# **SÉANCE 1 – Fondations du Front-End Moderne**

## 📚 PARTIE THÉORIQUE

### 1. HTML5 : Structure et Sémantique

#### Les balises sémantiques HTML5

html

*<!-- Structure de base HTML5 -->*

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Mon Site</title>

</head>

<body>

<header>En-tête du site</header>

<nav>Navigation</nav>

<main>

<article>Contenu principal</article>

<aside>Contenu secondaire</aside>

</main>

<footer>Pied de page</footer>

</body>

</html>

**Balises sémantiques importantes :**

* <header> : En-tête de section ou de page
* <nav> : Navigation principale
* <main> : Contenu principal (unique par page)
* <article> : Contenu autonome
* <section> : Section thématique
* <aside> : Contenu complémentaire
* <footer> : Pied de page
* <figure> et <figcaption> : Illustrations avec légende

### 2. CSS3 : Flexbox, Grid, Responsive Design

#### Flexbox - Mise en page flexible

css

.container {

display: flex;

flex-direction: row; */\* row, column, row-reverse, column-reverse \*/*

justify-content: space-between; */\* flex-start, center, flex-end, space-around \*/*

align-items: center; */\* flex-start, center, flex-end, stretch \*/*

flex-wrap: wrap; */\* nowrap, wrap, wrap-reverse \*/*

gap: 20px;

}

.item {

flex: 1; */\* flex-grow: 1, flex-shrink: 1, flex-basis: 0% \*/*

flex-basis: 200px;

}

#### CSS Grid - Grille bidimensionnelle

css

.grid-container {

display: grid;

grid-template-columns: repeat(3, 1fr); */\* 3 colonnes égales \*/*

grid-template-rows: auto 1fr auto;

gap: 20px;

grid-template-areas:

"header header header"

"sidebar main main"

"footer footer footer";

}

.header { grid-area: header; }

.sidebar { grid-area: sidebar; }

.main { grid-area: main; }

.footer { grid-area: footer; }

#### Responsive Design - Media Queries

css

*/\* Mobile First \*/*

.container {

width: 100%;

padding: 10px;

}

*/\* Tablette \*/*

@media (min-width: 768px) {

.container {

width: 750px;

margin: 0 auto;

}

}

*/\* Desktop \*/*

@media (min-width: 1024px) {

.container {

width: 1000px;

}

}

### 3. JavaScript ES6+ : Variables, Fonctions, DOM

#### Variables et types

javascript

*// ES6+ : let, const (pas var)*

let age = 25;

const PI = 3.14;

*// Types de données*

const string = "Texte";

const number = 42;

const boolean = true;

const array = [1, 2, 3];

const object = { nom: "Jean", age: 30 };

const nullValue = null;

const undefinedValue = undefined;

#### Fonctions modernes

javascript

*// Fonction classique*

function addition(a, b) {

return a + b;

}

*// Arrow function*

const addition = (a, b) => a + b;

*// Fonction avec paramètres par défaut*

const saluer = (nom = "Visiteur") => `Bonjour ${nom}`;

*// Destructuration*

const { nom, age } = { nom: "Marie", age: 28 };

const [premier, deuxieme] = [10, 20];

*// Spread operator*

const arr1 = [1, 2, 3];

const arr2 = [...arr1, 4, 5]; *// [1, 2, 3, 4, 5]*

#### Manipulation du DOM

javascript

*// Sélection d'éléments*

const element = document.getElementById('mon-id');

const elements = document.querySelectorAll('.ma-classe');

const premier = document.querySelector('.ma-classe');

*// Modification du contenu*

element.textContent = "Nouveau texte";

element.innerHTML = "<strong>Texte en gras</strong>";

*// Modification des styles*

element.style.color = "blue";

element.classList.add('active');

element.classList.remove('hidden');

element.classList.toggle('visible');

*// Événements*

element.addEventListener('click', (e) => {

console.log('Cliqué !', e.target);

});

*// Création d'éléments*

const newDiv = document.createElement('div');

newDiv.textContent = "Nouveau div";

document.body.appendChild(newDiv);

## 💻 TD 1 : EXERCICES GUIDÉS

### TD1.1 : Créer une structure HTML sémantique

html

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Blog Personnel</title>

</head>

<body>

<header>

<h1>Mon Blog Tech</h1>

<nav>

<ul>

<li><a href="#accueil">Accueil</a></li>

<li><a href="#articles">Articles</a></li>

<li><a href="#contact">Contact</a></li>

</ul>

</nav>

</header>

<main>

<section id="articles">

<h2>Derniers Articles</h2>

<article>

<header>

<h3>Introduction à JavaScript</h3>

<time datetime="2025-10-30">30 octobre 2025</time>

</header>

<p>JavaScript est le langage de programmation du web...</p>

<footer>

<a href="#lire-plus">Lire la suite</a>

</footer>

</article>

<article>

<header>

<h3>CSS Grid vs Flexbox</h3>

<time datetime="2025-10-28">28 octobre 2025</time>

</header>

<p>Comprendre les différences entre Grid et Flexbox...</p>

</article>

</section>

<aside>

<h3>À propos</h3>

<p>Développeur passionné par le front-end</p>

</aside>

</main>

<footer>

<p>&copy; 2025 Mon Blog Tech</p>

</footer>

</body>

</html>

### TD1.2 : Layout avec Flexbox

css

*/\* styles.css \*/*

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif;

line-height: 1.6;

}

header {

background: #333;

color: white;

padding: 1rem;

}

header h1 {

margin-bottom: 1rem;

}

nav ul {

display: flex;

list-style: none;

gap: 2rem;

}

nav a {

color: white;

text-decoration: none;

}

main {

display: flex;

max-width: 1200px;

margin: 2rem auto;

gap: 2rem;

padding: 0 1rem;

}

#articles {

flex: 3;

}

aside {

flex: 1;

background: #f4f4f4;

padding: 1rem;

}

article {

background: white;

padding: 1.5rem;

margin-bottom: 1.5rem;

border: 1px solid #ddd;

border-radius: 5px;

}

article header {

background: transparent;

color: #333;

padding: 0;

margin-bottom: 1rem;

}

footer {

background: #333;

color: white;

text-align: center;

padding: 1rem;

margin-top: 2rem;

}

*/\* Responsive \*/*

@media (max-width: 768px) {

main {

flex-direction: column;

}

nav ul {

flex-direction: column;

gap: 0.5rem;

}

}

### TD1.3 : Layout avec CSS Grid

css

.grid-layout {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

gap: 20px;

padding: 20px;

}

.card {

background: white;

border: 1px solid #ddd;

border-radius: 8px;

padding: 1.5rem;

box-shadow: 0 2px 4px rgba(0,0,0,0.1);

transition: transform 0.3s;

}

.card:hover {

transform: translateY(-5px);

box-shadow: 0 4px 8px rgba(0,0,0,0.2);

}

### TD1.4 : Manipulation DOM avec JavaScript

javascript

*// Exercice : Ajouter des événements*

document.addEventListener('DOMContentLoaded', () => {

*// 1. Changer le titre au clic*

const titre = document.querySelector('h1');

titre.addEventListener('click', () => {

titre.textContent = 'Titre modifié !';

titre.style.color = 'red';

});

*// 2. Compter les clics*

let compteur = 0;

const bouton = document.getElementById('btn-compteur');

const affichage = document.getElementById('affichage');

bouton.addEventListener('click', () => {

compteur++;

affichage.textContent = `Clics : ${compteur}`;

});

*// 3. Validation de formulaire*

const form = document.getElementById('contact-form');

form.addEventListener('submit', (e) => {

e.preventDefault();

const nom = document.getElementById('nom').value;

const email = document.getElementById('email').value;

if (nom === '' || email === '') {

alert('Veuillez remplir tous les champs');

return;

}

console.log('Formulaire envoyé:', { nom, email });

});

});

## 🚀 TP 1 : MINI-PAGE INTERACTIVE

### Objectif

Créer une page de portfolio interactive avec :

* Menu de navigation
* Section présentation
* Galerie de projets
* Formulaire de contact
* Interactions JavaScript

### Code complet du TP1 (voir tp/html)