JOUR 3 – JavaScript : Interactivité

🎯 Objectif

Rendre votre site interactif et dynamique en maîtrisant les fondamentaux de JavaScript.

📚 Bases JavaScript - Comprendre les Fondamentaux

🔤 Variables : Stocker des informations

Explication : Les variables sont des boîtes qui stockent des données. Pensez-y comme des étiquettes pour vos informations.

```
// const : constante (ne peut pas changer)
const nomUtilisateur = 'Ahmad';  // Toujours 'Ahmad'
const PI = 3.14159;              // Toujours 3.14159

// let : variable qui peut changer
let age = 28;                    // Commence à 28
age = 29;                        // Peut devenir 29

// var : ancienne méthode (évitez d'utiliser)
var vieilleVariable = 'obsolète';
```

🔧 Fonctions : Des blocs de code réutilisables

Explication : Les fonctions sont comme des machines : vous leur donnez quelque chose, elles font un travail et vous retournent un résultat.

```
// Fonction classique (déclarative)
function calculerSurface(longueur, largeur) {
    // Calcul simple
    const surface = longueur * largeur;

    // Retourner le résultat
    return surface;
}

// Fonction flèche (moderne)
const calculerPerimetre = (longueur, largeur) => {
    return 2 * (longueur + largeur);
};

// Fonction flèche simplifiée (une seule ligne)
const auCarre = (nombre) => nombre * nombre;
```

📊 Types de données

Explication : JavaScript peut manipuler différents types d'informations :

```
// Texte (chaînes de caractères)
const prenom = "Marie";
const message = 'Bonjour le monde !';
const citation = `Ceci est un "template literal" avec ${prenom}`;

// Nombres
const entier = 42;
const decimal = 3.14;
const negatif = -10;

// Booléens (vrai/faux)
const estMajeur = true;
const estVide = false;

// Tableaux (listes)
const fruits = ['pomme', 'banane', 'orange'];
const nombres = [1, 2, 3, 4, 5];

// Objets (dictionnaires)
const personne = {
    nom: 'Dupont',
    prenom: 'Jean',
    age: 30
};
```

Conditions : Prendre des décisions

Explication : Les conditions permettent à votre programme de faire des choix.

```
const age = 18;
const aUnPermis = true;

// if / else : si... sinon...
if (age >= 18) {
    console.log(' Majeur');
} else {
    console.log(' Mineur');
}

// if / else if / else : multiples conditions
if (age < 18) {
    console.log(' Mineur');
} else if (age < 65) {
    console.log(' Adulte');
} else {
    console.log(' Senior');
}

// Conditions combinées
if (age >= 18 && aUnPermis) {
    console.log(' Peut conduire');
}

// Opérateur ternaire (condensé)
const messageAge = age >= 18 ? 'Majeur' : 'Mineur';
```

Boucles : Répéter des actions

Explication : Les boucles évitent de répéter le même code plusieurs fois.

```
// for : boucle avec compteur
console.log('=== Compteur avec for ===');
for (let i = 0; i < 5; i++) {
    console.log(`i = ${i}`);
}

// while : boucle avec condition
console.log('\n=== Compteur avec while ===');
let compteur = 0;
while (compteur < 3) {
    console.log(`compteur = ${compteur}`);
    compteur++;
}

// forEach : parcourir un tableau
console.log('\n=== Parcourir un tableau ===');
const fruits = ['🍎 pomme', '🍌 banane', '🍊 orange'];
fruits.forEach((fruit, index) => {
    console.log(`Fruit ${index + 1}: ${fruit}`);
});
```

🌐 Manipulation du DOM : Parler à votre page HTML

🎯 Sélectionner des éléments

Explication : Le DOM (Document Object Model) est la structure de votre page. Pour la modifier, il faut d'abord sélectionner les éléments.

```
<!-- Exemple HTML pour nos sélections -->
<div id="container">
    <h1 class="titre">Mon titre principal</h1>
    <p class="paragraphe">Premier paragraphe</p>
    <p class="paragraphe">Deuxième paragraphe</p>
    <button id="monBouton">Cliquez-moi</button>
</div>
```

```
// Sélection par ID (unique)
const container = document.getElementById('container');

// Sélection par classe (peut retourner plusieurs éléments)
const paragraphes = document.getElementsByClassName('paragraphe');

// Sélection avec querySelector (moderne et flexible)
const titre = document.querySelector('.titre');        // Premier élément avec classe "titre"
const bouton = document.querySelector('#monBouton');    // Élément avec ID "monBouton"
const premierP = document.querySelector('p');           // Premier <p>

// Sélectionner TOUS les éléments correspondants
const tousLesParagraphes = document.querySelectorAll('.paragraphe');
console.log(`Nombre de paragraphes : ${tousLesParagraphes.length}`);
```

✏️ Modifier le contenu

Explication : Une fois un élément sélectionné, vous pouvez modifier son texte, son style, etc.

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

// Changer le texte
titre.textContent = 'Nouveau titre !';

// Changer le HTML (attention à la sécurité !)
titre.innerHTML = '<span style="color: red;">Titre en rouge</span>';

// Modifier les classes CSS
titre.classList.add('important');        // Ajouter une classe
titre.classList.remove('ancien');        // Supprimer une classe
titre.classList.toggle('visible');       // Ajouter si absent, supprimer si présent
titre.classList.contains('important');   // Vérifier si la classe existe

// Modifier les styles directement
titre.style.color = 'blue';
titre.style.fontSize = '24px';
titre.style.backgroundColor = '#f0f0f0';
```

Créer et ajouter des éléments

Explication : Vous pouvez créer de nouveaux éléments HTML et les ajouter à votre page.

```
// Créer un nouvel élément
const nouveauParagraphe = document.createElement('p');
nouveauParagraphe.textContent = 'Je suis un nouveau paragraphe !';
nouveauParagraphe.classList.add('nouveau');

// Ajouter à la fin du body
document.body.appendChild(nouveauParagraphe);

// Créer une liste dynamique
const fruits = ['pomme', 'banane', 'orange'];
const liste = document.createElement('ul');

fruits.forEach(fruit => {
    const element = document.createElement('li');
    element.textContent = fruit;
    liste.appendChild(element);
});

document.body.appendChild(liste);
```

Gestion des événements : Réagir aux actions de l'utilisateur

Événements de base

Explication : Les événements sont des actions (clic, survol, saisie) qui déclenchent du code JavaScript.

```
// Clic sur un bouton
const bouton = document.querySelector('#monBouton');

bouton.addEventListener('click', () => {
    alert('🎉 Bouton cliqué !');
});

// Clic avec accès à l'élément
bouton.addEventListener('click', function(event) {
    console.log('Élément cliqué :', this);
    console.log('Événement :', event);

    // Changer le texte du bouton
    this.textContent = 'Déjà cliqué !';
    this.style.backgroundColor = '#4CAF50';
});
```

📝 Gestion de formulaires

Explication : Les formulaires sont parfaits pour collecter et traiter les données utilisateur.

```
<form id="monFormulaire">
    <input type="text" id="nom" placeholder="Votre nom" required>
    <input type="email" id="email" placeholder="Votre email" required>
    <button type="submit">Envoyer</button>
</form>

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

```
const formulaire = document.querySelector('#monFormulaire');
const resultat = document.querySelector('#resultat');

formulaire.addEventListener('submit', function(event) {
    // Empêcher l'envoi normal du formulaire
    event.preventDefault();

    // Récupérer les valeurs
    const nom = document.querySelector('#nom').value;
    const email = document.querySelector('#email').value;

    // Validation simple
    if (nom.length < 2) {
        resultat.innerHTML = '<p style="color: red;">❌ Nom trop court</p>';
        return;
    }

    if (!email.includes('@')) {
        resultat.innerHTML = '<p style="color: red;">❌ Email invalide</p>';
        return;
    }

    // Succès
    resultat.innerHTML = `
        <p style="color: green;">✅ Données reçues !</p>
        <p><strong>Nom :</strong> ${nom}</p>
        <p><strong>Email :</strong> ${email}</p>
    `;

    // Vider le formulaire
    formulaire.reset();
});
```

🎯 Autres événements utiles

```
 const input = document.querySelector('#nom');
const image = document.querySelector('.ma-image');

// Événement de saisie (chaque caractère tapé)
input.addEventListener('input', function() {
    console.log('Contenu :', this.value);
});

// Événement de survol
image.addEventListener('mouseenter', function() {
    this.style.transform = 'scale(1.1)';
    this.style.transition = 'transform 0.3s';
});

image.addEventListener('mouseleave', function() {
    this.style.transform = 'scale(1)';
});

// Événement clavier
document.addEventListener('keydown', function(event) {
    if (event.key === 'Escape') {
        console.log('Touche Échap pressée');
    }
});
```

🚀 Projets pratiques complets

🍔 Menu burger (responsive)

HTML :

```
<header>
    <div class="logo">MonSite</div>
    <button class="menu-toggle" aria-label="Menu">
        <span></span>
        <span></span>
        <span></span>
    </button>
    <nav class="nav-menu">
        <a href="#accueil">Accueil</a>
        <a href="#services">Services</a>
        <a href="#contact">Contact</a>
    </nav>
</header>
```

CSS :

```
.menu-toggle {
    display: none;
    flex-direction: column;
    background: none;
    border: none;
    cursor: pointer;
    padding: 5px;
}

.menu-toggle span {
    width: 25px;
    height: 3px;
    background: #333;
    margin: 3px 0;
    transition: 0.3s;
}

.nav-menu {
    display: flex;
    gap: 20px;
}

/* Mobile */
@media (max-width: 768px) {
    .menu-toggle {
        display: flex;
    }

    .nav-menu {
        position: fixed;
        top: 0;
        right: -100%;
        width: 80%;
        height: 100vh;
        background: white;
        flex-direction: column;
        padding: 80px 20px;
        transition: right 0.3s;
        box-shadow: -2px 0 10px rgba(0,0,0,0.1);
    }

    .nav-menu.active {
        right: 0;
    }
}
```

JavaScript :

```
const menuToggle = document.querySelector('.menu-toggle');
const navMenu = document.querySelector('.nav-menu');

// Ouvrir/fermer le menu
menuToggle.addEventListener('click', () => {
    navMenu.classList.toggle('active');

    // Animation du bouton burger
    menuToggle.classList.toggle('active');
});

// Fermer le menu quand on clique sur un lien
document.querySelectorAll('.nav-menu a').forEach(link => {
    link.addEventListener('click', () => {
        navMenu.classList.remove('active');
        menuToggle.classList.remove('active');
    });
});

// Fermer le menu en cliquant dehors
document.addEventListener('click', (event) => {
    if (!menuToggle.contains(event.target) && !navMenu.contains(event.target)) {
        navMenu.classList.remove('active');
        menuToggle.classList.remove('active');
    }
});
```

🖼️ Galerie d'images avec navigation

HTML :

```
<div class="galerie">
    <div class="galerie-container">
        <img id="image-principale" src="https://picsum.photos/600/400?random=1" alt="Photo galerie">
        <div class="galerie-info">
            <span id="numero-image">1 / 5</span>
            <p id="description-image">Description de l'image</p>
        </div>
    </div>

    <div class="galerie-controles">
        <button id="precedent" aria-label="Image précédente">❮</button>
        <button id="play-pause" aria-label="Lecture/pause">▶</button>
        <button id="suivant" aria-label="Image suivante">❯</button>
    </div>

    <div class="miniatures">
        <!-- Miniatures générées par JavaScript -->
    </div>
</div>
```

CSS :

```
.galerie {
    max-width: 600px;
    margin: 20px auto;
    background: white;
    border-radius: 10px;
    overflow: hidden;
    box-shadow: 0 5px 20px rgba(0,0,0,0.1);
}

.galerie-container {
    position: relative;
```

```
}

.galerie-container img {
    width: 100%;
    height: 400px;
    object-fit: cover;
}

.galerie-info {
    position: absolute;
    bottom: 0;
    left: 0;
    right: 0;
    background: linear-gradient(transparent, rgba(0,0,0,0.7));
    color: white;
    padding: 20px;
}

.galerie-controles {
    display: flex;
    justify-content: center;
    align-items: center;
    gap: 20px;
    padding: 20px;
    background: #f5f5f5;
}

.galerie-controles button {
    background: #333;
    color: white;
    border: none;
    width: 50px;
    height: 50px;
    border-radius: 50%;
    font-size: 18px;
    cursor: pointer;
    transition: background 0.3s;
}

.galerie-controles button:hover {
    background: #555;
}

.miniatures {
    display: flex;
    gap: 10px;
    padding: 20px;
    background: #fafafa;
    overflow-x: auto;
}

.miniature {
    width: 80px;
    height: 60px;
    object-fit: cover;
    border-radius: 5px;
    cursor: pointer;
    opacity: 0.6;
    transition: opacity 0.3s;
}

.miniature.active {
    opacity: 1;
    border: 2px solid #333;
}
```

JavaScript :

```
class GalerieImages {
    constructor() {
        this.images = [
            { src: 'https://picsum.photos/600/400?random=1', thumb: 'https://picsum.photos/80/60?random=1', description: 'Paysage mo
            { src: 'https://picsum.photos/600/400?random=2', thumb: 'https://picsum.photos/80/60?random=2', description: 'Ville mode
            { src: 'https://picsum.photos/600/400?random=3', thumb: 'https://picsum.photos/80/60?random=3', description: 'Plage trop
            { src: 'https://picsum.photos/600/400?random=4', thumb: 'https://picsum.photos/80/60?random=4', description: 'Forêt auto
            { src: 'https://picsum.photos/600/400?random=5', thumb: 'https://picsum.photos/80/60?random=5', description: 'Architectu
        ];

        this.currentIndex = 0;
        this.isPlaying = false;
        this.intervalId = null;

        this.initElements();
        this.initEvents();
        this.createMiniatures();
        this.showImage(0);
    }

    initElements() {
        this.imagePrincipale = document.querySelector('#image-principale');
        this.boutonPrecedent = document.querySelector('#precedent');
        this.boutonSuivant = document.querySelector('#suivant');
        this.boutonPlayPause = document.querySelector('#play-pause');
        this.numeroImage = document.querySelector('#numero-image');
        this.descriptionImage = document.querySelector('#description-image');
        this.miniaturesContainer = document.querySelector('.miniatures');
    }

    initEvents() {
        this.boutonPrecedent.addEventListener('click', () => this.precedent());
        this.boutonSuivant.addEventListener('click', () => this.suivant());
        this.boutonPlayPause.addEventListener('click', () => this.togglePlayPause());

        // Navigation avec clavier
        document.addEventListener('keydown', (e) => {
            if (e.key === 'ArrowLeft') this.precedent();
            if (e.key === 'ArrowRight') this.suivant();
            if (e.key === ' ') {
                e.preventDefault();
                this.togglePlayPause();
            }
        });
    }

    createMiniatures() {
        this.images.forEach((image, index) => {
            const miniature = document.createElement('img');
            miniature.src = image.thumb;
            miniature.alt = image.description;
            miniature.classList.add('miniature');

            miniature.addEventListener('click', () => this.showImage(index));

            this.miniaturesContainer.appendChild(miniature);
        });

        this.miniatures = document.querySelectorAll('.miniature');
    }

    showImage(index) {
        // Valider l'index
        if (index < 0) index = this.images.length - 1;
```

```
        if (index < 0) index = this.images.length - 1;
        if (index >= this.images.length) index = 0;

        this.currentIndex = index;
        const image = this.images[index];

        // Mettre à jour l'image principale
        this.imagePrincipale.style.opacity = '0';

        setTimeout(() => {
            this.imagePrincipale.src = image.src;
            this.imagePrincipale.alt = image.description;
            this.numeroImage.textContent = `${index + 1} / ${this.images.length}`;
            this.descriptionImage.textContent = image.description;
            this.imagePrincipale.style.opacity = '1';
        }, 300);

        // Mettre à jour les miniatures
        this.miniatures.forEach((mini, i) => {
            mini.classList.toggle('active', i === index);
        });
    }

    precedent() {
        this.showImage(this.currentIndex - 1);
    }

    suivant() {
        this.showImage(this.currentIndex + 1);
    }

    togglePlayPause() {
        if (this.isPlaying) {
            this.pause();
        } else {
            this.play();
        }
    }

    play() {
        this.isPlaying = true;
        this.boutonPlayPause.textContent = '⏸';

        this.intervalId = setInterval(() => {
            this.suivant();
        }, 3000);
    }

    pause() {
        this.isPlaying = false;
        this.boutonPlayPause.textContent = '▶';

        if (this.intervalId) {
            clearInterval(this.intervalId);
            this.intervalId = null;
        }
    }
}

// Initialiser la galerie quand la page est chargée
document.addEventListener('DOMContentLoaded', () => {
    new GalerieImages();
});
```

✅ Checklist Jour 3

📚 Théorie

- [ ] Variables (const, let, var) comprises
- [ ] Fonctions (classiques et fléchées) maîtrisées
- [ ] Types de données connus