Rappel : HTML, CSS & JavaScript

Fondamentaux du développement web

Ibrahim ALAME

Formation Vue.js

20 novembre 2025

Plan

Objectifs de cette session

Pourquoi ce rappel ?

Vue.js nécessite une bonne maîtrise des technologies web de base

Ce que nous allons revoir :

- HTML : Structure et contenu
- CSS : Mise en forme et présentation
- JavaScript : Interactivité et logique

Important

Ces bases sont essentielles pour comprendre Vue.js !

Les trois piliers du web

HTML

Structure

Le squelette

CSS

Présentation

La peau

JavaScript

Comportement

Les muscles

Ensemble, ils créent des applications web modernes

HTML - Structure de base

HTML5 - Dernière version standard

HyperText Markup Language - Langage de balisage

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width">
  <title>Ma page web</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Bienvenue</h1>
  <p>Premier paragraphe.</p>

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

Éléments, balises et attributs

Balise ouvrante et fermante :

- `<p>Texte</p>`
- `<div>Contenu</div>`

Balises auto-fermantes :

- `<img src="photo.jpg" alt="Photo">`
- `<input type="text">`
- `<br>`

Attributs :

- `class="ma-classe"`
- `id="mon-id"`
- `href="https://..."`

Balises de contenu essentielles

Titres (Headings) :

```
<h1>Titre principal</h1>
<h2>Sous-titre</h2>
<h3>Section</h3>
```

Paragraphes et mise en forme :

```
<p>Un paragraphe de texte.</p>
<strong>Texte important (gras)</strong>
<em>Texte emphase (italique)</em>
<mark>Texte surligne</mark>
```

Saut de ligne :

```
<p>Ligne 1<br>Ligne 2</p>
```

Listes

Liste non ordonnée :

```
<ul>
  <li>Element 1</li>
  <li>Element 2</li>
  <li>Element 3</li>
</ul>
```

Résultat :

- Element 1
- Element 2
- Element 3

Liste ordonnée :

```
<ol>
  <li>Premier</li>
  <li>Deuxieme</li>
  <li>Troisieme</li>
</ol>
```

Résultat :

1. Premier
2. Deuxieme
3. Troisieme

Liens et images

Liens (ancres) :

```
<!-- Lien externe -->
<a href="https://vuejs.org">Site Vue.js</a>

<!-- Lien interne -->
<a href="#section1">Aller a la section 1</a>

<!-- Ouvrir dans un nouvel onglet -->
<a href="page.html" target="_blank">Lien</a>
```

Images :

```
<img src="images/logo.png" alt="Logo Vue.js">

<img src="photo.jpg" alt="Photo"
     width="300" height="200">
```

Div et Span - Conteneurs

div - Block

Conteneur de niveau bloc (occupe toute la largeur)

```
<div class="container">
  <h2>Titre de section</h2>
  <p>Contenu de la section</p>
</div>
```

span - Inline

Conteneur de niveau inline (dans le flux du texte)

```
<p>Texte avec un <span class="important">
   mot important</span> dedans.</p>
```

CSS - Introduction

CSS = Cascading Style Sheets

Langage de feuilles de style pour la présentation

Où écrire le CSS ?

1. **Externe** (recommandé) : `<link rel="stylesheet" href="style.css">`
2. **Interne** : Dans une balise `<style>` dans le `<head>`
3. **Inline** : Attribut `style="color: red;"` (à éviter)

Bonne pratique

Toujours séparer le contenu (HTML) de la présentation (CSS)

Syntaxe CSS de base

```
selecteur {
  propriete: valeur;
  autre-propriete: autre-valeur;
}
```

Exemple :

```
p {
  color: blue;
  font-size: 16px;
  margin: 10px;
}
```

Tous les paragraphes seront bleus, 16px, avec marge de 10px

Sélecteurs CSS

```css
/* Selecteur d'element */
p { color: black; }

/* Selecteur de classe */
.ma-classe { color: red; }

/* Selecteur d'ID */
#mon-id { color: blue; }

/* Selecteur descendant */
div p { color: green; }

/* Plusieurs selecteurs */
h1, h2, h3 { font-family: Arial; }

/* Selecteur d'attribut */
a[href^="https"] { color: green; }
```

Classes et IDs

Classes (réutilisables)
HTML :

```
<p class="highlight">
  Texte 1
</p>
<p class="highlight">
  Texte 2
</p>
```

CSS :

```
.highlight {
  background: yellow;
}
```

IDs (uniques)
HTML :

```
<div id="header">
  En-tete
</div>
```

CSS :

```
#header {
  background: navy;
  color: white;
}
```

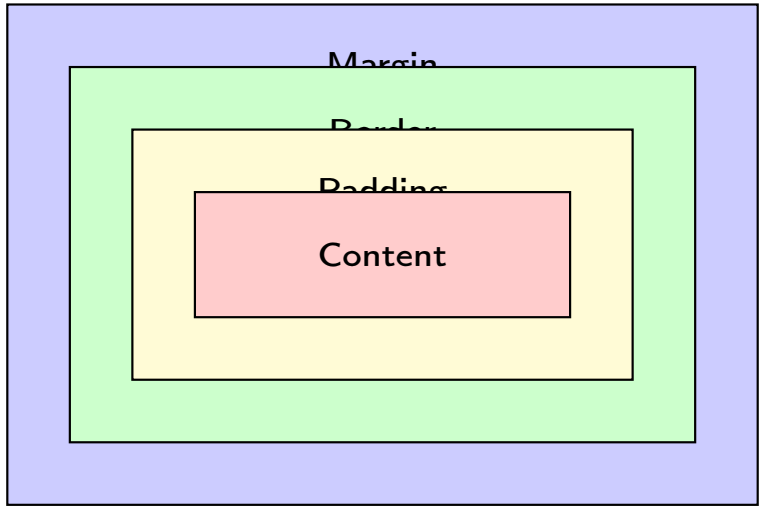
Règle

Un ID = unique sur la page • Une classe = réutilisable

Propriétés de texte

```css
p {
  /* Famille de police */
  font-family: Arial, sans-serif;

  /* Taille */
  font-size: 16px;

  /* Graisse */
  font-weight: bold; /* ou 400, 700, etc. */

  /* Style */
  font-style: italic;

  /* Couleur */
  color: #333;

  /* Alignement */
  text-align: center;

  /* Hauteur de ligne */
  line-height: 1.5;
}
```

Le modèle de boîte (Box Model)




```
.box {
  width: 200px;
  padding: 20px;
  border: 2px solid black;
  margin: 10px;
}
```

Margin et Padding

Padding (marge intérieure) :

```
padding: 10px;                    /* Tous cotes */
padding: 10px 20px;               /* Haut/Bas Gauche/
    Droite */
padding: 10px 20px 15px 25px; /* Haut Droite Bas
    Gauche */
padding-top: 10px;                /* Un seul cote */
```

Margin (marge extérieure) :

```
margin: 10px;                     /* Tous cotes */
margin: 0 auto;                   /* Centrer
    horizontalement */
margin-bottom: 20px;              /* Un seul cote */
```

Bordures

```css
/* Bordure complete */
border: 2px solid black;

/* Par cote */
border-top: 1px dashed red;
border-right: 3px dotted blue;

/* Bordures arrondies */
border-radius: 10px;
border-radius: 50%; /* Cercle */

/* Ombre de boite */
box-shadow: 5px 5px 10px rgba(0,0,0,0.3);
```

Display et Position

Display :

```
display: block;        /* Occupe toute la largeur */
display: inline;       /* Dans le flux du texte */
display: inline-block; /* Melange des deux */
display: none;         /* Masque l'element */
```

Position :

```
position: static;     /* Par defaut */
position: relative;   /* Relatif a sa position
   normale */
position: absolute;   /* Relatif au parent
   positionne */
position: fixed;      /* Fixe dans la fenetre */
position: sticky;     /* Melange relative + fixed
   */
```

Couleurs et Background

Couleurs :

```
color: red;                    /* Nom */
color: #FF0000;                /* Hexadecimal */
color: rgb(255, 0, 0);         /* RGB */
color: rgba(255, 0, 0, 0.5); /* RGBA (avec opacite
   ) */
```

Arrière-plan :

```
background-color: lightblue;
background-image: url('image.jpg');
background-size: cover;
background-position: center;
background-repeat: no-repeat;

/* Raccourci */
background: lightblue url('img.jpg') no-repeat center;
```

JavaScript - Introduction

JavaScript

Langage de programmation pour ajouter de l'interactivité

Où écrire le JavaScript ?

1. **Externe** (recommandé) : `<script src="script.js"></script>`
2. **Interne** : Dans une balise `<script>` avant `</body>`
3. **Inline** : Attributs `onclick="..."` (à éviter)

Caractéristiques :

- Langage interprété
- Typé dynamiquement
- Orienté objet (prototypes)
- Asynchrone

Variables : var, let, const

```
// var : ancienne facon (eviter)
var x = 10;

// let : variable modifiable
let nom = 'Alice';
nom = 'Bob'; // OK

// const : constante (non modifiable)
const PI = 3.14159;
// PI = 3; // ERREUR !

// const avec objet/tableau
const tableau = [1, 2, 3];
tableau.push(4); // OK (modifie le contenu)
// tableau = [5, 6]; // ERREUR (reassignation)
```

Types de données

```
// Types primitifs
let nombre = 42;                  // Number
let texte = 'Bonjour';            // String
let vrai = true;                  // Boolean
let rien = null;                  // Null
let nonDefini = undefined;        // Undefined

// Types complexes
let tableau = [1, 2, 3];
let objet = { nom: 'Alice', age: 25 };
let fonction = function() { return 'Hello'; };

// Verifier le type
console.log(typeof nombre);       // "number"
console.log(typeof texte);        // "string"
console.log(typeof tableau);      // "object"
```

Opérateurs

```
// Arithmetiques
let a = 10 + 5;    // 15
let b = 10 - 5;    // 5
let c = 10 * 5;    // 50
let d = 10 / 5;    // 2
let e = 10 % 3;    // 1 (modulo)
let f = 2 ** 3;    // 8 (puissance)

// Comparaison
10 == '10'     // true (egalite faible)
10 === '10'    // false (egalite stricte)
10 != '10'     // false
10 !== '10'    // true
10 > 5         // true
10 <= 10       // true

// Logiques
true && false   // false (ET)
true || false   // true (OU)
!true           // false (NON)
```

Conditions : if/else

```
let age = 18;

if (age >= 18) {
  console.log('Majeur');
} else if (age >= 13) {
  console.log('Adolescent');
} else {
  console.log('Enfant');
}

// Operateur ternaire
let statut = age >= 18 ? 'Majeur' : 'Mineur';

// Switch
switch (jour) {
  case 'lundi':
    console.log('Debut de semaine');
    break;
  case 'vendredi':
    console.log('Bientot le weekend !');
    break;
  default:
    console.log('Jour ordinaire');
}
```

Boucles

```javascript
// Boucle for classique
for (let i = 0; i < 5; i++) {
  console.log(i); // 0, 1, 2, 3, 4
}

// Boucle while
let j = 0;
while (j < 5) {
  console.log(j);
  j++;
}

// Boucle for...of (tableaux)
let fruits = ['pomme', 'banane', 'orange'];
for (let fruit of fruits) {
  console.log(fruit);
}

// Boucle for...in (objets)
let personne = { nom: 'Alice', age: 25 };
for (let cle in personne) {
  console.log(cle, personne[cle]);
}
```

Fonctions

```
// Declaration de fonction
function saluer(nom) {
  return 'Bonjour ' + nom;
}

console.log(saluer('Alice')); // "Bonjour Alice"

// Expression de fonction
const additionner = function(a, b) {
  return a + b;
};

// Fonction fleche (arrow function)
const multiplier = (a, b) => a * b;

// Parametre par defaut
function saluer(nom = 'invite') {
  return 'Bonjour ' + nom;
}

saluer();          // "Bonjour invite"
saluer('Bob');     // "Bonjour Bob"
```

Fonctions fléchées

```
// Syntaxe complete
const double = (x) => {
  return x * 2;
};

// Syntaxe courte (return implicite)
const double = x => x * 2;

// Plusieurs parametres
const additionner = (a, b) => a + b;

// Sans parametre
const direBonjour = () => 'Bonjour';

// Retour d'objet (attention aux parentheses)
const creerPersonne = (nom, age) => ({
  nom: nom,
  age: age
});

console.log(creerPersonne('Alice', 25));
// { nom: 'Alice', age: 25 }
```

Objets

```
// Creation d'objet
const personne = {
  nom: 'Alice',
  age: 25,
  ville: 'Paris',
  saluer: function() {
    return 'Bonjour, je suis ' + this.nom;
  }
};

// Acces aux proprietes
console.log(personne.nom);         // "Alice"
console.log(personne['age']);      // 25
console.log(personne.saluer());    // "Bonjour, je suis Alice"

// Modifier une propriete
personne.age = 26;

// Ajouter une propriete
personne.email = 'alice@example.com';

// Supprimer une propriete
delete personne.ville;
```

Tableaux

```
// Creation de tableau
let fruits = ['pomme', 'banane', 'orange'];

// Acces aux elements
console.log(fruits[0]);  // "pomme"
console.log(fruits.length); // 3

// Ajouter des elements
fruits.push('kiwi');          // Ajoute a la fin
fruits.unshift('fraise');     // Ajoute au debut

// Supprimer des elements
fruits.pop();         // Supprime le dernier
fruits.shift();       // Supprime le premier

// Trouver un element
let index = fruits.indexOf('banane'); // 1
let existe = fruits.includes('pomme'); // true

// Parcourir un tableau
fruits.forEach(fruit => {
  console.log(fruit);
});
```

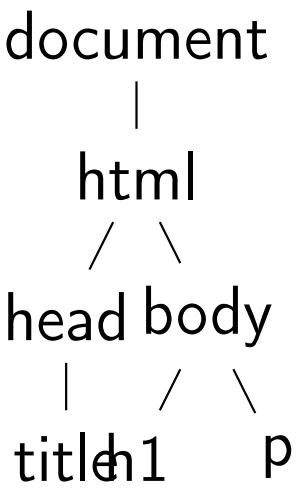
Méthodes de tableaux

```
let nombres = [1, 2, 3, 4, 5];

// map : transformer chaque element
let doubles = nombres.map(n => n * 2);
// [2, 4, 6, 8, 10]

// filter : filtrer les elements
let pairs = nombres.filter(n => n % 2 === 0);
// [2, 4]

// find : trouver le premier element
let premier = nombres.find(n => n > 3);
// 4

// reduce : reduire a une seule valeur
let somme = nombres.reduce((acc, n) => acc + n, 0);
// 15

// some : au moins un element respecte la condition
let aDesGrands = nombres.some(n => n > 4);
// true

// every : tous les elements respectent la condition
let tousPositifs = nombres.every(n => n > 0);
// true
```

Le DOM

DOM = Document Object Model

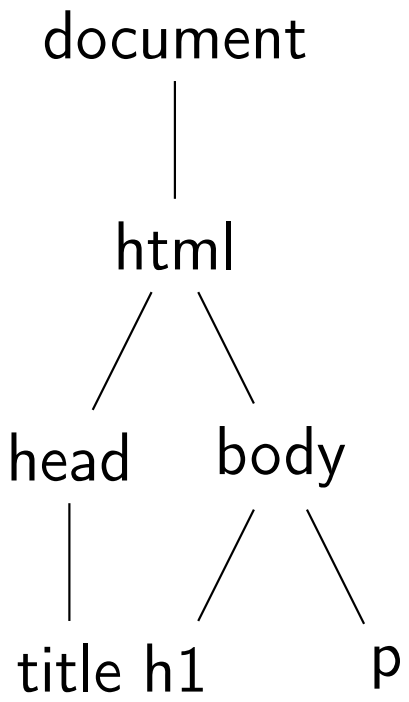
Représentation en mémoire de la page HTML sous forme d'arbre d'objets



JavaScript peut manipuler le DOM pour :

- Modifier le contenu
- Changer les styles

Le DOM



JavaScript peut manipuler le DOM pour :

- Modifier le contenu
- Changer les styles
- Ajouter/supprimer des éléments
- Réagir aux événements

Sélectionner des éléments

```javascript
// Par ID
let element = document.getElementById('mon-id');

// Par classe (retourne une liste)
let elements = document.getElementsByClassName('ma-classe');

// Par balise
let paragraphes = document.getElementsByTagName('p');

// Selecteur CSS (premier element)
let premier = document.querySelector('.ma-classe');

// Selecteur CSS (tous les elements)
let tous = document.querySelectorAll('.ma-classe');

// Exemple
let titre = document.querySelector('h1');
let boutons = document.querySelectorAll('.btn');
```

Modifier le contenu

```
let element = document.querySelector('#mon-element');

// Modifier le texte
element.textContent = 'Nouveau texte';

// Modifier le HTML
element.innerHTML = '<strong>Texte en gras</strong>';

// Attention : innerHTML peut etre dangereux
// Privilegier textContent pour du texte simple

// Modifier un attribut
element.setAttribute('class', 'nouvelle-classe');
element.src = 'nouvelle-image.jpg'; // Pour <img>
element.href = 'https://...';       // Pour <a>

// Obtenir un attribut
let classe = element.getAttribute('class');
let id = element.id; // raccourci pour getAttribute('id')

// Verifier l'existence d'un attribut
if (element.hasAttribute('data-info')) {
  // ...
}
```

Modifier le style

```
let element = document.querySelector('#mon-element');

// Modifier directement le style
element.style.color = 'red';
element.style.backgroundColor = 'yellow';
element.style.fontSize = '20px';

// Ajouter/retirer des classes CSS
element.classList.add('active');
element.classList.remove('hidden');
element.classList.toggle('visible'); // Ajoute ou retire

// Verifier si une classe existe
if (element.classList.contains('active')) {
  // ...
}

// Plusieurs classes a la fois
element.classList.add('class1', 'class2');
```

Créer et ajouter des éléments

```
// Creer un nouvel element
let nouveauDiv = document.createElement('div');
nouveauDiv.textContent = 'Nouveau contenu';
nouveauDiv.classList.add('ma-classe');

// Ajouter l'element a la page
let container = document.querySelector('#container');
container.appendChild(nouveauDiv);

// Inserer avant un autre element
let premier = container.firstChild;
container.insertBefore(nouveauDiv, premier);

// Supprimer un element
let elementASupprimer = document.querySelector('.a-supprimer');
elementASupprimer.remove();

// Ou via le parent
container.removeChild(elementASupprimer);

// Remplacer un element
let ancien = document.querySelector('.ancien');
let nouveau = document.createElement('div');
ancien.replaceWith(nouveau);
```

Les événements

Événements

Actions qui se produisent dans le navigateur (clic, frappe, chargement...)

Événements courants :

- `click` : Clic de souris
- `submit` : Soumission de formulaire
- `change` : Modification d'input
- `keydown/keyup` : Touche pressée/relâchée
- `mouseover/mouseout` : Survol souris
- `load` : Page/image chargée
- `DOMContentLoaded` : DOM prêt

addEventListener

```
// Selectionner l'element
let bouton = document.querySelector('#mon-bouton');

// Ajouter un gestionnaire d'evenement
bouton.addEventListener('click', function() {
  console.log('Bouton clique !');
});

// Avec fonction fleche
bouton.addEventListener('click', () => {
  console.log('Bouton clique !');
});

// Avec fonction nommee
function gererClic() {
  console.log('Bouton clique !');
}

bouton.addEventListener('click', gererClic);
```

L'objet event

```
bouton.addEventListener('click', function(event) {
  // event contient des infos sur l'evenement
  console.log(event.type);       // "click"
  console.log(event.target);     // Element clique
  console.log(event.currentTarget); // Element avec listener

  // Empecher le comportement par defaut
  event.preventDefault();

  // Arreter la propagation
  event.stopPropagation();
});

// Exemple avec un lien
let lien = document.querySelector('a');
lien.addEventListener('click', (e) => {
  e.preventDefault(); // Empeche le lien de naviguer
  console.log('Lien clique mais pas de navigation');
});

// Exemple avec formulaire
let form = document.querySelector('form');
form.addEventListener('submit', (e) => {
  e.preventDefault(); // Empeche l'envoi du formulaire
  // Valider les donnees avant envoi
});
```

Exemples d'événements

```
// Evenement sur input
let input = document.querySelector('input');
input.addEventListener('input', (e) => {
  console.log('Valeur:', e.target.value);
});

// Evenement clavier
document.addEventListener('keydown', (e) => {
  console.log('Touche pressee:', e.key);
  if (e.key === 'Enter') {
    console.log('Enter presse !');
  }
});

// Evenement souris
let div = document.querySelector('.zone');
div.addEventListener('mouseover', () => {
  div.style.backgroundColor = 'yellow';
});
div.addEventListener('mouseout', () => {
  div.style.backgroundColor = 'white';
});

// Attendre que le DOM soit pret
document.addEventListener('DOMContentLoaded', () => {
  console.log('DOM pret !');
  // Initialiser l'application ici
});
```

Exemple : Compteur interactif

HTML :

```
<div id="compteur">
  <h2>Compteur : <span id="valeur">0</span></h2>
  <button id="incrementer">+</button>
  <button id="decrementer">-</button>
  <button id="reinitialiser">Reset</button>
</div>
```

CSS :

```
#compteur { text-align: center; padding: 20px; }
button { margin: 5px; padding: 10px 20px; font-size: 16px; }
#valeur { color: blue; font-weight: bold; }
```

Exemple : Compteur interactif (JS)

```
// Attendre que le DOM soit pret
document.addEventListener('DOMContentLoaded', () => {
  // Selectionner les elements
  let valeurElement = document.querySelector('#valeur');
  let btnIncrementer = document.querySelector('#incrementer');
  let btnDecrementer = document.querySelector('#decrementer');
  let btnReset = document.querySelector('#reinitialiser');

  // Variable pour stocker la valeur
  let compteur = 0;

  // Fonction pour mettre a jour l'affichage
  function afficher() {
    valeurElement.textContent = compteur;
  }

  // Gestionnaires d'evenements
  btnIncrementer.addEventListener('click', () => {
    compteur++;
    afficher();
  });

  btnDecrementer.addEventListener('click', () => {
    compteur--;
    afficher();
  });

  btnReset.addEventListener('click', () => {
    compteur = 0;
    afficher();
  });
});
```

Récapitulatif

Ce que nous avons vu :

- **HTML** : Structure (balises, attributs, sémantique)
- **CSS** : Style (sélecteurs, box model, positionnement)
- **JavaScript** : Logique (variables, fonctions, objets)
- **DOM** : Manipulation de la page
- **Événements** : Interactivité

Prochaine étape

Pratique avec un mini-projet (TD de 2h)

Ces bases sont essentielles pour Vue.js !

Points clés à retenir

1. **Séparation des préoccupations**
   - HTML = Structure
   - CSS = Présentation
   - JS = Comportement
2. **Sélecteurs et ciblage**
   - Classes pour le CSS
   - IDs avec parcimonie
   - querySelector/querySelectorAll
3. **Événements**
   - addEventListener (pas inline)
   - Objet event
   - preventDefault / stopPropagation

Ressources

Documentation :

- MDN Web Docs : `https://developer.mozilla.org`
- W3Schools : `https://www.w3schools.com`
- Can I Use : `https://caniuse.com`

Outils de développement :

- Chrome DevTools (F12)
- Firefox Developer Tools
- VS Code avec extensions

Prêts pour le TD ?

Questions ?

Pause avant le TD

Prochain : Mini-projet pratique (2h)