Développement web côté client

HTML/CSS/JavaScript

Pr. Abdelali El Gourari

Introduction

- **Développement web**
  - ***"…une discipline qui consiste, par l'usage de langages de programmation web, à programmer des sites web ou applications web (ou web mobile) destinés à être publiés…"***
  - ***"… processus d'écriture d'un site ou d'une page web dans un langage technique…"***
  - ***"…. Le processus de développement web comprend, entre autres, la conception de sites web, le développement de contenu web, l'élaboration de scripts côté client ou côté serveur…"***
  - **La science de traitement et de présentation de l'information**

Introduction

- **Modèle client/serveur**
  - **Serveur de fichiers/d'exécutables**
    - Partage de données
    - Diffusion des exécutables des applications partagées
  - **Serveur de configurations**
    - Diffusion des paramètres de configuration partagés
  - **Serveur d'applications**
    - Serveur de calcul
    - Serveur web
    - Software-as-a-service (SaaS, cloud)

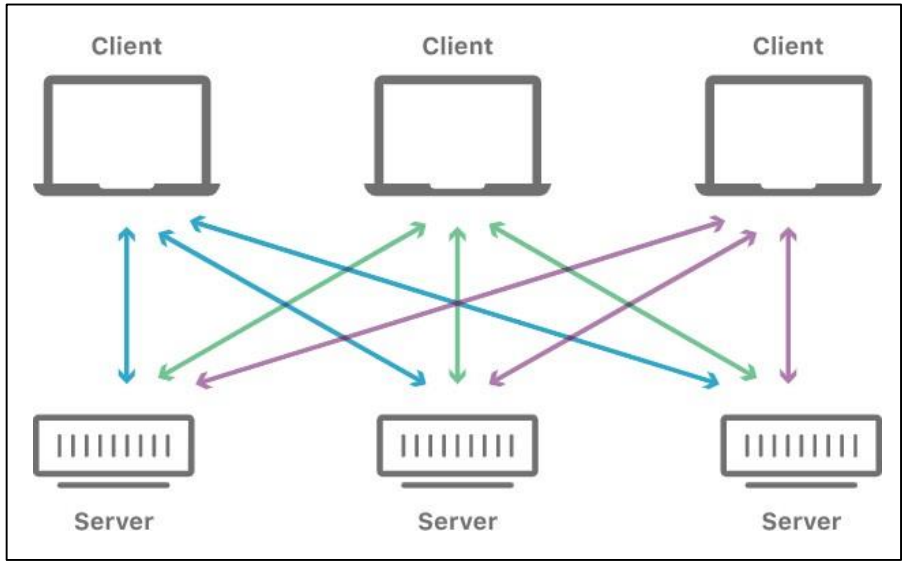
Introduction

- **Modèle client/serveur**
  - **Intérêts**
    - Installation et mise à jours des applications au niveau des serveurs
    - Cohérence entre postes: OS, configurations de base, logiciels
    - Sécurité
  - **Inconvénients et limites**
    - Performances
    - Risque d’indisponibilité générale (pannes serveur ou réseau)
    - Gestion des configurations spécifiques
    - Sécurité

Introduction

- **Côté client**

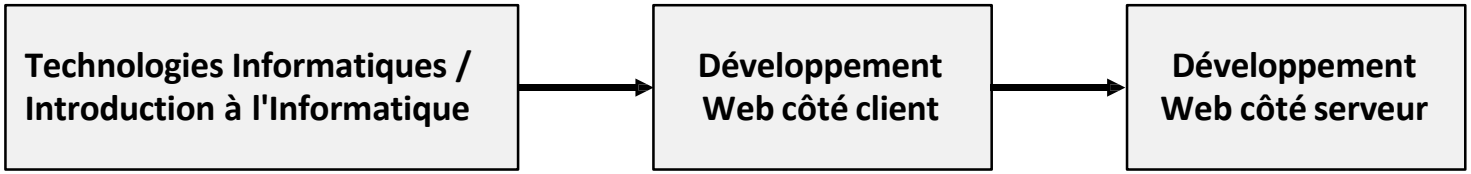
https://www.cloudflare.com/



  - **Tout ce qui s'affiche ou se déroule sur le client (appareil de l'utilisateur).**
    - Le contenu de la page - HTML
    - La mise en page – CSS
    - Le comportement - JavaScript

Introduction

- **Côté serveur**
  - **à voir plus tard…**



Structure minimale d'une page HTML5

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Ma première page</title>
  </head>

  <body>
    Contenu de ma première page.
  </body>
</html>
```

Les balises

- **HTML est un langage à balises.**
  - **Une balise est ce qui est entouré de chevrons < et >**
    - Les **conteneurs** : <x attribut="valeur">contenu</x>
      - <x> : balise ouvrante
      - </x> : balise fermante
    - Les **marqueurs** : <x attribut="valeur"**/**>
      - Le "/" n'est pas obligatoire mais permet de distinguer un marqueur d'une balise ouvrante
  - **Aux balises s'ajoutent des attributs qui les complètent en fournissant des informations supplémentaires**
    - <balise attribut="valeur">
    -

Les balises

- **Rappel**
  - <!DOCTYPE html>
  - <html></html>
  - <head></head>
  - <body></body>
  - <meta charset="utf-8" />
  - <title></title>
  -
  - <hr/>
  -
  - <h1></h1>
  - <h2></h2>
  -
  -
  - <mark> </mark>
  - <blockquote> </blockquote>
  -
  -
  -
  -
  - <table></table>
  - <tr><tr>
  - <td><td>
  - <a>
  - <img>

CSS - Cascading Style Sheets

- **Permet de mettre en forme le code HTML**
  - **Choix de couleur de texte, sélection de la police de caractères, de la taille de la police, de l'image de fond...**
  - **Mise en page du site : menu à gauche en vertical, en-tête en haut toujours visible...**
- **Le CSS s'écrit au choix**
  - **Dans un fichier .css**
  - **Dans l'en-tête <head></head>**
  - **Dans les balises du HTML via l'attribut**

Structure d'un fichier CSS

- **Un fichier CSS contient des règles CSS**
  - **Les noms (sélecteurs) des balises dont on souhaite modifier l'aspect.**
  - **Des propriétés CSS**
  - **Une valeur pour chaque propriété**

```
sélecteur {
        propriété : valeur;
        propriété : valeur;
}
```

```
h1
{
    color: green;
    font-size: 20pt;
    font-weight: bold;
}
p
{
    color: red;
}
em, strong
{
     font-weight: bold;
}
```

Distinguer les balises : class et id

- **Exemple**
  - **On veut que seulement certains paragraphes soient mis sous une certaine forme**
  - **Plutôt que de mettre un attribut style à chacune des balises concernée on peut utiliser les attributs suivants valables sur n'importe quelle balise**
    - class
    - id

```
<h1 class="classe1"> </h1>
<p class="classe1"> </p>
```

```
.classe1
{
        color: red;
}
```

```
<p id="ceParagraphe"></p>
```

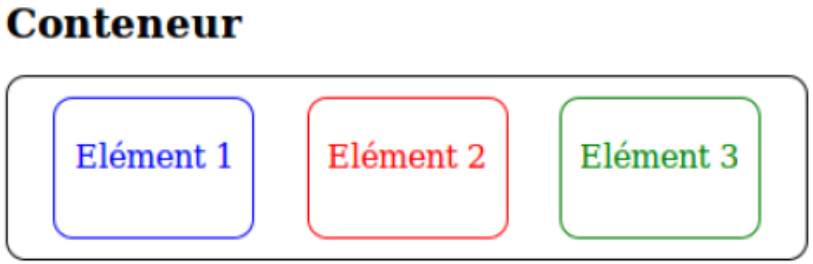
```
#ceParagraphe
{
        color: red;
}
```

Les propriétés CSS

- **Rappel**
  - Taille du texte
  - Police
  - Mise en forme du texte
  - Alignement
  - Couleur du texte
  - Couleur de fond
  - Image de fond
  - La transparence
  - Les bordures
  - Changement d'apparence dynamique au survol, au click
  - Taille de bloc
  - Les marges

Flexbox

Avec Flexbox une page est vue comme un conteneur qui contient plusieurs éléments.
Dans une page Web on peut avoir plusieurs conteneurs contenant eux-mêmes d'autres conteneurs.



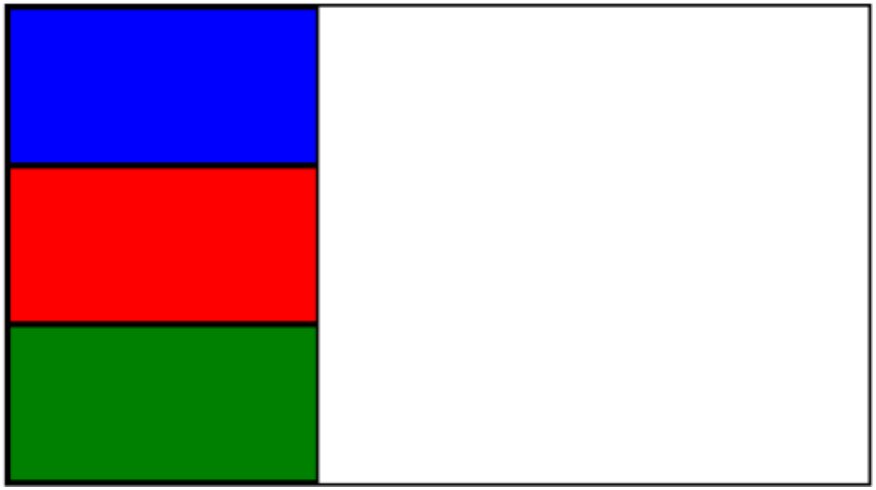
En HTML un conteneur sera une balise et chaque élément également (en général un `div`) :

```
<div id="conteneur">
<div id="element1"></div>
<div id="element2"></div>
<div id="element3"></div>
</div>
```

Flexbox

Par défaut les éléments se mettent les uns au-dessus des autres :

```
#conteneur
{
    border : 1px black solid;
}
```



Flexbox

Propriété `display : flex;`

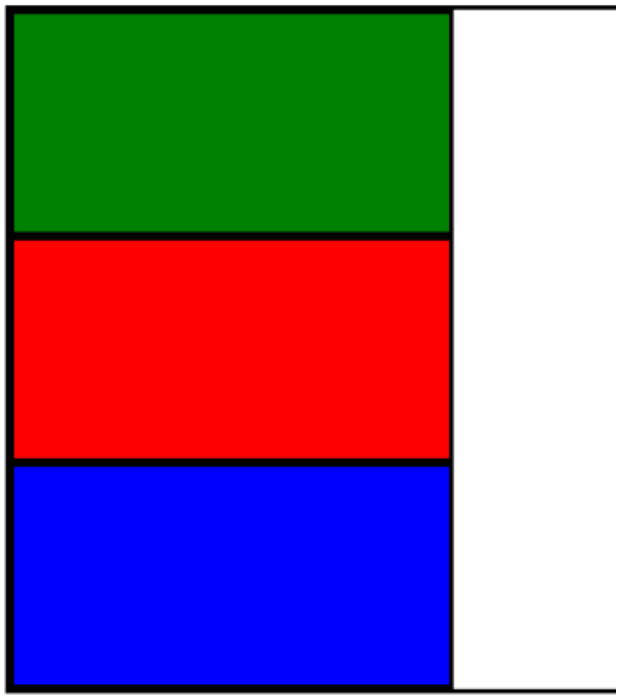
```
#conteneur
{
    border : 1px black solid;

    display : flex;
}
```

Les éléments se placent par défaut en ligne !

Flexbox

Propriété `flex-direction` qui peut prendre les valeurs :

- `row` en ligne (par défaut)
- `column` en colonne
- `row-reverse` en ligne inversés
- `column-reverse` en colonne renversés



`column-reverse`

On peut changer l'ordre des blocs sans modifier le HTML !

Flexbox

La propriété `flex-wrap` peut prendre les valeurs :

`no-wrap`

`wrap`

`wrap-reverse`

`no-wrap` a écrasé chaque bloc pour qu'ils puissent rentrer dans le conteneur.

Aligner sur l'axe principal

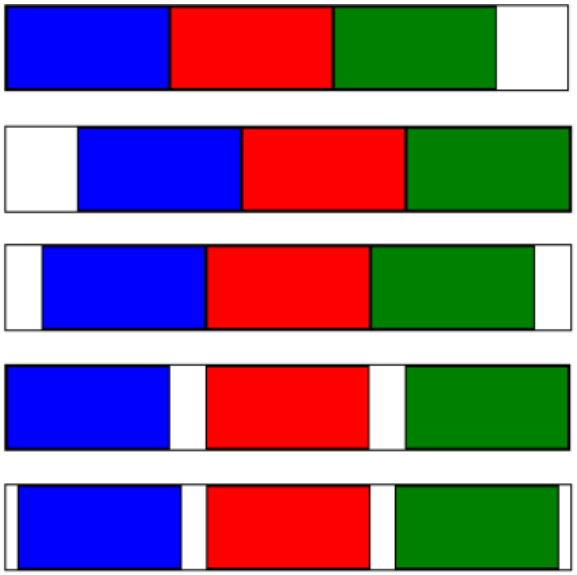
L'organisation des éléments avec Flexbox se déclare soit horizontalement (par défaut) soit verticalement.
Cette déclaration définit l'axe principal.
L'axe secondaire (cross axis) est alors l'autre axe perpendiculaire à l'axe principal. Il est donc :

- vertical si l'axe principal est horizontal
- horizontal si l'axe principal est vertical

Pour aligner sur l'axe principal, utiliser `justify-content` dont les valeurs possibles sont :

- `flex-start` : éléments alignés au début (par défaut)
- `flex-end` : éléments alignés à la fin
- `center` : éléments alignés au centre
- `space-between` : éléments répartis sur tout l'axe
- `space-around` : éléments répartis sur tout l'axe avec espace autour de chacun

Aligner sur l'axe principal



flex-start

flex-end

center

space-between

space-around

```
#conteneur
{
    border : 1px black solid;
    width : 350px;
    display : flex;
    flex-direction : raw;
    justify-content : space-around;
}
```

On peut évidemment le faire aussi verticalement :

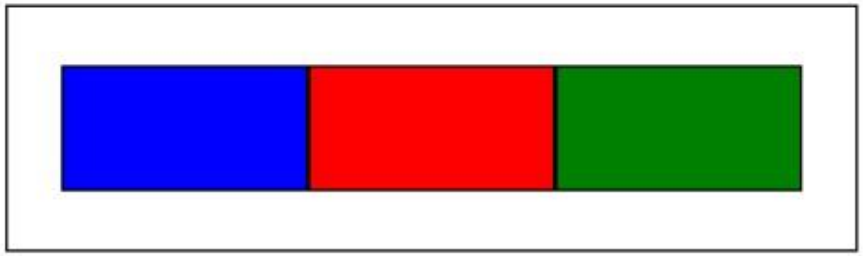
`flex-direction : column;`

Aligner sur l'axe secondaire

On utilise pour cela la propriété `align-items` qui peut prendre les valeurs :

- `stretch` : les éléments s'étendent sur tout l'axe (par défaut)
- `flex-start` : alignés au début
- `flex-end` : alignés à la fin
- `center` : alignés au centre

```
#conteneur
{
    border : 1px black solid;
    width : 350px;
    height : 100px;
    display : flex;
    flex-direction : raw;
    justify-content : center;
    align-items : center;
}
```



JavaScript

Introduction

Nous avons fait des pages statiques en L1: mise à part quelques effets (survol de souris, sélection ...) le fond et la forme sont prédéterminés et sont chargés par le client pour être affichés tels quells.

- Avec JavaScript on peut modifier l'apparence dynamiquement, par exemple changer la couleur de fond sur le survol d'un paragraphe particulier
- JavaScript s'effectue côté client. Son code est du coup intégré dans un fichier du site et est interprété par le navigateur
- JavaScript est un langage de programmation (au même titre que VB) qui n'a rien à voir avec Java
- JavaScript peut être intégré directement dans un fichier HTML
  ou être écrit dans un fichier indépendant d'extension .js

Introduction

- **JavaScript (JS)**
  - **A été initialement créé pour “rendre les pages web vivantes”**
    - Côté client, dans un navigateur web
  - **scripts : peuvent être écrits directement dans une page HTML et exécutés automatiquement au chargement des pages**
    - Pas besoin d'une compilation pour fonctionner
  - **Aujourd'hui**
    - Framework : Bootstrap, jquery, vue.js
    - côté serveur : node.js
    - tensorflow.js : machine learning, AI
    - …
- **Dans cette UE**
  - **Côté client, dans un navigateur web**
    - Référence : https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference (principal manuel avec des exemples et d’autres informations)
  - **Framework : Bootstrap**

Développement JS : éditeurs de texte

- **Visual Studio Code**
  - A utiliser si vous avez pas d'un éditeur de texte préféré
  - Cross-platform : Windows, Linux, et macOS
  - Gratuit, leger
- **Les autres**
  - nano, vi, notepad++, sublime text, visual studio…
  - … utilisez à vos risques et périls ☺

Ajouter du code JavaScript à la page

- **Les balises <script></script>**
  - **Peuvent être insérés dans n'importe quelle partie d'un document HTML**

```
<!DOCTYPE HTML>
<html>
<body>
    <p>Voici mon script</p>
    <script>
    alert( 'Hello, world!' );
    </script>
</body>
</html>
```

- **Scripts externes**
  - **Placer le code JS dans un fichier séparé**
  - **Le fichier de script est attaché à HTML avec l'attribut src**
    - <script src="../chemin/vers/votre/script1.js"></script>
    - <script src="../chemin/vers/votre/script2.js"></script>

Structure du code

- **Chaque instruction est généralement écrite sur une ligne distincte terminé par un point virgule**
  - Les points virgules peuvent être omis dans la plupart des cas mais **ne faites pas ça…**
- **Commentaires**
  - **sur une ligne : // (deux barres obliques)**
  - **multi lignes : /* … */**
    - commencé par **/***
    - terminée par ***/**

Les variables

- **Déclarer une variable**
  - **`let nom_de_variable`**
    - Déclaration de variable moderne
  - **`var nom_de_variable`**
    - similaire à let mais n'a pas de portée limitée aux blocs
    - mais la visibilité est étendue à la fonction ou globale (si la variable est déclarée hors fonction)
  - **`const nom_de_variable`**
    - Déclarer une constante
- **Nom de variable**
  - **Le nom ne doit contenir que des lettres, des chiffres, des symboles $ et _**
  - **Le premier caractère ne doit pas être un chiffre**
  - **La casse est importante**
  - **Les mots réservés (`let, class, return, function`) ne peuvent pas être utilisés**

Les types de données

number	let n = 123; n = 12.345;
bigint	let n = 12345678901234567890123456789012345678901234567890n
string	let str = "Hello";
boolean	let drapeau = true
null	let n = null;
undefined	let n;
object	à avoir plus tard
symbol	

Interaction

- **alert**
  - affiche un message

```
alert("Hello");
```

- **prompt**
  - affiche un message demandant à l'utilisateur de saisir du texte. Il renvoie le texte ou null, si vous cliquez sur le bouton Annuler/Esc

```
let n = prompt("Entrez un entier", 5);
```

- **confirm**
  - affiche un message et attend que l'utilisateur appuie sur "OK" ou "Annuler"
  - retourne true pour OK et false pour Annuler/Esc

```
let ok = confirm("OK ?");
```

Opérateurs

Addition	+
Soustraction	-
Multiplication	*
Division	/
Modulo	%
Exponentiation	**
Affectation	=
Incrémentation	++
Décrémentation	--
Binaire	AND(&), OR(|), XOR(^), NOT(~)
Comparaison	>, <, =>, <=, ==, ===, !=

Conditionnelle et boucles

```
if (condition) {
    //instructions
}
else {
    //instructions
}
```

```
let i = 2
if (i < 3) {
    alert(i);
}
```

```
while (condition) {
    //instructions
}
```

```
let i = 3;
while (i) {
    alert(i);
    i--;
}
```

```
do {
    //instructions
} while (condition)
```

```
for (début; condition; étape)
{
    //instructions
}
```

```
for (let i = 0; i < 3; i++) {
        alert(i);
}
```

Fonctions

Les fonctions sont un moyen de compacter des fonctionnalités en vue de leur **réutilisation**. Quand vous avez besoin de la procédure, vous pouvez **appeler une fonction**, par son nom, ***au lieu de réécrire la totalité du code chaque fois***. Nous avons déjà utilisé des fonctions plus haut, par exemple :

```
let myVariable = document.querySelector("h1");

alert("Bonjour !");
```

Ces fonctions (**querySelector** et **alert**) sont disponibles dans le navigateur et vous pouvez les utiliser où bon vous semble.

Si vous voyez quelque chose qui ressemble à un nom de variable et qui est suivi de parenthèses — **()** —, c'est probablement une fonction. Les fonctions prennent souvent des arguments — bouts de données dont elles ont besoin pour faire leur travail. Ils sont placés entre parenthèses, séparés par des virgules s'il y en a plusieurs.

Fonctions

Par exemple, la fonction **alert()** fait apparaître une fenêtre de pop-up dans la fenêtre du navigateur, mais vous devez donner une chaîne comme argument pour indiquer à la fonction ce que l'on souhaite écrire dans cette fenêtre.

La bonne nouvelle est que vous pouvez définir vos propres fonctions — par exemple, vous pouvez écrire une fonction toute simple qui prend deux arguments et les multiplie :

```
function multiply(num1, num2) {
  let result = num1 * num2;
  return result;
}
```

Vous pouvez déclarer cette fonction dans la console avant de l'utiliser plusieurs fois :

```
multiply(4, 7);
multiply(20, 20);
multiply(0.5, 3);
```

Fonctions

Expression de fonction :

```
// Déclaration d'une fonction anonyme et assignation à une variable

let multiply = function(a, b) {
return a * b;
};

// Utilisation de la fonction multiply

console.log(multiply(2, 3)); // Affichera 6
```

Fonctions

Fonction fléchée (Arrow Function) :

```
// Déclaration d'une fonction fléchée

let subtract = (a, b) => {
return a - b;
};

// Utilisation de la fonction subtract

console.log(subtract(5, 3)); // Affichera 2
```

Fonctions

Fonction avec paramètres par défaut :

```
// Déclaration d'une fonction avec paramètres par défaut

function greet(name = 'World') {
return `Hello, ${name}!`;
}

// Utilisation de la fonction greet

console.log(greet());                // Affichera "Hello, World!"
console.log(greet('Ali'));           // Affichera "Hello, Ali!"
```

Fonctions

Fonction prenant une fonction comme argument (fonction de rappel) :

```
// Déclaration d'une fonction qui prend une fonction en argument
function operate(a, b, operation) {
return operation(a, b);
}

// Déclaration de fonctions à utiliser comme opérations

function add(a, b) { return a + b; }
function multiply(a, b) {
return a * b;
}

// Utilisation de la fonction operate avec différentes opérations
console.log(operate(2, 3, add));              // Affichera 5
console.log(operate(2, 3, multiply));         // Affichera 6
```

Fonctions

Fonction retournant une fonction (fonction génératrice) :

```
// Déclaration d'une fonction retournant une fonction

function makeMultiplier(factor) {
return function(number) {
return number * factor;
};
}
// Création d'une fonction de multiplication par 5

let multiplyBy5 = makeMultiplier(5);

// Utilisation de la fonction multiplyBy5

console.log(multiplyBy5(3));       // Affichera 15
console.log(multiplyBy5(4));       // Affichera 20
```

Fonctions

Exercice_1:
Écrivez une fonction nommée ***calculateAverage*** qui prend en paramètre un tableau de nombres et retourne la moyenne de ces nombres.

Exercice_2:
Écrivez une fonction nommée ***countVowels*** qui prend en paramètre une chaîne de caractères et retourne le nombre de voyelles qu'elle contient.

Exercice_3:
Écrivez deux fonctions : ***celsiusToFahrenheit*** qui convertit une température en degrés Celsius en degrés Fahrenheit, et ***fahrenheitToCelsius*** qui fait l'inverse. Les formules de conversion sont :

Celsius en Fahrenheit : (°C × 9/5) + 32
Fahrenheit en Celsius : (°F - 32) × 5/9

Fonctions

Exercice_4:

Écrivez une fonction nommée ***validatePassword*** qui prend en paramètre un mot de passe sous forme de chaîne de caractères et retourne true si le mot de passe est valide selon les critères suivants :

Doit contenir au moins 8 caractères.
Doit contenir au moins une lettre majuscule et une lettre minuscule.
Doit contenir au moins un chiffre.

Exercice_5:

Écrivez une fonction nommée isPalindrome qui prend en paramètre une chaîne de caractères et retourne true si elle est un palindrome (c'est-à-dire qu'elle se lit de la même manière de gauche à droite et de droite à gauche), et false sinon.

Fonctions

```
function calculateAverage(numbers) {
        const sum = numbers.reduce((total, num) => total + num, 0);
        return sum / numbers.length;
}
```

```
function countVowels(str) {
const vowels = ['a', 'e', 'i', 'o', 'u'];
let count = 0;
for (let char of str.toLowerCase()) {
if (vowels.includes(char)) {
count++;
}
}
return count;
}
```

Fonctions

```
function celsiusToFahrenheit(celsius) {
return (celsius * 9/5) + 32;
}
function fahrenheitToCelsius(fahrenheit) {
return (fahrenheit - 32) * 5/9;
}
```

```
function validatePassword(password) {

const hasUpperCase = /[A-Z]/.test(password);
const hasLowerCase = /[a-z]/.test(password);
const hasDigit = /\d/.test(password);

return password.length >= 8 && hasUpperCase && hasLowerCase && hasDigit;
}
```

Fonctions

```
function sortArray(numbers) {
return numbers.sort((a, b) => a - b);
}
```

```
function isPalindrome(str) {
const reversedStr = str.split('').reverse().join('');
return str === reversedStr;
}
```

Tableaux (Arrays) :

Déclaration d'un tableau :

```
let numbers = [1, 2, 3, 4, 5];
let fruits = ['apple', 'banana', 'orange'];
```

Accès aux éléments d'un tableau :

```
console.log(numbers[0]); // Affiche le premier élément du tableau numbers (1)
console.log(fruits.length); // Affiche la longueur du tableau fruits (3)
```

Modification des éléments d'un tableau :

```
fruits[1] = 'grape'; // Remplace 'banana' par 'grape'
numbers.push(6); // Ajoute 6 à la fin du tableau numbers
```

Tableaux (Arrays) :

Parcours d'un tableau :

```
for (let i = 0; i < numbers.length; i++) {
console.log(numbers[i]);
}
```

Méthodes de tableau :

```
numbers.pop(); // Supprime le dernier élément du tableau numbers
```

Objets (Objects) :

Déclaration d'un objet :

```
let person = {
            name: ‘Ali',
            age: 27,
            city: ‘Maroc‘
 };

let car = {
            brand: 'Toyota',
            model: 'Camry',
            year: 2024
};
```

Accès aux propriétés d'un objet :

```
console.log(person.name); // Affiche la propriété 'name' de l'objet person (‘Ali')

console.log(car['model']); // Affiche la propriété 'model' de l'objet car ('Camry')
```

Objets (Objects) :

Modification des propriétés d'un objet :

```
person.age = 35; // Modifie la valeur de la propriété 'age' de l'objet person

car.color = 'blue'; // Ajoute une nouvelle propriété 'color' à l'objet car
```

Parcours des propriétés d'un objet :

```
for (let key in person) {

console.log(`${key}: ${person[key]}`);
}
```

Objets (Objects) :

Méthodes d'objet :

```
delete person.city; // Supprime la propriété 'city' de l'objet person

let keys = Object.keys(car); // Retourne un tableau contenant les clés de l'objet car

let values = Object.values(person); // Retourne un tableau contenant les valeurs de l'objet person
```

Objets (Objects) :

Dans cet exemple, **livre** est un objet avec les propriétés **titre** et **auteur**, ainsi qu'une méthode **afficherInfo** qui affiche ces informations dans la console.

```
// Définition de l'objet Livre

const livre = {

        titre: 'JavaScript for Beginners',
        auteur: ‘Ali',
        afficherInfo: function() {
                console.log(`Titre: ${this.titre}, Auteur: ${this.auteur}`);
        }
};

// Utilisation de la méthode afficherInfo

livre.afficherInfo(); // Affiche "Titre: JavaScript for Beginners, Auteur: Ali"
```

EVENEMENTS ET OBJETS

1. Programmation événementielle
2. Principe
3. Les événements
4. Gestionnaire d'événement
5. Classe et Objet
6. Objets d'une page Web
7. Objets divers

1 . Programmation événementielle

✓ La programmation événementielle permet la **gestion d' événements**.

✓ Un événement est généralement associé à une action de l'utilisateur : **appui sur une touche**, **click ou déplacement de la souris**, …

✓ En HTML classique, il y a très peu d'événements qui sont gérés : **click sur un lien** ou sur **un bouton de formulaire**. Leur gestion est automatique et non modifiable.

✓ Le **javascript** va permettre de **gérer** et **contrôler** d'autres **événements**.

2 . Principe

L'objet	Événement --------->	Gestionnaire d'événement
(bouton)	(clic de souris)	(envoi du formulaire)
OBJECT	**EVENT**	**EVENT HANDLER**

➢ En javascript, on va pouvoir associer à chaque **événement** une **action** : une fonction, c- à -d le gestionnaire d'événement (**EVENT HANDLER**).

3 . Les événements

➢ Différents événements implémentés en Javascript :

L'utilisateur clique sur un bouton, un lien ou tout autre élément : **Clik**
La page est chargée par le navigateur : **Load**
L'utilisateur quitte la page : **Unload**
L'utilisateur place le pointeur de la souris sur un élément : **MouseOver**
Le pointeur de la souris quitte un lien ou tout autre élément : **MouseOut**
Un élément de formulaire a le focus (devient la zone d'entrée active) : **Focus**
Un élément de formulaire perd le focus : **Blur**
La valeur d'un champ de formulaire est modifiée : **Change**
L'utilisateur sélectionne un champ dans un élément de formulaire : **Select**
L'utilisateur clique sur le bouton submit pour envoyer un formulaire : **Submit**
L'utilisateur appuie sur une touche : **Keydown**

Autres événements : **Abort, Error, Move, Resize, KeyPress, KeyUp, DblClick, MouseDown, MouseUp, MouseMove, Reset**

5 . Classes et Objet

Ne pas confondre :

- Javascript est un langage basé sur les objets
- C++, Java et C# sont des langages orientés objets

Notions de base :

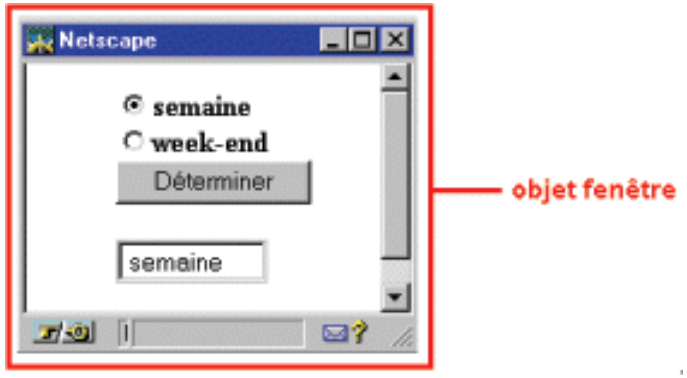
- Une classe est la description d'un **ensemble** de :
  - **propriétés** (les données) et de
  - **méthodes** (les fonctions).

- Un objet est une **instance** de classe (c'est à dire une variable de type classe).

6 . Objets d'une page Web

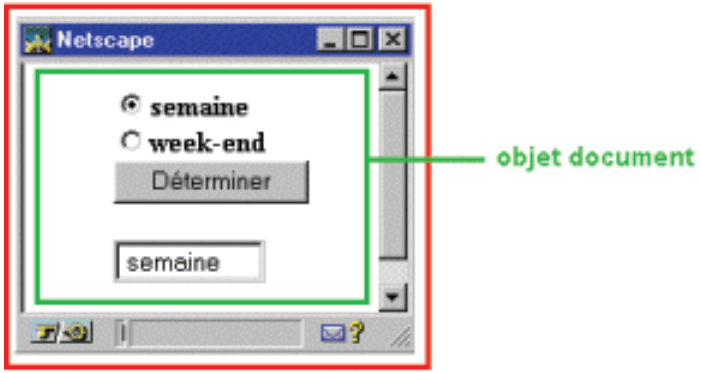
(I)

- Lorsqu'une page Web est chargée dans un navigateur, **Javascript identifie plusieurs objets** pour y représenter les informations.
- Ces objets sont classés de manière hiérarchique.
- L'objet le plus haut dans la hiérarchie est l'objet de la classe **`window`** (fenêtre).

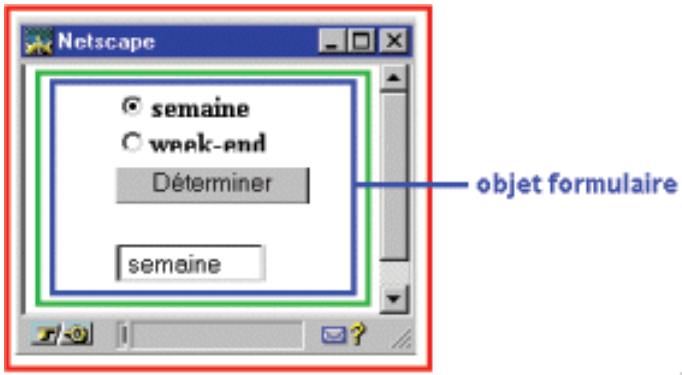


6 . Objets d'une page Web

➢ Dans cette fenêtre, il y a un document HTML : c'est l'objet **document**. Donc, L'objet fenêtre contient l'objet document (c'est la notion de hiérarchie).

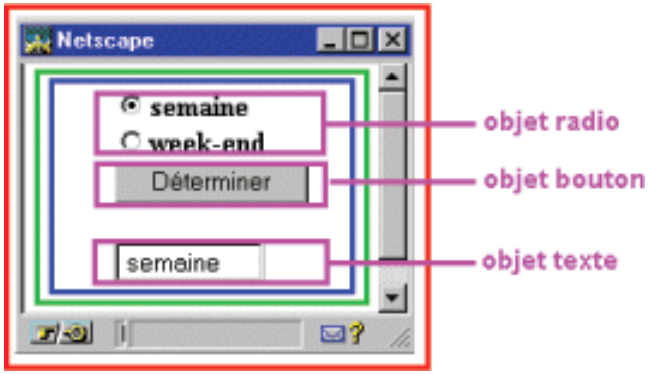


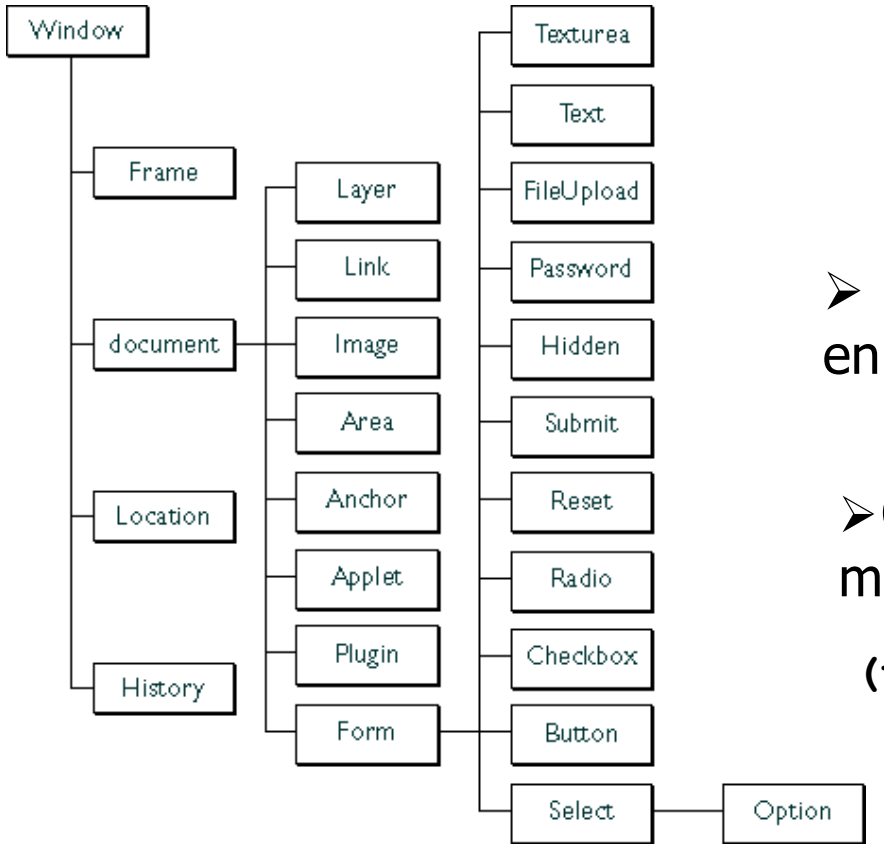
➢ Dans ce document, on trouve un formulaire au sens HTML : c'est l'objet formulaire. Donc, l'objet fenêtre contient un objet document qui lui contient un objet formulaire (hiérarchie des objets).



6 . Objets d'une page Web

Dans ce document, on trouve trois objets : l'objet radio, l'objet bouton, et l'objet texte. Donc, l'objet fenêtre contient…





- Le modèle hiérarchique des objets en Javascript.

- On accèdera à l'objet bouton de la manière suivante :

```
(window).document.form.button
```

7 . Objets divers

- Pour l'instant, on peut considérer la notion de **classe** comme la généralisation de la notion de **type**.

- Par conséquent, nous retrouvons naturellement, parmi les classes d'objets prédéfinies dans le langage Javascript, les classes suivantes :
  - Boolean : les booléens
  - Number : les valeurs numériques (entiers ou réels)
  - String : les chaînes de caractères
  - Array : les tableaux

- D'autres classes prédéfinies et souvent utilisées existent :
  - Date : les dates
  - Math : formules mathématiques
  - Navigator : caractéristiques du navigateur
  - RegExp : les expressions régulières

Recherches: getElement*

- **getElementById()**
  - Si un élément a l'attribut **id**, on peut atteindre cet élément en utilisant la méthode:

```
<input type="text" id="fname" value="Aabecede">

<script>
    let fname = document.getElementById("fname");
    alert("Hello " + fname.value);
</script>
```

```
<div id="divH">
</div>

<script>
    let divH = document.getElementById("divH");
    divH.style.background = "red";
</script>
```

3 . Les événements

Les événements peuvent être associés à des éléments HTML pour déclencher des actions en réponse à des interactions de l'utilisateur. Voici un exemple de code qui ajoute un événement de clic à un bouton :

```
<!DOCTYPE html>
<html>
  <head>
        <title> Événements en JavaScript </title>
        <script scr="./">  </script>

  </head>
<body>
        <button id="monBouton"> Cliquez ici </button>
</body>
</html>
```

4 . Gestionnaire d'événement

```
<script>

// Sélection du bouton
        const bouton = document.getElementById('monBouton');
// Ajout de l'événement de clic
        bouton.addEventListener('click', function() {
        alert('Le bouton a été cliqué !');
        });

</script>
```

Dans cet exemple, un événement de clic est ajouté au bouton avec l'aide de '**addEventListener()'.** Lorsque l'utilisateur clique sur le bouton, une boîte de dialogue d'alerte affiche le message "Le bouton a été cliqué !".

Recherches: getElement*

- **getElementsByTagName(tag)**
  - **cherche les éléments avec la balise donné et renvoie l'ensemble de ces éléments**
- **getElementsByClassName(className)**
  - **renvoie les éléments qui ont la classe CSS donnée**
- **getElementsByName(name)**
  - **renvoie les éléments qui ont l'attribut name, dans tout le document.**

Propriétés et attributs de nœud

- **Différence ?**
  - **Les propriétés – sont ce qui se trouve dans les objets DOM.**
  - **Les attributs – sont ce qui est écrit en HTML.**

Propriétés	Attributs
nodeType nodeName/tagName innerHTML outerHTML value/data textContent hidden	… à revoir : balises HTML et leurs attributs

Que faire avec JavaScript?

- **Modifier contenu HTML**

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function myFunction() {
            document.getElementById("demo").innerHTML = "Hello JavaScript!"
        }
    </script>
</head>
<body>
    <h2>What Can JavaScript Do?</h2>
    <p id="demo">JavaScript can change HTML content.</p>
    <button type="button" onclick='myFunction()'>Click Me!</button>
</body>
</html>
```

Que faire avec JavaScript?

- **Modifier contenu HTML**

Que faire avec JavaScript?

- **Modifier attributs HTML**

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function turnOn() {
            document.getElementById('myImage').src='pic_bulbon.gif'
        }
        function turnOff() {
            document.getElementById('myImage').src='pic_bulboff.gif'
        }
    </script>
</head>
<body>
    <h2>What Can JavaScript Do?</h2>
    <p>JavaScript can change HTML attribute values.</p>
    <p>In this case JavaScript changes the value of the src (source) attribute of an image.</p>

    <button onclick="turnOn()">Turn on the light</button>
    <img id="myImage" src="pic_bulboff.gif" style="width:100px">
    <button onclick="turnOff()">Turn off the light</button>
</body>
</html>
```

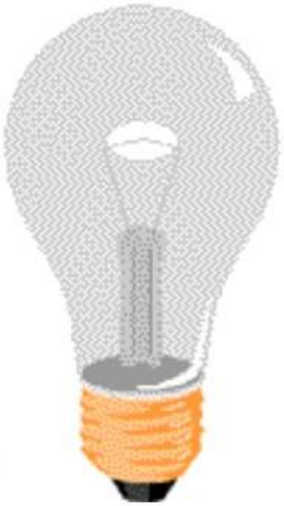
Que faire avec JavaScript?

- **Modifier attributs HTML**

What Can JavaScript Do?

JavaScript can change HTML attribute values.

In this case JavaScript changes the value of the src (source) attribute of an image.



Turn on the light Turn off the light

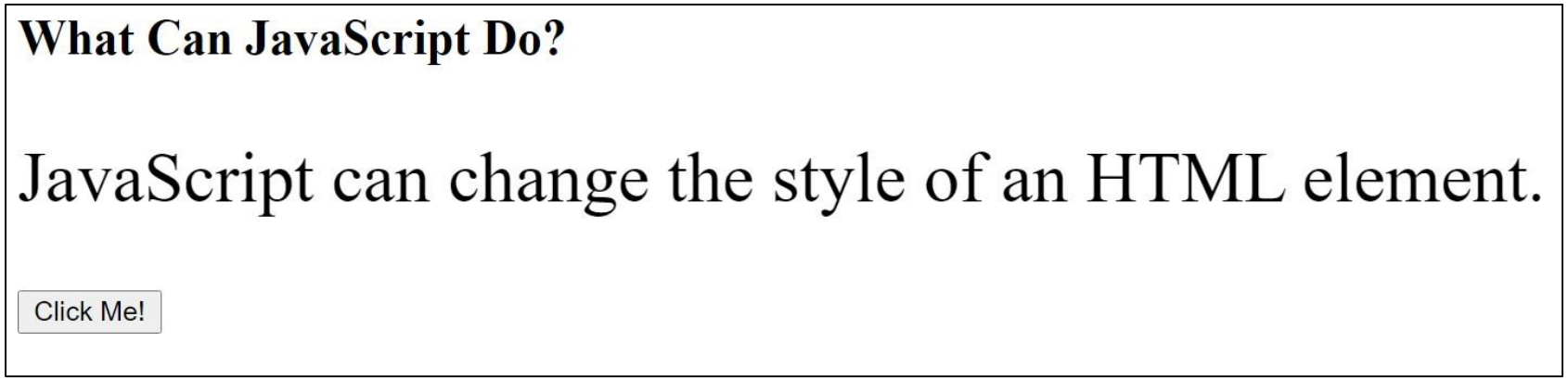
Que faire avec JavaScript?

- **Modifier CSS**

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function changeSize() {
            document.getElementById('demo').style.fontSize='35px'
        }
    </script>
</head>
<body>
    <h2>What Can JavaScript Do?</h2>

    <p id="demo">JavaScript can change the style of an HTML element.</p>

    <button type="button" onclick="changeSize()">Click Me!</button>
</body>
</html>
```

Que faire avec JavaScript?

- **Modifier CSS**



Que faire avec JavaScript?

- **Afficher/cacher éléments HTML**

```html
<!DOCTYPE html>
<html>
<head>
    <script>
        function hideAndShow() {
            var p = document.getElementById('demo');
            var s = p.style.display;
            if(s == 'none') {
                p.style.display = 'block'
            } else {
                p.style.display = 'none'
            }
        }
    </script>
</head>
<body>
    <h2>What Can JavaScript Do?</h2>
    <p id="demo">JavaScript can hide HTML elements.</p>
    <button type="button" onclick="hideAndShow()">Click Me!</button>
</body>
</html>
```

Exercices

Exercice_1

Écrire une application de **liste de tâches** qui permet d'**ajouter**, d'**afficher** et de **supprimer** des tâches. Utilisez des **objets** pour représenter les tâches et manipulez le DOM pour interagir avec l'utilisateur.

Exercice_2

Écrire une application pour gérer une bibliothèque de livres. L'application doit permettre d'**ajouter** des livres, de les **afficher** et de les **trier par titre**. Utilisez des **objets** pour représenter les livres et manipulez le DOM pour interagir avec l'utilisateur.