

CSS



HTML

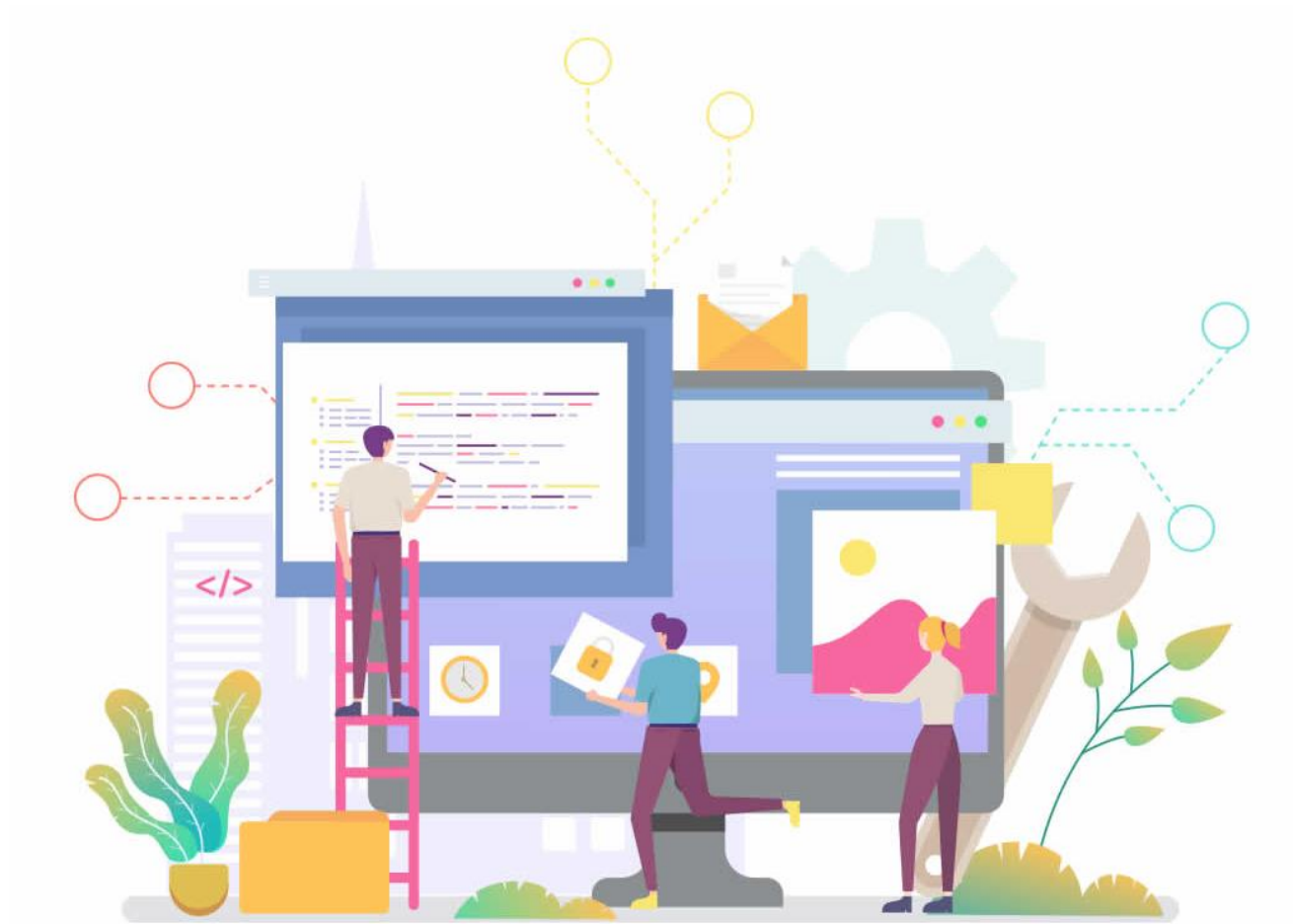


JS



FORMATION WEB BASIC : HTML, CSS, Javascript et Typescript
Mai 2025

PRÉSENTATION





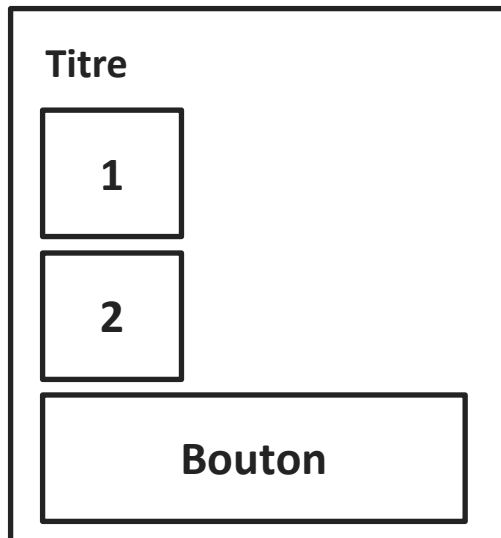
1. Présentation des technologies
2. HTML
3. CSS
4. Javascript
5. Typescript

LA BASE DU WEB

HTML



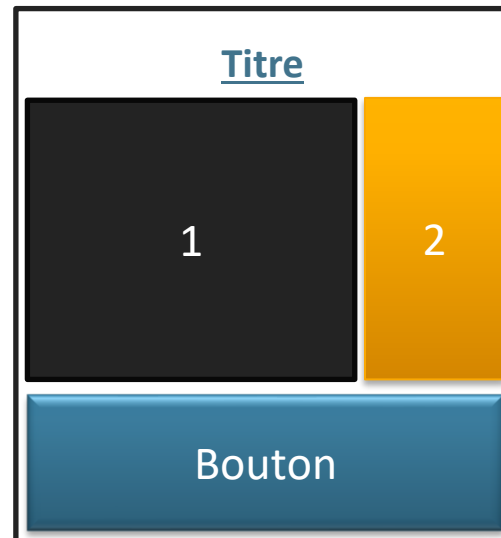
Structure



CSS



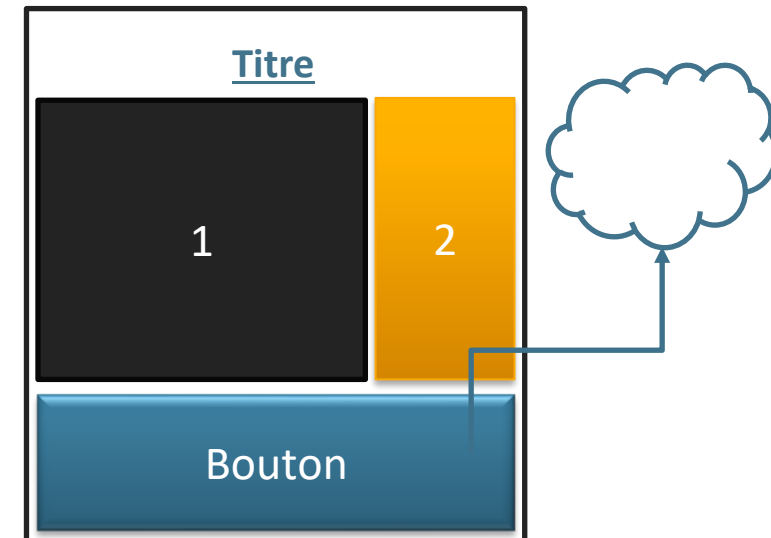
Style



JS



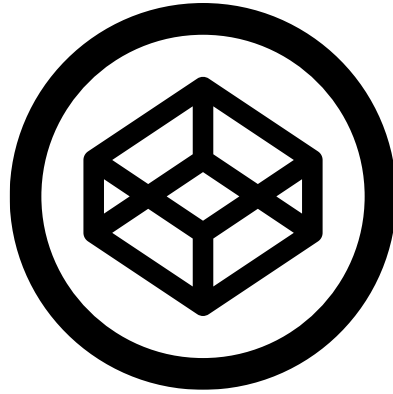
Comportement



NOS OUTILS



VS Code



Codepen IO



Chrome

HTML

- HTML (**H**yper**T**ext **M**arkup **L**anguage) est le langage standard de balisage utilisé pour structurer le contenu des pages web
- Créé en 1991 par Tim Berners-Lee, il a été la base du World Wide Web
- Utilise une syntaxe à base de **balises et leurs attributs** (comme <p>, <a>, <div>) pour décrire la structure d'un document
- Les balises HTML sont hiérarchiques et organisées en nœuds imbriqués dans un arbre DOM (Document Object Model)
- HTML n'est pas un langage de programmation, mais un **langage de description de structure**
- Évolutions majeures :
 - HTML 4 (1997) : standardisation importante, support de CSS
 - xHTML (2000) : version stricte basée sur XML
 - HTML5 (2014) : ajout d'éléments sémantiques (<header>, <section>, etc.), support natif de l'audio/vidéo, Canvas, Web Storage, ...

HTML : STRUCTURE GLOBALE

```
1  <!DOCTYPE html>
2  <html lang="fr">
3    <head>
4      <!-- Méta informations -->
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <meta name="description" content="Exemple de page HTML">
8      <meta name="author" content="Mustapha">
9
10     <!-- Titre de la page -->
11     <title>Ma page HTML exemple</title>
12
13     <!-- Favicon -->
14     <link rel="icon" href="/favicon.ico" type="image/x-icon" />
15
16     <!-- Feuilles de style -->
17     <link rel="stylesheet" href="styles.css" />
18     <link rel="stylesheet" href="https://fonts.googleapis.com/..." />
19
20     <!-- Scripts JavaScript -->
21     <script src="script.js"></script>
22   </head>
23   <body>
24
25   </body>
26 </html>
27
```

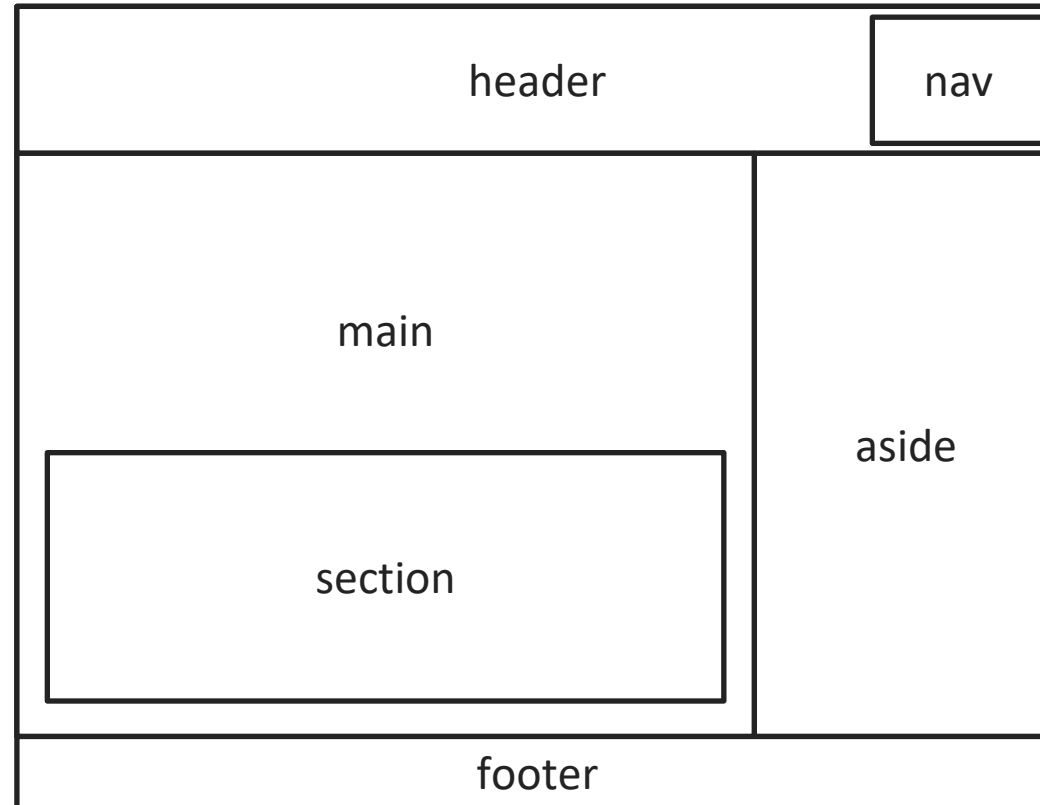
HTML : LE BODY

- **Générique** : <div> et
- **Structure** : <header>, <nav>, <main>, <section>, <article>, <aside>, <footer>
- **Texte** : <p>, de <h1> à <h6>, , <blockquote>, <cite>
- **Liste** : , , , <dl>, <dt>, <dd>
- **Média** : , <audio>, <video>, <canvas>, <svg>
- **Interaction et formulaire** : <a>, <button>, <form>, <input>, <select>, <textarea>, <label>
- **Tableau** : <table>, <thead>, <tbody>, <tfoot>, <tr>, <td>, <th>
- **Script & style** : <script>, <style>, <noscript>

<http://developer.mozilla.org/fr/docs/Web/HTML>

HTML : LA STRUCTURE

- Les balises de structure servent à organiser le contenu d'une page HTML de façon sémantique, en définissant les grandes zones comme l'en-tête, la navigation, le contenu principal, les sections, les articles ou le pied de page. (<https://html5doctor.com/downloads/h5d-sectioning-flowchart.pdf>)



HTML : LE TEXTE

- Les balises de texte permettent de structurer le contenu écrit, en créant des paragraphes, titres, mises en valeur ou citations pour faciliter la lecture.

```
1  <balise-texte>Mon texte</balise-texte>
```

- **Les titres** : de <h1> à <h6>
- **Le paragraphe** : <p>
- **Les mise en valeur** : ,
- **Les citations** : <blockquote>, <cite>
- **Les dépréciées** : <i>,

HTML : LES LISTES

- Les balises de liste sont utilisées pour présenter des collections d'éléments sous forme ordonnée, non ordonnée ou en paires terme-définition, afin de clarifier les relations entre ces éléments.

Liste ordonnée

```
1 <ol>
2   <li>Premier chapitre</li>
3   <li>Deuxième chapitre</li>
4   <li>Troisième chapitre</li>
5 </ol>
```

Liste non-ordonnée

```
1 <ul>
2   <li>Pommes</li>
3   <li>Bananes</li>
4   <li>Fraises</li>
5 </ul>
```

Paires termes-définition

```
1 <dl>
2   <dt>HTML</dt>
3   <dd>HyperText Markup Language</dd>
4   <dt>CSS</dt>
5   <dd>Cascading Style Sheets</dd>
6 </dl>
```

HTML : MEDIAS

- Les balises de média permettent d'insérer et de gérer des contenus visuels ou sonores comme des images, vidéos, dessins vectoriels ou éléments interactifs.

```
1 
```

- **Mutltimédia** : , <audio>, <video>
- **Graphismes et dessin** : <canvas>, <svg>

HTML : INTERACTION ET FORMULAIRE

- Les balises d'interaction offrent des moyens pour l'utilisateur de naviguer, soumettre des données ou interagir avec la page via des liens, boutons et formulaires.

- **Liens :**

```
1 <a href="https://google.com" target="_blank">Lien vers le moteur de recherche</a>
```

- **Formulaire :**

```
1 <form>
2   <!-- Liste de choix -->
3   <select>
4     <option>M.</option>
5     <option>Mme.</option>
6   </select>
7
8   <!-- Inputs -->
9   <label for="text-field">Nom</label>
10  <input id="text-field" type="text" /> <!-- Texte standard -->
11  <input type="password" /> <!-- Les caractères sont masqués lors de la saisie -->
12  <input type="radio" /> <!-- Bouton radio -->
13  <input type="checkbox" /> <!-- Case à cocher -->
14  <input type="file" /> <!-- Sélection de fichier sur le poste -->
15  <input type="number" /> <!-- Texte limité à des nombres (très utile en RWD) -->
16  <input type="range" /> <!-- Slider -->
17  <input type="color" /> <!-- Sélecteur de couleur -->
18  <input type="submit" /> <!-- Bouton de soumission d'un formulaire -->
19  <button>Enregistrer</button>
20 </form>
```

HTML : TABLEAU

- Les balises de tableau servent à organiser et afficher des données structurées en lignes et colonnes, facilitant la lecture de tableaux complexes.

```
1 <table>
2   <thead> <!-- Entête du tableau -->
3     <tr> <!-- Ligne -->
4       <th rowspan="2">Nom</th> <!-- Colonne d'entête -->
5       <th colspan="2">Civilité</th>
6     </tr>
7     <tr>
8       <th>Mr.</th>
9       <th>Mme.</th>
10    </tr>
11  </thead>
12  <tbody> <!-- Corps / Contenu du tableau -->
13    <tr>
14      <td>John Dorian</td> <!-- Colonne -->
15      <td>X</td>
16      <td></td>
17    </tr>
18    <tr>
19      <td>Elliott Reed</td>
20      <td></td>
21      <td>X</td>
22    </tr>
23    <tr>
24      <td>Perry Cox</td>
25      <td>X</td>
26      <td></td>
27    </tr>
28  </tbody>
29  <tfoot> <!-- Pied du tableau -->
30    <tr>
31      <td>Total</td>
32      <td>2</td>
33      <td>1</td>
34    </tr>
35  </tfoot>
36 </table>
```








HTML : TP

- Créer une page HTML comprenant :
 - Un header avec :
 - Une image (on ajoutera les attributs width et height pour limiter la taille de l'image)
 - Un titre : Mon agenda
 - Un main avec :
 - Un titre de second niveau : Ma semaine
 - Un paragraphe : avec un lorem généré
 - Un formulaire : un champ text simple avec un label « Activité », une liste avec les jours de la semaine et un bouton avec comme texte « Ajouter »
 - Un tableau avec pour entête : jour, activité et dans le corps insérer quelques lignes d'exemples
 - Un footer avec :
 - Une liste non ordonnée comprenant des liens vers 3 sites de votre choix (attention à ouvrir la page dans un nouvel onglet)

CSS

- CSS (**C**ascading **S**tyle **S**heets) est un langage né en 1996 pour définir **le style et la mise en forme des documents HTML** (couleurs, polices, marges, disposition...)
- Il repose sur un formalisme déclaratif basé sur des règles ciblant **des éléments HTML** via des **sélecteurs**
- Il sépare le **contenu** (HTML) de la **présentation**, facilitant la maintenance et la cohérence visuelle
- Il évolue par niveaux (CSS1, CSS2, **CSS3**), mais désormais il suit un modèle modulaire en amélioration continue
- Les grandes avancées récentes incluent : Flexbox, Grid, variables CSS, animations, media queries, et le support croissant de la thématisation et de l'accessibilité

CSS : STYLE ET MISE EN FORME

-  **Couleurs et arrière-plan** : color, background-color, background-image, background-size, background-position, opacity
-  **Texte et typographie** : font-family, font-size, font-weight, font-style, line-height, text-align, text-decoration, text-transform, letter-spacing, word-spacing
-  **Boîte et dimensions** : width, height, padding, margin, border, border-radius, box-sizing, overflow
-  **Positionnement et affichage** : display, visibility, position, top, right, bottom, left, z-index, float, clear
-  **Images et médias** : object-fit, object-position, aspect-ratio, background-repeat
-  **Interactivité et effets visuels** : cursor, pointer-events, transition, animation, transform, filter, box-shadow
-  **Responsive** : media queries

<http://developer.mozilla.org/fr/docs/Web/CSS>

CSS : LES SÉLECTEURS

- De type : div, p, h1
- De classe : .ma-classe
- D'identifiant : #mon-id
- Universel : *
- D'attribut : [type="text"], [disabled]
- Combinés :
 - Descendant : div p
 - Enfant direct : div > p
 - Frère adjacent : div + p
 - Frères généraux : div ~ p
- Pseudo-classes : :hover, :focus, :first-child, :nth-child(2n)
- Pseudo-éléments : ::before, ::after

```
1 <div></div>
```

```
1 <div class="presentation"></div>
```

```
1 <div id="main-content"></div>
```

```
1 <input type="text" />
```

```
1 div id="1">
2   <div id="2">
3     <p></p> <!-- Enfant direct de 2 et le descendant de 1 et 2 -->
4   </div>
5 </div>
6 <p></p> <!-- Frère adjacent et général de 1 -->
7 <p></p> <!-- Frère général de 1 -->
```

CSS : LES UNITÉS ET VALEURS

- **Unités absolues :**

- **px** (pixels),
- **cm** (centimètres), **mm** (millimètres), **in** (pouces)

- **Unités relatives :**

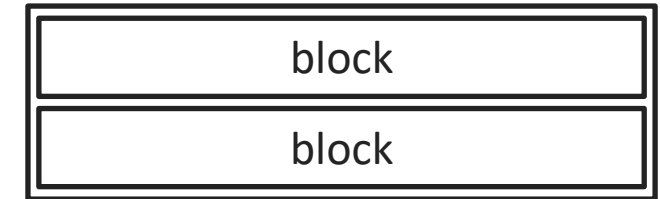
- **em** (relative à la taille de police parente), **rem** (relative à la taille de police racine),
- **%** (pourcentage relatif à un conteneur),
- **vw/vh** (viewport width/height, relatif à la taille de la fenêtre)

- **Valeurs de couleurs :**

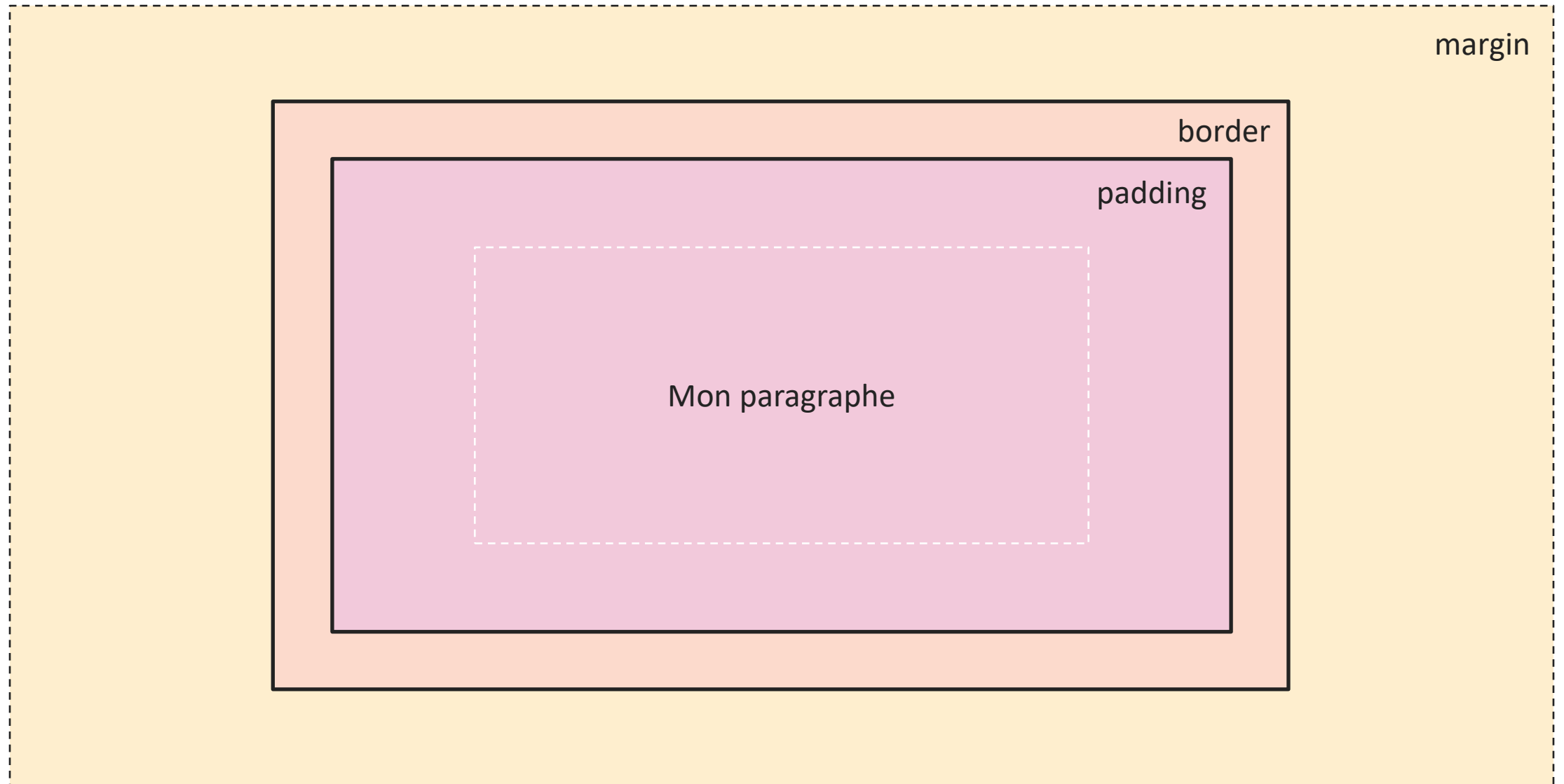
- Noms prédéfinis (red, blue, green, etc.)
- Hexadécimal (**#RRGGBB** ou raccourci **#RGB**)
- **RGB / RGBA** (rgb(255,0,0), rgba(255,0,0,0.5))
- **HSL / HSLA** (hsl(120, 100%, 50%), hsla(120, 100%, 50%, 0.5))

CSS : LE FLUX

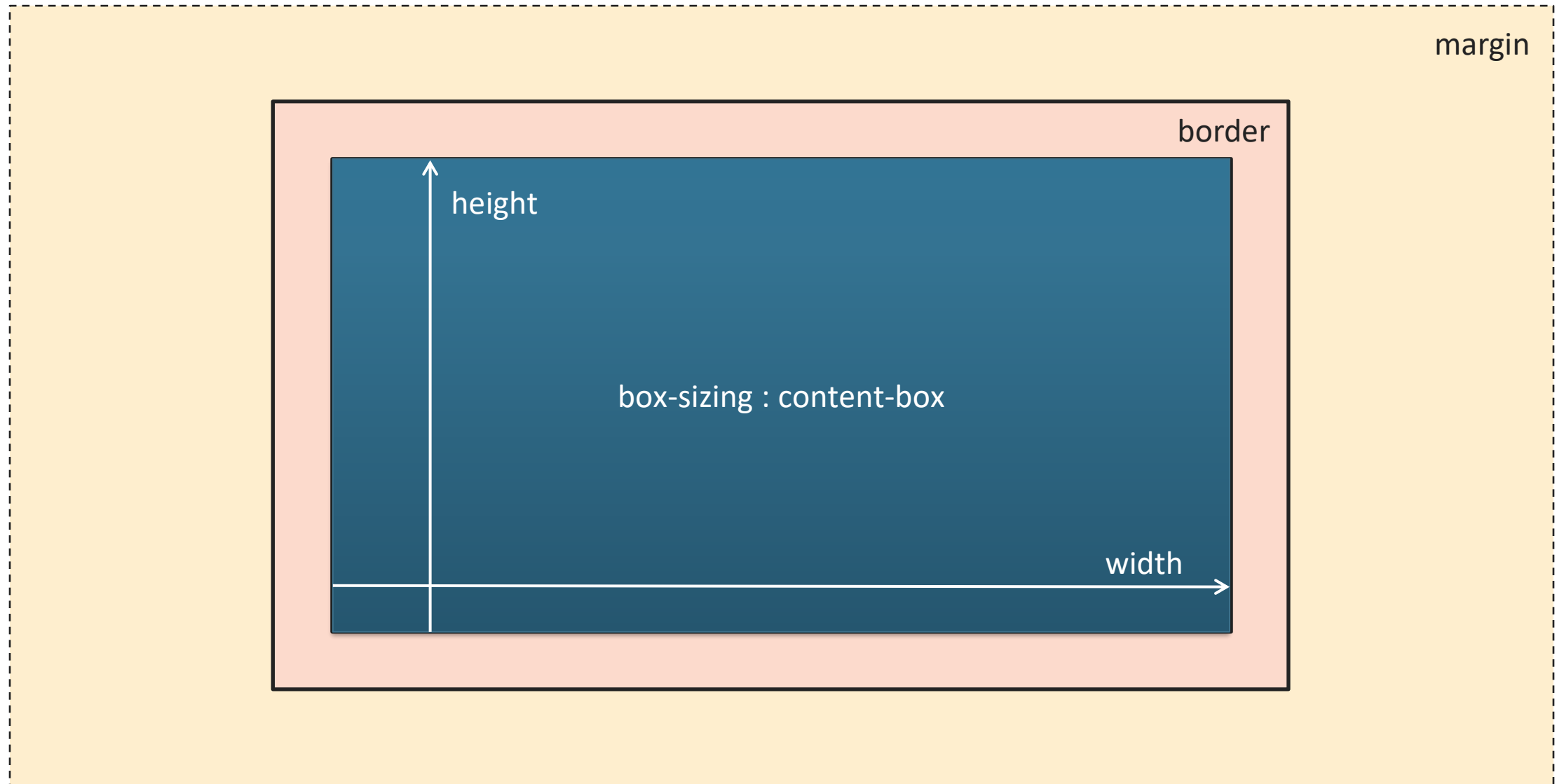
- Comportement de rendu :
 - **Block** : un élément de type block occupe toute la largeur disponible et commence sur une nouvelle ligne.
<div>, <p>, <section>
 - **Inline** : un élément inline s'affiche dans le flux du texte, sans retour à la ligne, et ne prend que la place nécessaire.
, <a>,
- Rupture de flux :
 - **Display** : flex, grid, inline-block, ...
 - **Position** : absolute, static, ...



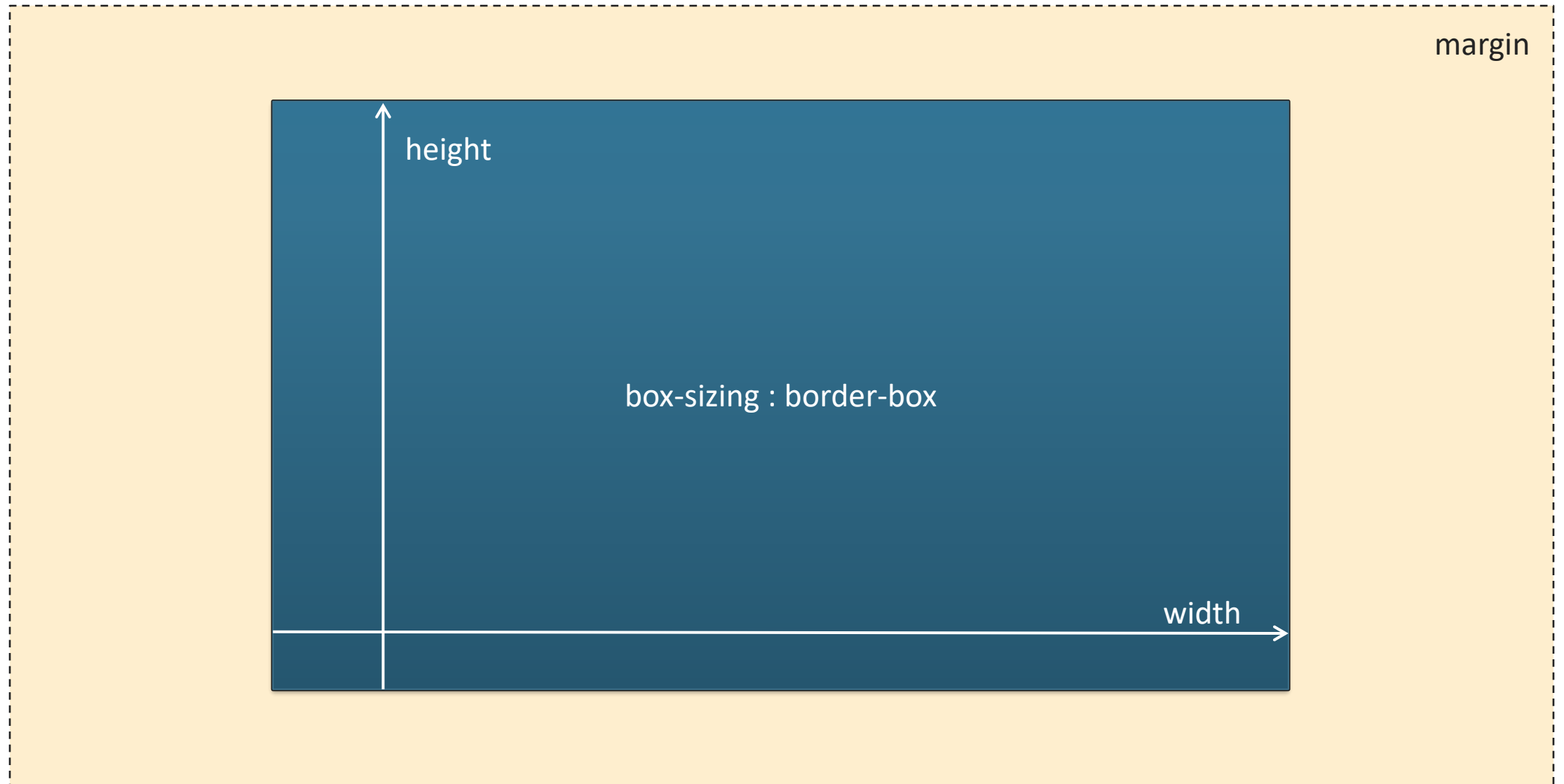
CSS : LA BOÎTE



CSS : LA BOÎTE



CSS : LA BOÎTE



CSS : INTÉGRATION EN HTML

- **Inline** : ajouter un attribut style directement sur un élément HTML pour appliquer du CSS ponctuel

```
1 <h1 style="text-align: center; color: red; font-weight: bold;">Titre de la page</h1>
```

- **Dans le style** : utiliser une balise style dans le head pour écrire du CSS directement dans la page

```
1 <head>
2   <style>
3     h1 {
4       text-align: center;
5       color: red;
6       font-weight: bold;
7     }
8   </style>
9 </head>
```

- **Par lien externe** : insérer une balise link dans le head pour charger un fichier CSS externe

```
1 <link rel="stylesheet" href="styles.css" />
```

```
1 h1 {
2   text-align: center;
3   color: red;
4   font-weight: bold;
5 }
```


CSS : TP

- Créer une feuille de style css à côté du fichier HTML et référencer cette feuille par une balise link dans le head. Dans cette feuille la première instruction est : Définir tous les éléments en border-box
- Conjointement avec le formateur :
 - Agencer par flex notre header, lui donner toute la largeur de la page et lui mettre un fond noir
 - Limiter la taille de l'image du header dans le CSS et non plus dans l'HTML
 - Mettre en gras le titre du header et lui mettre une police Arial blanche de taille 22px
 - Souligner et enlever le gras du titre du contenu
 - Réagencer et styliser le formulaire pour avoir sur une ligne le label du champ et le champ, sur une deuxième ligne la liste et sur une dernière ligne le bouton. Le formulaire doit être encadré d'une bordure arrondie (5px) et avoir une largeur de 800px au minimum. Le bouton doit être aligné le plus à droite possible
 - Dans le tableau, les cellules ont une bordure partagée et de couleur grise (#CCCCCC) avec une opacité de 30%. Les colonnes d'entête seront en gras.
 - Dans le pied de page, la liste sera présentée horizontalement dans toute la largeur de la page. Le pied de page sera en fond noir et tous les éléments descendants auront une police blanche
 - Bonus :
 - Une ligne sur deux du corps du tableau aura un fond bleu clair
 - La liste du pied de page n'affichera plus de puce

JAVASCRIPT

- JavaScript est un **langage de programmation non typé** créé en 1995 par Brendan Eich pour rendre les **pages web interactives côté client** (navigateur).
- C'est un **langage interprété**, dynamique et multi-paradigme (impératif, fonctionnel, orienté objet).
- Initialement simple, il a beaucoup évolué avec les normes **ECMAScript** (ES5, ES6 en 2015 et suivantes) qui ont apporté **classes, modules, promesses**, async/await, ...
- JavaScript est aujourd'hui utilisé aussi côté serveur (**Node.js**), dans les applications mobiles, desktop, et même pour l'IA.
- Il permet de manipuler le **DOM**, gérer les **événements**, faire des requêtes réseau, et bien plus.

JAVASCRIPT : NOTIONS CLÉS

- **Variables et types** : comprendre comment déclarer (let, const, var) et utiliser les types de base (number, string, boolean, null, undefined)
- **Opérateurs et expressions** : opérations mathématiques, logiques, comparaison, affectation
- **Structures de contrôle** : conditions (if, else, switch), boucles (for, while)
- **Fonctions et classes** : définir des classes, déclarer et appeler des fonctions, comprendre les paramètres et le retour
- **Manipulation du DOM** : sélectionner et modifier des éléments HTML avec document.querySelector et element.innerHTML ou element.style
- **Gestion des événements** : écouter et réagir aux actions utilisateur comme clics, saisie clavier, changement de valeur d'un input, ...

JAVASCRIPT : VARIABLES ET TYPES

- **Variables** : déclarées avec `let` (modifiable), `const` (constante), ou `var` (ancienne façon, à éviter)

```
1 const delaiRafrachissement = 60;  
2 let nom = "";
```

- **Types primitifs** : `number` (ex: 42), `string` (ex: "texte"), `boolean` (true/false), `null` (absence de valeur), `undefined` (non défini), `symbol` (identifiant unique)

```
1 let isActive = false;  
2 let panier = null;
```

- **Types complexes** : `object` (tableaux, objets, fonctions)

```
1 const employee = {  
2   nom: 'Robert',  
3   age: 38  
4 }
```

```
1 const list = [  
2   'Pommes',  
3   'Bananes'  
4 ]
```

- **Typage dynamique** : une variable peut contenir des valeurs de types différents au cours du temps.

JAVASCRIPT : VARIABLES ET TYPES

- **String :**

- `.length`, `.toLowerCase()`, `.toUpperCase()`
- `.includes()`, `.indexOf()`, `.slice()`, `.substring()`
- `.replace()`, `.split()`, `.trim()`

- **Number :**

- `parseInt()`, `parseFloat()`, `.toFixed()`, `.toString()`
- `Math.round()`, `Math.floor()`, `Math.ceil()`, `Math.random()`, `Math.abs()`

- **Array :**

- `.length`, `.push()`, `.pop()`, `.shift()`, `.unshift()`
- `.slice()`, `.splice()`, `.indexOf()`, `.includes()`
- `.map()`, `.filter()`, `.reduce()`, `.forEach()`, `.find()`

JAVASCRIPT : OPÉRATEURS ET EXPRESSIONS

- **Opérateurs arithmétiques** : +, -, *, /, % (modulo), ** (puissance)

```
1 3 + 5 // Addition = 8
2 "3" + "5" // Concaténation = "35"
```

- **Opérateurs d'assignation** : =, +=, -=, *=, /=, %=

```
1 let a = 3;
2 a += 5 // a vaut 8
```

- **Opérateurs de comparaison** : == (égalité souple), === (égalité stricte), !=, !==, <, >, <=, >=

```
1 14 == "14" // Vrai
2 14 === "14" // Faux
```

- **Opérateurs logiques** : && (et), || (ou), ! (non)

```
1 isActive = !isActive
```

JAVASCRIPT : STRUCTURES DE CONTROLE

- **Conditionnelles** : if, else if, else, switch, case, default

```
1  if(isActif && !isEmpty) {  
2    // ...  
3  } else if (isEmpty) {  
4    // ...  
5  } else {  
6    // ...  
7  }
```

```
1  switch(state) {  
2    case 'NEW': {  
3      // ...  
4      break;  
5    }  
6    case 'OLD': {  
7      // ...  
8      break;  
9    }  
10   default: {  
11     // ...  
12   }  
13 }
```

- **Boucles** : for, while

```
1  // Itération classique  
2  for(let i = 0; i < 10; i++) { }  
3  
4  // For of  
5  for(let element of list) { }  
6  
7  // For in  
8  for(let prop in objet) { }
```

```
1  // Boucle while  
2  while(isActif) { }  
3  
4  // Boucle do ... while  
5  do { } while (isActif)
```

JAVASCRIPT : FONCTIONS ET CLASSES

- **Fonctions**

- `function` : mot-clé pour déclarer une fonction nommée
- Fonctions fléchées : syntaxe concise (`const f = (...) => { ... }`)
- Paramètres : transmis entre parenthèses (`function(x, y) { ... }`)
- Valeur de retour : via `return` (ou `undefined` si absent)
- Paramètres par défaut : `function(x = 0) { ... }`

- **Classes**

- `class` : mot-clé pour définir une classe
- `constructor` : méthode spéciale appelée à l'instanciation
- `this` : fait référence à l'instance courante
- Méthodes : déclarées sans `function` dans la classe
- Héritage : `class B extends A { super(); ... }`
- Méthodes statiques : `static maMethode() { }`
- Encapsulation (ES2022+) : propriétés privées avec `#nom`

```
1 // Fonction classique
2 function add(a, b = 0) {
3     return a + b;
4 }
5
6 // Arrow function
7 add = (a, b = 0) => a + b;
```

```
1 export class Vehicule {
2     #nom;
3     #matricule;
4
5     constructor(nom, matricule) {
6         this.#nom = nom;
7         this.#matricule = matricule;
8     }
9
10    hasToBeDestroyed(ref) {
11        return ref === this.#matricule;
12    }
13
14    static generateEmptyVehicule() {
15        return new Vehicule('', '');
16    }
17 }
```


JAVASCRIPT : MANIPULATION DU DOM

- **Sélection d'éléments HTML**

- `document.querySelector(sel)` : sélectionne le premier élément correspondant au sélecteur CSS
- `document.querySelectorAll(sel)` : sélectionne tous les éléments correspondants au sélecteur CSS

- **Modifier le contenu**

- `element.innerHTML` : modifie le contenu HTML
- `element.textContent` : modifie le texte brut (sans interprétation HTML)

- **Modifier le style**

- `element.style.propriété` : change un style inline (`element.style.color = 'red'`)
- `element.classList.add()` / `.remove()` / `.toggle()` : ajoute ou retire des classes CSS dynamiquement

- **Autres manipulations utiles**

- `element.setAttribute(nom, valeur)` : modifie un attribut HTML
- `element.appendChild(nœud)` : ajoute un élément dans le DOM
- `element.remove()` : supprime un élément du DOM

JAVASCRIPT : GESTION DES ÉVÈNEMENTS

- **Écouter des événements**

- `element.addEventListener(type, callback)` : attache un écouteur d'événement
`button.addEventListener('click', handleClick)`

- **Types d'événements courants :**

- `click` : clic souris
- `keydown`, `keyup` : appui / relâchement clavier
- `change` : modification (ex : case à cocher, liste déroulante)

- **Gérer l'événement dans la fonction**

- Le callback reçoit un objet event
- Accès à l'élément déclencheur : `event.target`
- Empêcher le comportement par défaut : `event.preventDefault()`

JAVASCRIPT : TP

- Mettre le bouton en disabled par défaut
- Dans le fichier HTML, intégrer une balise script en bas de page avec l'attribut defer. Dans cette balise, intégrer les fonctionnalités suivantes :
 - Lorsque le champ « Activité » est modifié, s'il la valeur n'est pas vide, activer la bouton « ajouter »
 - Au clic sur ajouter, ajouter une ligne dans le tableau avec ces nouvelles valeurs.
 - Au clic sur ajouter, remettre à vide le champ « Activités »
 - Dans une div après le tableau, générer en un tableau comprenant les jours de la semaine, et calculant pour chaque jour le nombre d'activités. Ce tableau doit se mettre à jour à chaque nouvelle ligne ajoutée
- Bonus :
 - Ajouter un bouton pour supprimer une ligne du tableau
 - Enregistrer et charger les données du tableau depuis le localStorage

TYPESCRIPT

- TypeScript est un surensemble de JavaScript créé par Microsoft en 2012, qui ajoute un système de typage statique facultatif.
- Il suit une syntaxe proche de JavaScript, mais permet de détecter les erreurs à la compilation, grâce à des annotations de types, des interfaces, des classes, des énumérations, etc.
- Le code TypeScript est transcompilé en JavaScript, pour être exécuté dans n'importe quel navigateur ou environnement Node.js.
- TypeScript est devenu la norme dans de nombreux projets modernes (comme Angular), grâce à ses outils puissants d'autocomplétion, de refactoring et de sécurité de code.
- Il évolue rapidement, avec des versions fréquentes intégrant de nouvelles fonctionnalités (types conditionnels, inférence améliorée, tuples variadiques, etc.).

TYPESCRIPT

```
1  export class Vehicule {
2
3      private nom: string;
4      private matricule: string;
5
6      constructor(nom: string, matricule: string) {
7          this.nom = nom;
8          this.matricule = matricule;
9      }
10
11      hasToBeDestroyed(ref: string): boolean {
12          return ref === this.matricule;
13      }
14
15  }
16
17  interface Animal {
18      nom: string;
19      race?: string;
20  }
21
22  export class Chien implements Animal {
23      nom: string;
24      isBarking: boolean;
25
26      whoIsBarking(): string | boolean {
27          return this.isBarking && this.nom;
28      }
29
30  }
```



QUESTIONS / RÉPONSES

RESSOURCES

- <https://developer.mozilla.org/fr/docs/Web>
- <https://www.w3.org/>
- <https://html5doctor.com/downloads/h5d-sectioning-flowchart.pdf>
- <https://flexboxfroggy.com/#fr>
- <https://cssgridgarden.com/#fr>
- <https://codepen.io/>
- https://openweb.eu.org/articles/cascade_css
- <https://codepen.io/cyrilvernier/post/preseance-css>
- <https://caniuse.com/>