Chapitre 6 – IHM WEB côté

**Compétences** :

* Identifier les différents composants graphiques permettant d’interagir avec une application Web. Identifier les événements que les fonctions associées aux différents composants graphiques sont capables de traiter.
* Analyser et modifier les méthodes exécutées lors d’un clic sur un bouton d’une page Web.

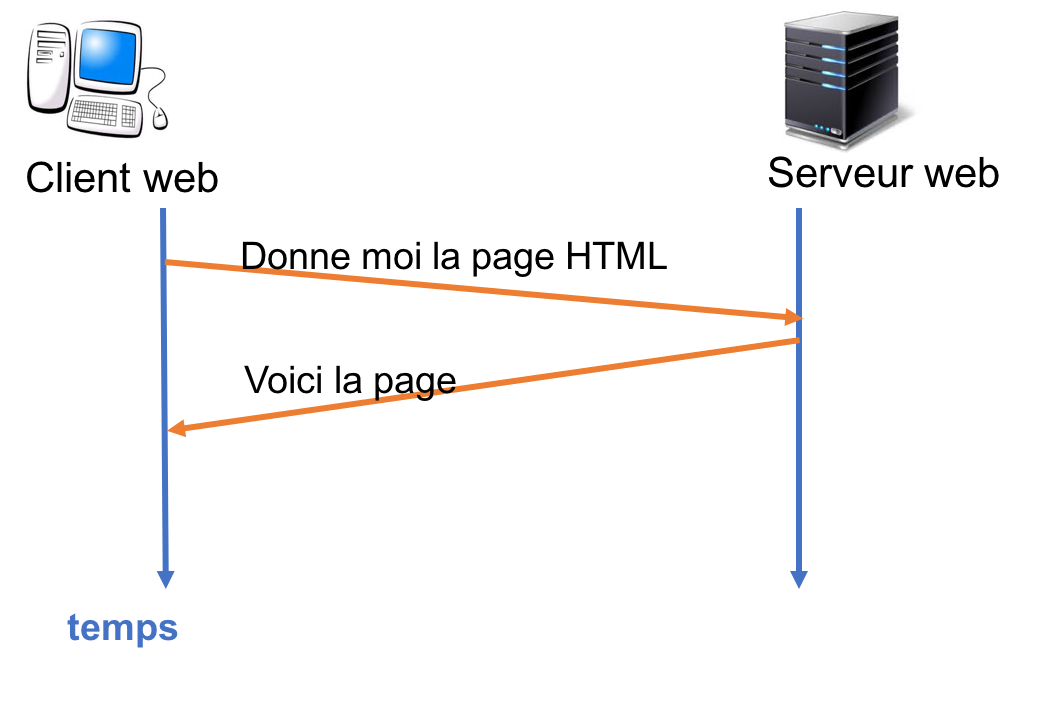
# Introduction

## Client – Serveur

Une image contenant capture d’écran

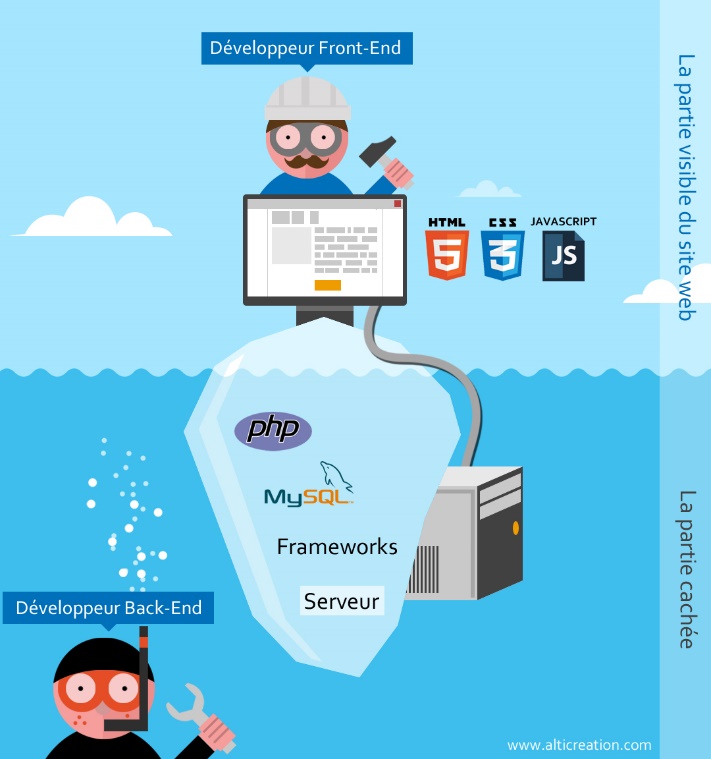
Description générée automatiquement

## Protocole HTTP



* Un protocole est ensemble de règles qui permettent à 2 ordinateurs de communiquer ensemble
* **HTTP (HyperText Transfert Protocol)**
* Permettre au client d'effectuer des requêtes à destination d'un serveur web. En retour, le serveur web va envoyer une réponse
* Une requête HTTP utilise une **méthode :**
  + **GET** : C'est la méthode la plus courante pour demander une ressource. Elle est sans effet sur la ressource.
  + **POST** : Cette méthode est utilisée pour soumettre des données en vue d'un traitement (côté serveur). Typiquement c'est la méthode employée lorsque l'on envoie au serveur les données issues d'un formulaire.
  + **DELETE** : Cette méthode permet de supprimer une ressource sur le serveur.
  + **PUT** : Cette méthode permet de modifier une ressource sur le serveur
* Le "**HTTPS**" est la version "**sécurisée**" du protocole HTTP (Données chiffrées)

## Différence entre "client" et "serveur" | entre front-end et back-end

 Une image contenant Visage humain, personne, sourire, homme

Description générée automatiquement

# IHM avec JavaScript

HTML, CSS et JavaScript

JavaScript est un langage de programmation essentiel dans la création d'Interfaces Homme-Machine (IHM) interactives et dynamiques sur le Web. Grâce à ses fonctionnalités, il permet d'améliorer l'expérience utilisateur en rendant les interfaces plus réactives et personnalisées.

Pour résumer:

1. **HTML** pour la structure et le contenu du document
2. **CSS** pour la mise en forme
3. **JavaScript** pour le comportement, l’interactivité => IHM

Le JS va permettre d’agir sur le document web et d’en modifier profondément le contenu et la structure (HTML) ou l’apparence (CSS).

## Intégration dans la page HTML

* Le JS peut être inséré dans la page dans la balise conteneur **<script></script>.**
* Celle-ci peut se mettre dans le header et/ou le body, une ou plusieurs fois.

<html>

<head>

</head>

<body>

**<script>**

**//Code JS**

**</script>**

</body>

</html>

## Intégration dans UNE PAGE JS EXTERNE

On va demander au navigateur d’importer un fichier \*.js en utilisant l’attribut src de la balise script :

**<script type="text/javascript" src="hello.js"></script>**

### Adressage relatif ou absolu (rappel)

* Adressage absolu : le chemin d'accès au fichier en partant de la "racine".

(sous Windows) : **C:\Users\Aryx\Desktop\monFichier.pdf**

(sous linux) /home/toto.txt

* Adressage relatif : le chemin d'accès au fichier par rapport au "répertoire courant"

|  |  |
| --- | --- |
| Une image contenant texte  Description générée automatiquement | Si le répertoire courant est *C:\Programme*  Et que je souhaite atteindre le fichier "**monFichier.pdf"** |
| Chemin Absolu :  C:/Users/Aryx/Desktop/monFichier.pdf |
| Chemin Relatif :  ../Users/Aryx/Desktop/monFichier.pdf |

## Les bases du langage

* Chaque instruction en JS se termine par un ;
* Une bonne pratique est de se limiter à une instruction par ligne.
* Les blocs en JS sont délimités par des accolades { }, et les espaces/tab ne sont pas prises en compte.
* Commentaires mono ligne : // commentaire jusque fin de ligne  
  Commentaires multi lignes : /\* ceci est

un commentaire \*/

### Déclaration de Variables

* **var :** Ancienne méthode, portée globale ou fonction.
* **let :** Portée limitée au bloc. Variable locale.
* **const :** Déclare une constante, non réassignable.

var x = 10; // Variable globale

let y = 20; // Variable locale

const z = 30; // Constante

### Types de Données

* Primitifs : Number, String, Boolean, Undefined, Null, Symbol, BigInt.
* Objets : Array, Object, Function.

let age = 25; // Number

let name = "Alice"; // String

let isActive = true; // Boolean

let fruits = ["Apple", "Banana"]; // Array

let person = { name: "Bob", age: 30 }; // Object

### Les opérateurs logiques

|  |  |  |
| --- | --- | --- |
| Opérateur | Type de logique | Utilisation |
| && | ET | valeur1 && valeur2 |
| || | OU | valeur1 || valeur2 |
| ! | NON | !valeur |

### Interactions avec l’utilisateurs

* La fonction alert("mon texte..") permet d’afficher une boite de dialogue avec du texte donnée en paramètre
* La fonction prompt() permet d’afficher une invite de saisie à l’écran (avec une boîte de dialogue).
  + La fonction retourne un string

let age = prompt('Entrez votre age');

alert(age); // Affiche age

**Exercice** : A l’aide des fonctions alert('texte') et prompt('texte') écrivez un petit script qui permet de calculer votre âge à partir de votre année de naissance

### console.log en JavaScript

La méthode console.log() est une fonction intégrée à JavaScript qui permet d'afficher des messages dans la **console du navigateur (F12 ou inspecter l’élément sur le navigateur)**. C'est l'un des outils les plus couramment utilisés par les développeurs pour le **debugging** ou pour suivre l'état d'exécution d'un programme.

console.log("Bonjour, Monde !");

let name = "Alice";

let age = 25;

console.log("Nom :", name, "| Âge :", age);

### Structures de Contrôle

**Conditions :**

if (x > 5) {

console.log("x est grand");

} else {

console.log("x est petit");

}

**Boucles :**

for (let i = 0; i < 5; i++) {

console.log(i); // Affiche 0 à 4

}

let j = 0;

while (j < 5) {

console.log(j);

j++;

}

**Fonctions :**

// Déclaration

function addition(a, b) {

return a + b;

}

console.log(addition(5, 3)); // 8

## Manipulation DOM

* Le **DOM** (Document Object Model) est une interface de programmation qui représente la structure d'une page web sous la forme d'une arborescence hiérarchique. Il permet aux développeurs d'accéder et de manipuler dynamiquement les éléments HTML et CSS d'un document via **JavaScript**.
* On va pouvoir accéder aux éléments de la page HTML pour éventuellement les modifier.
* Il existe de nombreuse méthode pour le faire dont :
  + **document.getElementById("ID")**
  + va chercher la balise d'identificateur **<balise id="ID" >** dans le code
* Une alternative, utiliser les sélecteurs CSS :
  + **document.querySelector('selecteurCSS')**

### Sélection d’Eléments

let element1 = document.getElementById("myId");

let element2 = document.querySelector("#myId");

let elements = document.querySelectorAll(".myClass");

### Modification du DOM

Quand on a accédé à un élément, on va pouvoir le modifier en indiquant quelle nouvelle valeur donner à l'attribut.

element1.textContent = "Nouveau texte";

document.querySelector('p').innerHTML = "nouveau texte"

document.querySelector('#img27').src = "nouvelleValeur.jpg"

element1.style.color = "red";

document.getElementById("*td"*).style.*background*=*"red"*

var element = document.getElementById("myDIV");

element.classList.add("mystyle");

### Ajout d'Événements

* On va pouvoir déclencher du JS sur la base d'évènements qui se produisent sur la page

**En HTML :**

<button onclick="**alert('yes you can')**" >cliquez Moi</button>

**En JS :**

element.addEventListener("click", () => {

alert("Élément cliqué !");

});

Liste des “events"

|  |  |
| --- | --- |
| **Event** | **Description** |
| onchange | An HTML element has been changed |
| onclick | The user clicks an HTML element |
| onmouseover | The user moves the mouse over an HTML element |
| onmouseout | The user moves the mouse away from an HTML element |
| onkeydown | The user pushes a keyboard key |
| onload | The browser has finished loading the page |

## Les boucles pour construire un tableau HTML

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

<table id="t1"></table>

<script>

      let html = "";

      for (let i = 0; i < 5; i++) {

        html += "<tr>";

                for (let j = 0; j < 5; j++) {

              html += '<td><img src="incredible.PNG"></td>';

          }

        html += "</tr>"

      }

      document.querySelector("#t1").innerHTML += html;

</script>

Autre exemple :

<table id="table1"></table>

<button onclick="genererTable()">Générer le tableau</button>

<script type="text/javascript">  
      function genererTable(){

        let table1 = document.querySelector("#table1");

        let html = "";

        let cpt = 1;

        let i = 0;

        for (i; i<5; i++) {

          html += "<tr>";

            for(let j = 0; j<5; j++){

              html += "<td id=" + cpt + ">" + cpt + "</td>";

              cpt++;

            }

          html += "</tr>";

        }

        table1.innerHTML = html;

        document.getElementById("4").innerHTML = "TEST";

      }

</script>

Résultat :

Une image contenant capture d’écran, texte, ligne, Parallèle

Description générée automatiquement

# TP1 : Bulb generator v1.0

Réalisez une page Web comportant l'image d'une ampoule, une légende sur son état, et deux boutons.

=> Si on clique sur le bouton "Allumée" l'ampoule doit être allumée et la légende associée doit l'indiquer.

=> Si on clique sur le bouton "Eteinte" l'ampoule doit être éteinte et la légende associée doit l'indiquer.

# TP2 : Nombre Mystrère v2.0

* Réalisez "trouver un nombre mystère" avec HTML, CSS et JS.
* Le principe est particulièrement simple : un nombre mystère est généré de manière aléatoire entre 0 et 100,
* L'utilisateur lui voit un tableau de 100 cases. Il clique sur ses propositions successives pour se rapprocher et trouver le nombre recherché. Pour chaque tentative, l'affichage est modifié pour lui indiquer si le nombre proposé est trop petit ou trop grand.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 |  |  |  |  |  |  |  |  |
|  |  |  | 14 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | 48 |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | 99 | 100 |

# Mémo visuel

Une image contenant texte, capture d’écran, Police, Page web

Description générée automatiquementUne image contenant texte, capture d’écran, Police, diagramme

Description générée automatiquement

Une image contenant texte, clipart, dessin humoristique, illustration

Description générée automatiquement