté par le navigateur web. Il permet par exemple de modifier le contenu d'une page web, de modifier le style d'une page web, de modifier la structure d'une page web, etc.

> **Remarque** : le JavaScript est un langage de programmation à part entière. Il est donc possible de créer des applications web uniquement avec du JavaScript. On parle alors d'application web monopage.

Intégrer du JavaScript dans une page web

On peut intégrer du JavaScript dans une page web de deux manières :

- en utilisant la balise `<script>` dans le code HTML :

```
<p> Un paragraphe </p>

<script>
    // Code JavaScript
</script>
```

- en utilisant un fichier JavaScript externe :

```
<p> Un paragraphe </p>

<script src="script.js"></script>
```

```
// script.js
// Code JavaScript
```

> **Remarque** : On privilégie l'utilisation d'un fichier JavaScript externe pour séparer le code HTML du code JavaScript.

Exercice 1

index.html

```
<!DOCTYPE html>
<head>
    ...
</head>
<body>
    <h1>Titre</h1>
    <p>Paragraphe</p>
    <button onclick="fonctionMagique()"> Cliquez ici </button>
```

script.js

```
function fonctionMagique() {
    alert("Ca marche !");
}
```

Ajouter le lien vers le fichier script.js dans le fichier index.html pour que le bouton affiche une alerte avec le texte "Ca marche !".

Boutons

Récupérer un élément

On identifie un élément dans la page html par un identifiant unique. Cet identifiant est appelé **id**. Il est possible de récupérer un élément HTML dans le code JavaScript en utilisant la méthode `getElementById()`.

```
<button id="boutonMagique"> Cliquez ici </button>
```

```
// Récupération du bouton
const bouton = document.getElementById("boutonMagique");
```

On peut aussi directement ajouter une action sur un élément HTML en utilisant certain mot clé. Pour faire une action au clic, on peut utiliser l'attribut `onclick` par exemple :

Avec le code dans le html :

```
<button onclick="alert('Cliquez ici')"> Cliquez ici </button>
```

Ou pour appeler une fonction du code JavaScript :

```
<button onclick="maFonction()"> Cliquez ici </button>
```

```
function maFonction() {
    alert("Cliquez ici");
}
```

Les événements

Les événements sont des actions qui se produisent dans une page web. On va pouvoir réagir à ces événements en exécutant du code JavaScript.

Il existe de nombreux événements sur une page web. On peut par exemple citer :

- `click` : lorsqu'on clique sur un élément
- `mouseover` : lorsqu'on passe la souris sur un élément
- `mouseout` : lorsqu'on retire la souris d'un élément
- `keydown` : lorsqu'on appuie sur une touche du clavier
- `submit` : lorsqu'on valide un formulaire

> C'est ce qu'on a vu précédemment avec l'attribut `onclick` par exemple.

Ajouter un événement

Pour ajouter un événement sur un élément HTML, on va utiliser la méthode `addEventListener()`.

```
// Dans le code JavaScript
// Récupération du bouton
const bouton = document.getElementById("boutonMagique");
```

Maintenant, on va pouvoir ajouter un événement sur ce bouton. On va par exemple ajouter un événement au clic sur ce bouton.

```
// Dans le code JavaScript
// Récupération du bouton
const bouton = document.getElementById("boutonMagique");

// On créé une fonction
function clicBouton() {
    alert("Cliquez ici");
}

// Ajout d'un événement au clic sur le bouton avec le nom de la fonction
bouton.addEventListener("click", clicBouton);
```

> **Remarque** : Cela correspond à la méthode `onclick` qu'on a vu précédemment. Mais on va privilégier l'utilisation de `addEventListener()` pour séparer le code HTML du code JavaScript.

Exercice 2

```
<!DOCTYPE html>
<head>
    <script src="script.js"></script>
</head>

<body>
    <h1> Les Dragons </h1>

    <p> Les dragons sont des créatures légendaires représentées comme des sortes de gigantesques reptiles écailleux, généralement ailés, munis de griffes et parfois de crocs, possédant des pouvoirs magiques ou cracheurs de feu. </p>

    <button id="boutonMagique"> Cliquez ici </button>
</body>
```

A partir de cette page html, ajouter un événement au clic sur le bouton pour afficher une alerte avec le texte "Les dragons existent !".

> **Remarque** : On peut utiliser la méthode `alert("contenu")` pour afficher une alerte.

Modifier le contenu d'un élément

On peut modifier directement le contenu d'un élément HTML en utilisant la propriété `innerHTML`.

```
<p id="paragraphe"> Un paragraphe </p>

<button onclick="modifierParagraphe()"> Cliquez ici </button>
```

```
// Récupération du paragraphe
const paragraphe = document.getElementById("paragraphe");

// Fonction pour modifier le contenu du paragraphe
function modifierParagraphe() {
    // On change le contenu du paragraphe
    paragraphe.innerHTML = "Un autre paragraphe";
}
```

Exercice 3

```
<!DOCTYPE html>
<head>
    <script src="script.js"></script>
</head>

<body>
    <h1> Les Dragons </h1>

    <p id="paragraphe"> Les dragons sont de petits lézards </p>

    <button onclick="modifierParagraphe()"> Mettre à jour  </button>
</body>
```

A partir de cette page html, ajouter un événement au clic sur le bouton pour modifier le contenu du paragraphe avec le texte "Les dragons sont de gentils animaux très sympathiques".

Modifier le style d'un élément

On peut modifier directement le style d'un élément HTML en utilisant la propriété `style`.

```
<p id="paragraphe"> Un paragraphe </p>

<button onclick="modifierStyle()"> Cliquez ici </button>
```

```
// Récupération du paragraphe
const paragraphe = document.getElementById("paragraphe");

// Fonction pour modifier le style du paragraphe
function modifierStyle() {
    // On change le style du paragraphe
    paragraphe.style.color = "red";
```

```
    paragraphe.style.fontSize = "20px";
}
```

Formulaires

Les formulaires permettent de récupérer des informations de l'utilisateur. On va par exemple pouvoir récupérer son nom, son prénom, son adresse, etc. On va ensuite pouvoir utiliser ces informations pour les enregistrer dans une base de données, pour les afficher, etc.

Structure d'un formulaire

Un formulaire est composé de plusieurs éléments HTML :

- `form` : le formulaire
- `input` : un champ de saisie
- `label` : un label pour un champ de saisie
- `button` : un bouton pour valider le formulaire

```
<form>
    <label for="nom"> Nom : </label>
    <input type="text" id="nom" name="nom">

    <label for="prenom"> Prénom : </label>
    <input type="text" id="prenom" name="prenom">

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

> **Remarque** : On peut utiliser l'attribut `for` sur un label pour lier le label à un champ de saisie. On peut aussi utiliser l'attribut `name` sur un champ de saisie pour identifier le champ de saisie.

Récupérer les informations d'un formulaire

Pour récupérer les informations d'un formulaire, on va utiliser la propriété `value` sur un champ de saisie.

```
<form>
    <label for="nom"> Nom : </label>
    <input type="text" id="nom" name="nom">

    <label for="prenom"> Prénom : </label>
    <input type="text" id="prenom" name="prenom">

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

```
// Récupération du champ de saisie du nom
const nom = document.getElementById("nom");

// Récupération du champ de saisie du prénom
const prenom = document.getElementById("prenom");

// Fonction pour récupérer les informations du formulaire
function recupererInformations() {
    // On récupère le nom
    const nomUtilisateur = nom.value;

    // On récupère le prénom
    const prenomUtilisateur = prenom.value;

    // On affiche les informations
    alert("Nom : " + nomUtilisateur + "\nPrénom : " + prenomUtilisateur);
}
```

Pour que cette fonction soit appelée lorsqu'on valide le formulaire, on va ajouter un événement sur le formulaire.

```
<form onsubmit="recupererInformations()">
    ...
</form>
```

> **Remarque** : On utilise l'événement `onsubmit` pour réagir à la validation du formulaire. On pourrait aussi utiliser l'événement `onclick` sur le bouton pour réagir au clic sur le bouton.

Exercice 4

Créez un formulaire (uniquement la partie HTML) pour demander "Quel Dragon êtes-vous ?". Le formulaire doit contenir les champs suivants :

- Nom
- Prénom
- Age

Ajoutez un bouton pour valider le formulaire.

Exercice 5

A partir du formulaire de l'exercice précédent, ajoutez un événement sur le formulaire pour récupérer l'age et afficher une alerte avec le texte "Vous êtes un dragon de [age] ans".

Client / Serveur

Quand vous lancer un programme sur votre machine, celui qui demande quelque chose est appelé le **client**. Si tout se passe sur sa propre machine, alors il n'y a que le client. Mais si le client demande quelque chose à

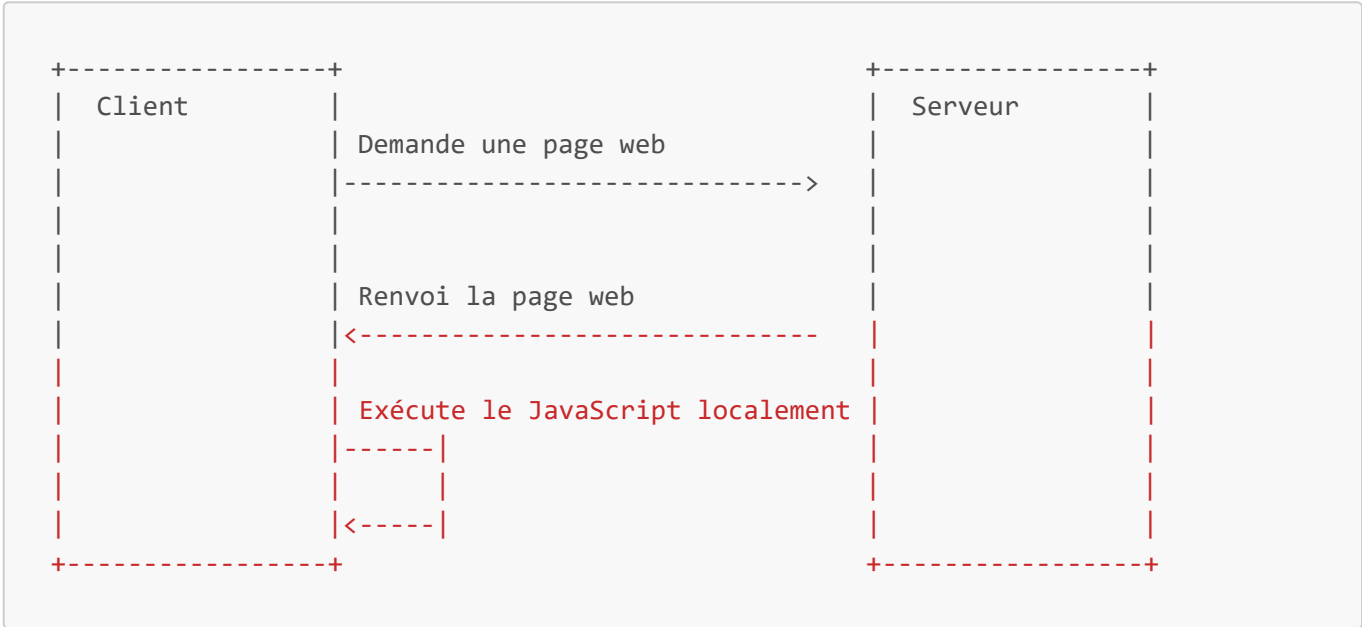
une autre machine, alors cette autre machine est appelée le **serveur**. C'est lui qui va répondre à la demande du client.

Le JavaScript est exécuté par votre navigateur web. C'est donc votre navigateur web qui est le client. Tout se passe chez vous !

Quand vous allez sur un site sur le web, celui-ci est hébergé sur un serveur. Votre navigateur web va donc demander des informations à ce serveur. C'est le serveur qui va répondre à votre demande. Le javascript de la page ne sera exécuté qu'après, sur votre navigateur web.

Il faut bien faire la distinction entre ce qui se passe de façon "locale" (chez le client) et ce qui se passe sur le web (chez le serveur).

Exemple : récupérer une page web



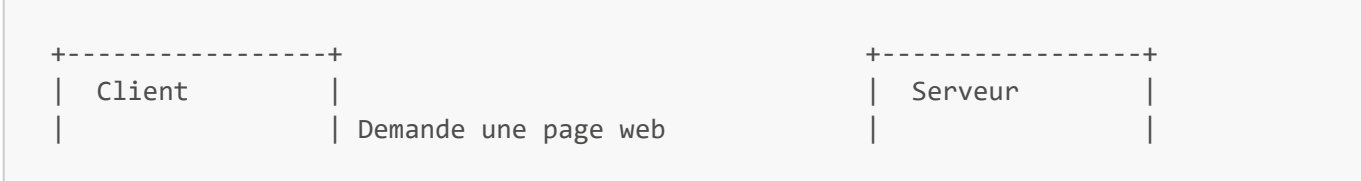
Les requêtes HTTP

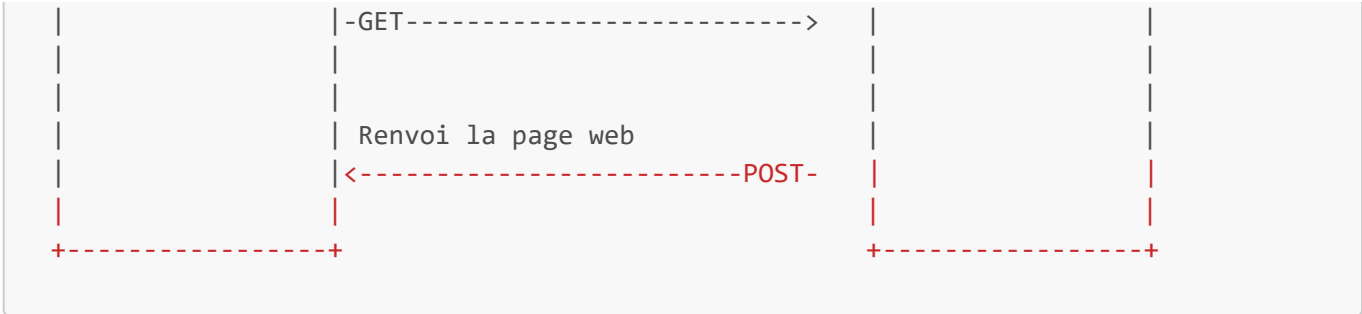
Pour communiquer entre le client et le serveur, on utilise le protocole HTTP. C'est un protocole de communication qui permet d'échanger des informations entre le client et le serveur.

Ce protocole est basé sur des requêtes de différents types standards :

- `GET` : pour récupérer des informations
- `POST` : pour envoyer des informations
- `PUT` : pour modifier des informations
- `DELETE` : pour supprimer des informations
- ... (et bien d'autres)

Exemple : récupérer une page web





Les formulaires utilisent la méthode `POST` pour envoyer les informations du formulaire au serveur. Ensuite, le serveur va pouvoir utiliser ces informations pour les enregistrer dans une base de données par exemple. Quand tout se passe localement, on utilise la méthode `GET` ensuite pour récupérer des informations.

Les formulaires envoient déjà des requêtes HTTP. Mais on peut aussi envoyer des requêtes HTTP manuellement en JavaScript. Pour cela, on va utiliser la méthode `fetch()`.

> Cela permet par exemple de récupérer des informations sur un autre site web.

```
// On envoie une requête GET à l'adresse https://monsite.com
fetch("https://monsite.com")
    .then(function(response) {
        // On récupère la réponse
        return response.text();
    })
    .then(function(text) {
        // On affiche la réponse
        console.log(text);
    });

    // Ce script va afficher le code HTML de la page https://monsite.com
```

> **Remarque** : On utilise la méthode `then()` pour récupérer la réponse de la requête. C'est une méthode asynchrone. Cela signifie que le code ne sera pas exécuté dans l'ordre. On va d'abord envoyer la requête, puis on va récupérer la réponse. C'est pour cela qu'on utilise la méthode `then()` pour récupérer la réponse.

Conclusion

L'IHM sur le web est le coeur de l'intéractivité des pages web. C'est ce qui permet à l'utilisateur d'interagir avec le site web. On a vu comment ajouter des événements sur des éléments HTML pour réagir à des actions de l'utilisateur ou des formulaires. Vous pouvez bien sûr aller plus loin en regardant les (très) nombreuses autres possibilités du JavaScript. Surtout, n'hésitez pas à vous renseigner sur internet pour aller plus loin et amusez-vous !