



CONCEVOIR DES INTERFACES GRAPHIQUES AVEC HTML5, CSS3 ET JAVASCRIPT

Victor Dupré





Panorama

Programme

1. Salutations
2. Introduction & un peu d'histoire
3. Le développement Web
4. HTML5
5. CSS3
6. Premiers exercices...



1. Salutations

VICTOR DUPRÉ

FR.VICTOR.DUPRE@LIVE.FR

FREELANCE REACT.JS & REACT
NATIVE

2. Introduction

A) Terminologie

- **HTTP** : (HyperText Transfer Protocol) protocole de communication utilisé pour le web
- **HTTPS**: variante avec authentication et chiffrement
- **URL** : (Uniform Resource Locator) chaîne de caractères décrivant l'emplacement d'une ressource

[protocole]://[nom d'hôte]/[chemin]

<http://example.com/un/chemin/page.html>

- 
- **Hyperlien** : (lien) élément dans une ressource associé à une URL
 - **HTML** : (HyperText Markup Language) language de description d'un document (titres, paragraphes, images, liens, etc.)
 - **Serveur** : hôte sur lequel fonctionne un logiciel serveur
 - **Site web** : ensemble de pages web et de ressources liées et accessible par une adresse web

B) World Wide Web

Un système **hypertexte** public sur internet

Inventé par Tim Berners-Lee et Robert Cailliau

Confusion entre web et internet

C) Naissance du Web

En **1989**: Développé au CERN

Création des technologies: **HTML, URL & HTTP**

En **1992**: 26 sites en lignes ..

En **2014**: plus d'un milliard

D) Quelques dates

En **1995**: Début de la guerre des navigateurs

En **1996**: CSS1

En **1997**: HTML 3.2 & 4

En **1998**: Google & Mozilla, CSS2

En **2001**: Wikipédia

En **2004**: Facebook, Web 2.0

E) Trafic Web

Sur le Web

61% des visiteurs sont des robots

Seulement 39% d'humains ..

Des robots ?

Des bots d'indexation

Des bots DCMA

etc..

F) W3C

World Wide Web Consortium

Organisme de normalisation à but non lucratif,
fondé en octobre 1994

Consortium international (434 entreprises
partenaires)



G) L'universalité

Accessible avec les outils les plus divers

Protocoles universel

Normalisation via le W3C

H) La décentralisation

Aucune organisation imposée

Hyperlien universel

Absence de registre centralisé (hormis DNS)

Favorise la création de sites et notamment d'annuaire et de moteur de recherche

Inconvénient : 404

I) Evolution des technologies

Avant

- HTML/CSS uniquement
- Un peu de Javascript au cas où
- Page de quelques ko

Maintenant

- Application complexe
- Page moyenne de 2.3Mo (autant que Doom)

3. Développement Web

A) Site web statique

HTML/CSS uniquement

- Pas de langage de programmation mais de présentation de documents
- Pas d'interactivité possible avec le visiteur (soumission de formulaire, authentication, ..)
- Aucun dynamisme
- Pas de lecture de BDD
- Ajout d'une page entraîne la modication de toutes les pages

B) Site web dynamique

- Programme serveur qui va générer les pages HTML pour nous

Plusieurs technos disponibles:

- Serveur HTTP Apache, Nginx, IIS, ..
- Langage PHP, ASP, Ruby, Python, Java, ..
- BDD MySQL, SQLServer, SQLite, PostGreSQL, Oracle, DB2, ..

C) Pour commencer

Présentation de documents grâce au couple HTML/CSS

Ressource locale

- Affichage de notre document dans un navigateur
- Notre document n'est pas présent sur le réseau

Et surtout, des outils **corrects** !

4. HTML5



A) HyperText Markup Language

Format de données conçu pour représenter les pages web

Langage de balisage textuel, peut être édité dans n'importe quel éditeur de texte

Page web -> document HTML -> ensemble de balise HTML

B) Rendering engines

Programme qui rend du contenu balisé (HTML)
avec des informations de formatage (CSS)

Gecko/Quantum (Mozilla Firefox)

WebKit (Safari, Opera)

Blink (Chrome, Opera, ..)

Trident (IE, ..)

C) HyperText Markup Language

Document HTML: ensemble **entête & corps**

Par convention l'extension donnée au fichier est **.html**

Le fichier de base d'un site web (statique) se nomme **index.html**

Un document HTML est un simple fichier contenant du texte formaté avec des balises HTML

D) Bonnes pratiques....

index.html 

Index.html 

Accueil.html 

accueil.html 

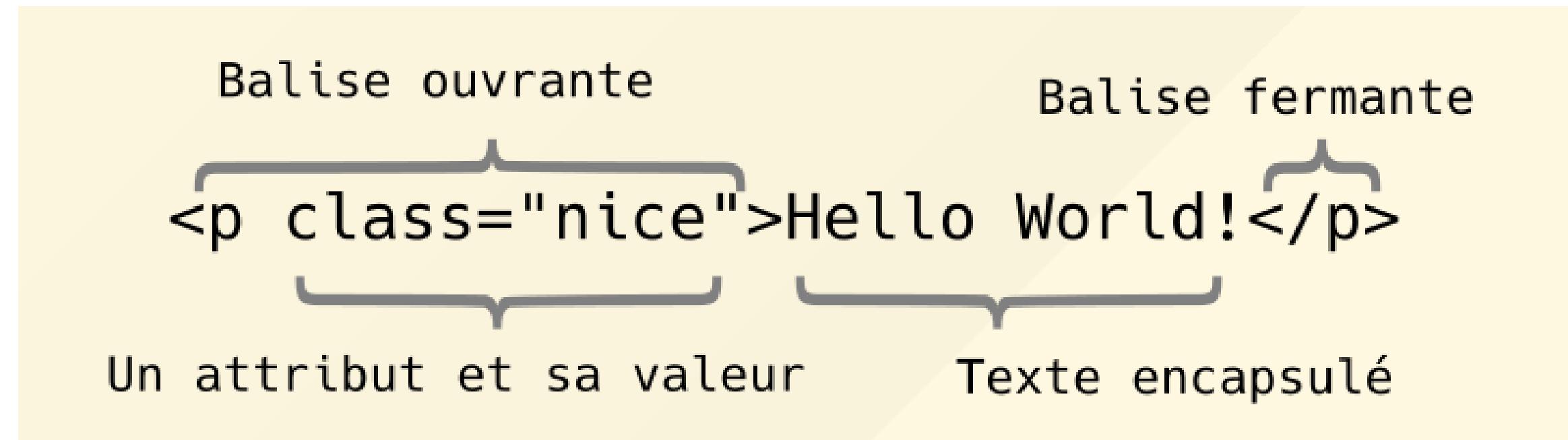
mon_fichier.txt 

Mon spuer Fichier.txt 

filename 

Mon dssier 

E) Anatomie d'une balise HTML



Tout texte entre les chevrons (en anglais brackets) < et > est considéré une balise.

<p> est une balise ouvrante, </p> est la balise fermante qui lui correspond; elles délimitent le texte concerné.

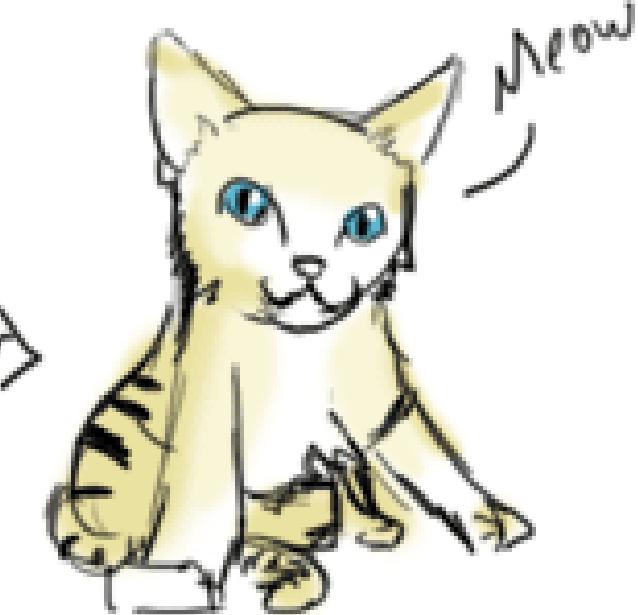
Always close your tags!

<lightsaber>



</lightsaber>

Note how the closing tag
stops the saber.



<lightsaber>



without the closing tag,
kittens will die.



Protect the kittens and close your tags.

E) Anatomie d'une balise HTML

On peut appliquer plusieurs balises au même texte, à condition de respecter la règle d'emboîtement:

- toute balise B ouverte à l'intérieur d'une balise A doit également être fermée à l'intérieur de A.

```
<p>Emboîtement <em>correct</em></p>

<p>Emboîtement <em>incorrect</p></em>
```

E) Anatomie d'une balise HTML

Certaines balises en plus du contenu textuel ont besoin d'information supplémentaire (qui n'apparaît pas dans le document).

Cette information est donnée par des attributs, qui ont la forme nom=valeur et sont placés entre les chevrons de la balise ouvrante, après son nom.

```
<lightsaber style="color:green;">  </lightsaber>
```

```
<lightsaber style="color:blue;">  </lightsaber>
```

```
<lightsaber style="color:red;">  </lightsaber>
```

style attribute has its value in quotes!

E) Anatomie d'une balise HTML

Certaines balises particulières n'attendent pas de contenu textuel.

Ces balises vides n'ont donc pas de balise fermante correspondante :

```

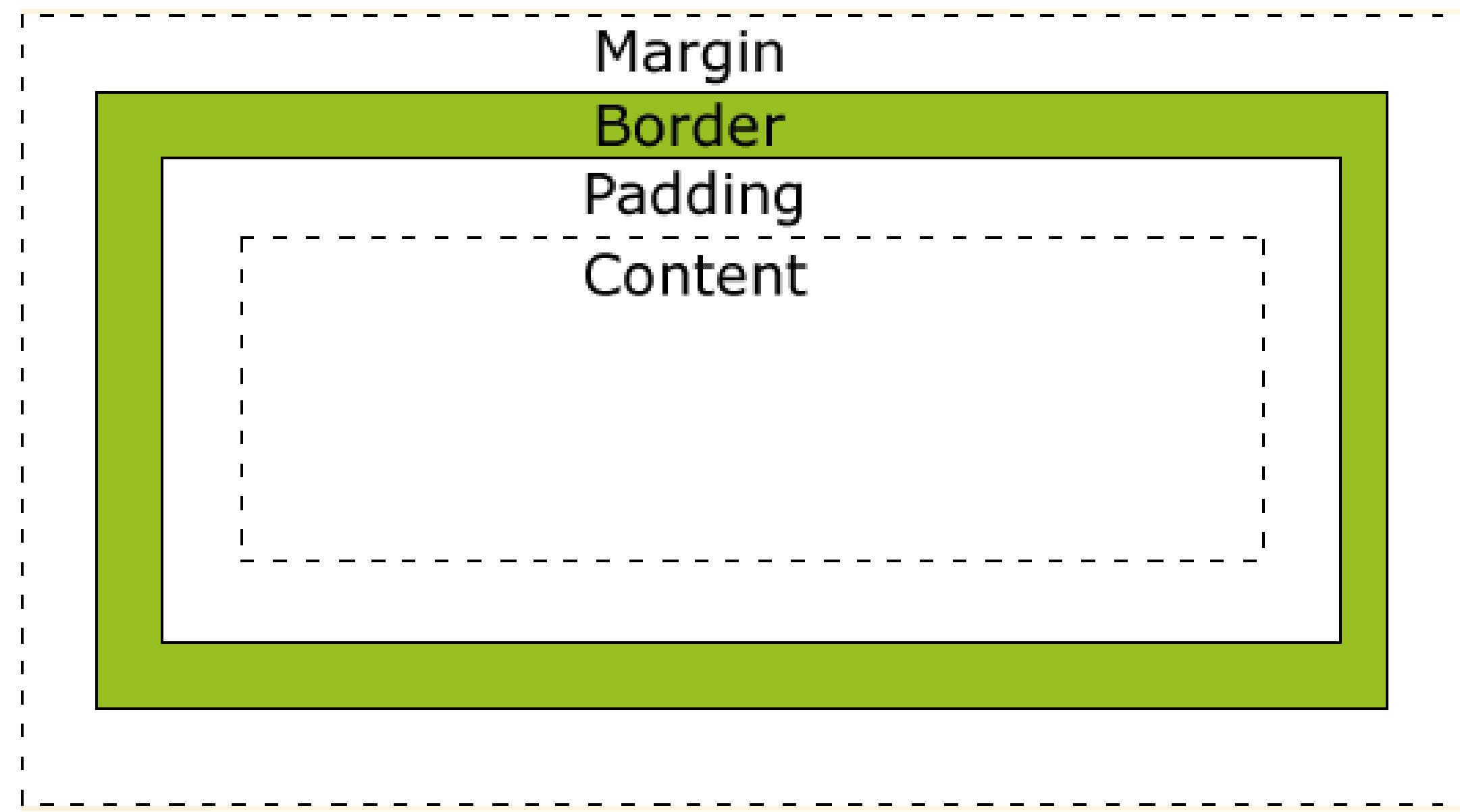
```

HTML considère tous les caractères d'espacements (espaces, retours à la ligne...) comme des séparations entre mots, et les affiche comme une simple espace.

Cela donne de la souplesse dans la mise en page du code HTML (notamment en utilisant l'indentation, comme en programmation).

E) Anatomie d'une balise HTML

Présentation de la boite autour de notre balise, marge interne et externe et la bordure.



⚠ ATTENTION ⚠

Depuis la version 4, HTML vise à décrire la structure logique du document, c'est à dire le fond et non la forme.

La mise en forme (qu'on appelle parfois structure physique) est gérée par une feuille de style qui sera décrite en CSS.

Cette séparation est indispensable pour assurer l'adaptabilité du contenu, et cette adaptabilité est incontournable dans un environnement ouvert comme le Web.

F) Squelette de base HTML

```
<!DOCTYPE html>
```

Un marqueur pour identifier le type de standard en début de fichier

```
<head></head>
```

Une entête

```
<body></body>
```

Un corps

F) Squelette de base HTML

```
<!DOCTYPE html>
<html>
    <head></head>
    <body></body>
</html>
```

G) L'entête "HEAD"

Sert à indiquer le titre du document;

Des meta-données sur celui-ci (auteur, date de création, de maj, etc..)
qui ne sont pas directement affichés;

Des styles, des scripts, etc..

Ce sont des informations sur le document, peu de balises y sont autorisées, son contenu se sera jamais affiché sur la page !

G) L'entête "HEAD"

```
<title>Hello Title!</title>
```

Indique ... le titre du document

```
<link rel="stylesheet" type="text/css"  
      href="assets/style.css" />
```

Ajoute un lien vers une feuille de style CSS

H) L'encodage

Il existe différentes normes pour coder les accents dans les chaînes de caractère.

- 👉 utf-8 est la plus récente
- 👉 par défaut latin1 est la norme historique pour les langues occidentales
- 👉 L'utilisation d'utf-8 est fortement recommandée de nos jours !

I) Squelette de base HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Hello HtmlWORLD!</title>
        <meta charset='UTF-8' />
    </head>
    <body></body>
</html>
```

J) Le corps "Body"

- 👉 Contient l'ensemble de la page qui sera visible dans le navigateur
- 👉 C'est le fond de notre document, CSS en sera la forme
- 👉 Les données sont encapsulées dans des balises correspondant au type de la donnée (titre, paragraphe, image, liste, etc..)

Un document typique est une séquence de :

- titres (ou en-tête, en anglais heading),
- et de paragraphes.

K) Titres et paragraphes en HTML

Les paragraphes sont délimités par la balise **<p>**

Les titres sont délimités par les balises **<h1>, <h2>, ..., <h6>**

```
<h1>Ceci est un titre</h1>
<p>Ceci est un paragraphe.</p>
<p>Ceci est un deuxième paragraphe.</p>
```

K) Titres et paragraphes en HTML

```
<h1>Ma dissertation</h1>
<h2>Thèse</h2>
  <p>Paragraphe d'introduction</p>
  <h3>Argument 1</h3>
    <p>...</p> <p>...</p> <p>...</p>
  <h3>Argument 2</h3>
    <p>...</p> <p>...</p> <p>...</p>
<h2>Antithèse</h2>
  <h3>Contre-argument 1</h3>
    <p>...</p> <p>...</p> <p>...</p>
  <h3>Contre-argument 2</h3>
    <p>...</p> <p>...</p> <p>...</p>
<h2>Synthèse</h2>
  <p>...</p> <p>...</p> <p>...</p>
```

Le contrôle de passage à la ligne

Apprenez à gérer les sauts de ligne dans votre contenu HTML.

```
<!DOCTYPE html>
<html>
<head>
    <title>Mon Deuxième Document</title>
</head>
<body>
    <p>Ceci est un paragraphe.<br>
    Voici une nouvelle ligne après le saut de ligne.<br>
    Encore une autre ligne après un autre saut de ligne.</p>
</body>
</html>
```

Le formatage du texte, l'alignement

Découvrez comment formater le texte et aligner le contenu dans vos pages

```
<!DOCTYPE html>
<html>
<head>
    <title>Mon Troisième Document</title>
</head>
<body>
    <p><em>Texte en italique</em></p>
    <p><strong>Texte en gras</strong></p>
    <p><u>Texte souligné</u></p>
    <p style="text-align: center;">Ce paragraphe est centré.</p>
    <p style="text-align: right;">Ce paragraphe est aligné à droite.</p>
</body>
</html>
```

La taille, la couleur et la police

Personnalisez l'apparence du texte en modifiant sa taille, sa couleur et sa police dans HTML.

```
<!DOCTYPE html>
<html>
<head>
    <title>Mon Quatrième Document</title>
</head>
<body>
    <p style="font-size: 20px;">Ce texte est plus grand.</p>
    <p style="color: blue;">Ce texte est de couleur bleue.</p>
    <p style="font-family: Arial, sans-serif;">Ce texte utilise la police .</p>
</body>
</html>
```

Les caractères spéciaux

Utilisez les entités HTML pour insérer des caractères spéciaux dans votre contenu.

```
<!DOCTYPE html>
<html>
<head>
    <title>Mon Quatrième Document</title>
</head>
<body>
    <p style="font-size: 20px;">Ce texte est plus grand.</p>
    <p style="color: blue;">Ce texte est de couleur bleue.</p>
    <p style="font-family: Arial, sans-serif;">Ce texte utilise la police .</p>
</body>
</html>
```

Les commentaires

Apprenez à ajouter des commentaires pour rendre votre code HTML plus lisible.

```
<!DOCTYPE html>
<html>
<head>
    <title>Mon Sixième Document</title>
</head>
<body>
    <!-- Ceci est un commentaire. Il ne sera pas affiché à l'écran. -->
    <p>Ceci est un paragraphe.</p>
</body>
</html>
```

L) Sections

👉 Les titres définissent en fait une structure de plus haut niveau :

- chaque titre indique le début d'une section,
- qui se termine au prochain titre de même niveau;

👉 Une section contient donc :

- un titre,
- des paragraphes,
- une ou plusieurs section(s) de niveau suivant.

L) Sections

- 👉 Jusqu'à HTML 4.01, la structuration en sections était laissée implicite.
- 👉 Depuis HTML5, on est encouragé à utiliser la balise **<section>** ,
- 👉 d'autant que les anciens navigateurs l'ignoreront purement et simplement.

L) Sections

```
<h1>Ma dissertation</h1>
<section>
  <h2>Thèse</h2>
  <p>Paragraphe d'introduction</p>
  <section>
    <h3>Argument 1</h3>
    <p>...</p> <p>...</p> <p>...</p>
  </section>
  <section>
    <h3>Argument 2</h3>
    <p>...</p> <p>...</p> <p>...</p>
  </section>
</section>
<section>
  <h2>Antithèse</h2>
</section>
```

M) Sections spécialisées

Les balises suivantes représentent des sections avec une sémantique particulière.

-	
<code><main></code>	contenu principal de la page
<code><article></code>	contenu auto-suffisant
<code><aside></code>	encart
<code><header></code>	en-tête de la section parent (ou du doc)
<code><footer></code>	pied de la section parent (ou du doc)
<code><nav></code>	section contenant des liens (navigation)

N) Listes

Une liste est un paragraphe d'un type particulier, contenant une énumération d'éléments.

Liste non ordonnée

```
<ul>
  <li>sucre</li>
  <li>céréales</li>
  <li>lait</li>
</ul>
```

Liste ordonnée

```
<ol>
  <li>sucre</li>
  <li>céréales</li>
  <li>lait</li>
</ol>
```

O) Squelette de base HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello HtmlWORLD!</title>
    <meta charset='UTF-8' />
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>May the Force be with you!</p>
  </body>
</html>
```

P) Span et div

HTML fournit deux balises neutres (c.à.d. sans sémantique particulière) :

**** qui s'affiche en mode inline, et
<div> qui d'affiche en mode block.

Ces éléments sont utiles pour rajouter de la structuration au document, notamment en portant des classes personnalisées. Il est toutefois préférable d'utiliser une balise sémantique proche du contenu qu'une balise neutre.

Les tableaux

Apprenez à créer un tableau simple en HTML pour afficher des données tabulées.

```
<!DOCTYPE html>
<html>
<head>
    <title>Tableau Basique</title>
</head>
<body>
    <table border="1">
        <tr>
            <th>Nom</th>
            <th>Âge</th>
            <th>Ville</th>
        </tr>
        <tr>
            <td>Marie</td>
            <td>25</td>
            <td>Paris</td>
        </tr>
        <tr>
            <td>Pierre</td>
            <td>22</td>
            <td>Lyon</td>
        </tr>
        <tr>
            <td>Julie</td>
            <td>27</td>
            <td>Marseille</td>
        </tr>
    </table>
</body>
</html>
```

Cellules de tableau et fusion des cellules

Apprenez à créer des tableaux en HTML en utilisant des cellules et à fusionner des cellules pour organiser votre contenu.

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemple de Tableau</title>
</head>
<body>
    <table border="1">
        <tr>
            <td>Cellule 1-1</td>
            <td>Cellule 1-2</td>
        </tr>
        <tr>
            <td colspan="2">Cellule fusionnée 2-1 et 2-2</td>
        </tr>
    </table>
</body>
</html>
```

Gestion de la taille du tableau

Découvrez comment définir la largeur et la hauteur d'un tableau dans HTML.

```
<!DOCTYPE html>
<html>
<head>
    <title>Tableau avec taille définie</title>
</head>
<body>
    <table width="300" height="200" border="1">
        <tr>
            <td>Cellule 1-1</td>
            <td>Cellule 1-2</td>
        </tr>
        <tr>
            <td>Cellule 2-1</td>
            <td>Cellule 2-2</td>
        </tr>
    </table>
</body>
</html>
```

En-tête et légende

Utilisez les balises d'en-tête et de légende pour structurer votre tableau et le rendre plus compréhensible.

```
<!DOCTYPE html>
<html>
<head>
    <title>Tableau avec en-tête et légende</title>
</head>
<body>
    <table border="1">
        <caption>Informations sur les étudiants</caption>
        <thead>
            <tr>
                <th>Nom</th>
                <th>Âge</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Marie</td>
                <td>25</td>
            </tr>
            <tr>
                <td>Pierre</td>
                <td>22</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

Les bordures

Personnalisez les bordures de votre tableau pour le rendre plus esthétique.

```
<!DOCTYPE html>
<html>
<head>
    <title>Tableau avec bordures personnalisées</title>
    <style>
        table {
            border-collapse: collapse;
        }
        th, td {
            border: 2px solid black;
            padding: 8px;
        }
    </style>
</head>
<body>
    <table>
        <tr>
            <th>Nom</th>
            <th>Âge</th>
        </tr>
        <tr>
            <td>Marie</td>
            <td>25</td>
        </tr>
        <tr>
            <td>Pierre</td>
            <td>22</td>
        </tr>
    </table>
</body>
</html>
```

Les groupes de colonnes et de lignes

Utilisez les balises `<colgroup>` pour regrouper les colonnes et appliquer des styles à des groupes spécifiques de colonnes dans un tableau en HTML.

```
<!DOCTYPE html>
<html>
<head>
    <title>Superheros and Sidekicks</title>
    <style>
        .batman {
            background-color: #f1c40f;
        }
        .flash {
            background-color: #3498db;
        }
    </style>
</head>
<body>
    <table>
        <caption>Superheros and sidekicks</caption>
        <colgroup>
            <col>
            <col span="2" class="batman">
            <col span="2" class="flash">
        </colgroup>
        <tr>
            <td> </td>
            <th>Batman</th>
            <th>Robin</th>
            <th>The Flash</th>
            <th>Kid Flash</th>
        </tr>
        <tr>
            <th>Skill</th>
            <td>Smarts</td>
            <td>Dex, acrobat</td>
            <td>Super speed</td>
            <td>Super speed</td>
        </tr>
    </table>
</body>
</html>
```

La balise de liens

Découvrez comment créer des liens hypertextes en utilisant la balise `<a>` en HTML.

```
<!DOCTYPE html>
<html>
<head>
    <title>Page d'exemple avec lien</title>
</head>
<body>
    <p>Visitez <a href="https://www.example.com">Example.com</a> pour plus
</body>
</html>
```

Les différents types de liens

Apprenez à créer différents types de liens : vers une autre page, dans une page, vers un site Web, de téléchargement, etc.

```
<!DOCTYPE html>
<html>
<head>
    <title>Différents types de liens</title>
</head>
<body>
    <p><a href="autre-page.html">Lien vers une autre page</a></p>
    <p><a href="#section">Lien ancre dans la même page</a></p>
    <p><a href="https://www.example.com">Lien vers un site Web externe</a></p>
    <p><a href="document.pdf" download>Télécharger un fichier PDF</a></p>
</body>
</html>
```

Les Target

Utilisez l'attribut target pour spécifier où ouvrir le lien hypertexte : dans une nouvelle fenêtre, dans la même fenêtre, etc.

```
<!DOCTYPE html>
<html>
<head>
    <title>Utilisation de l'attribut target</title>
</head>
<body>
    <p><a href="https://www.example.com" target="_blank">Ouvrir dans une nouvelle fenêtre</a></p>
    <p><a href="autre-page.html" target="_self">Ouvrir dans la même fenêtre</a></p>
</body>
</html>
```

L'attribut titre

Ajoutez l'attribut title pour afficher une infobulle lorsque l'utilisateur survole le lien avec la souris.

```
<!DOCTYPE html>
<html>
<head>
    <title>Utilisation de l'attribut title</title>
</head>
<body>
    <p><a href="https://www.example.com" title="Visitez Example.com">Lien
</body>
</html>
```

La couleur des liens

Personnalisez la couleur des liens en utilisant des styles CSS.

```
<!DOCTYPE html>
<html>
<head>
    <title>Couleur des liens</title>
    <style>
        a {
            color: blue;
        }
    </style>
</head>
<body>
    <p><a href="https://www.example.com">Lien en bleu</a></p>
</body>
</html>
```

Liens et feuilles de style

Comprenez comment les liens hypertextes peuvent interagir avec les feuilles de style CSS.

```
<!DOCTYPE html>
<html>
<head>
    <title>Liens et feuilles de style</title>
    <style>
        /* Styles pour les liens */
        a {
            color: green;
            text-decoration: underline;
        }

        /* Styles pour les liens au survol de la souris */
        a:hover {
            color: red;
        }
    </style>
</head>
<body>
    <p><a href="https://www.example.com">Lien avec styles CSS</a></p>
</body>
</html>
```

Insérer une image

Apprenez à insérer une image dans une page HTML à l'aide de la balise .

```
<!DOCTYPE html>
<html>
<head>
    <title>Insertion simple d'une image</title>
</head>
<body>
    
</body>
</html>
```

L'espace autour d'une image

Utilisez les propriétés CSS pour contrôler l'espace autour d'une image.

```
<!DOCTYPE html>
<html>
<head>
    <title>Espace autour d'une image</title>
    <style>
        img {
            margin: 10px;
            padding: 5px;
            border: 1px solid black;
        }
    </style>
</head>
<body>
    
</body>
</html>
```

L'alignement d'une image

Alignez une image horizontalement ou verticalement dans son conteneur.

```
<!DOCTYPE html>
<html>
<head>
    <title>Alignement d'une image</title>
    <style>
        img {
            display: block;
            margin: 0 auto;
        }
    </style>
</head>
<body>
    
</body>
</html>
```

L'insertion d'un lien sur une image

Transformez une image en un lien cliquable vers une autre page ou un site Web.

```
<!DOCTYPE html>
<html>
<head>
    <title>Image avec lien</title>
</head>
<body>
    <a href="https://www.example.com">
        
    </a>
</body>
</html>
```

La déclaration de formulaire

Apprenez à créer un formulaire en utilisant la balise <form> en HTML.

```
<!DOCTYPE html>
<html>
<head>
    <title>Déclaration de formulaire</title>
</head>
<body>
    <form action="traitement.php" method="post">
        <!-- Ajoutez ici les éléments du formulaire -->
    </form>
</body>
</html>
```

Zone de texte à une ligne

Utilisez la balise `<input>` avec l'attribut `type="text"` pour créer une zone de texte à une ligne dans un formulaire.

```
<!DOCTYPE html>
<html>
<head>
    <title>Zone de texte à une ligne</title>
</head>
<body>
    <form>
        <label for="nom">Nom :</label>
        <input type="text" id="nom" name="nom">
    </form>
</body>
</html>
```

Menu déroulant

Utilisez la balise <select> avec les balises <option> pour créer un menu déroulant dans un formulaire.

```
<!DOCTYPE html>
<html>
<head>
    <title>Menu déroulant</title>
</head>
<body>
    <form>
        <label for="pays">Pays :</label>
        <select id="pays" name="pays">
            <option value="france">France</option>
            <option value="espagne">Espagne</option>
            <option value="italie">Italie</option>
        </select>
    </form>
</body>
</html>
```

Boutons : radio, checkbox, d'envoi

Utilisez les balises `<input>` avec différents types d'attributs pour créer des boutons dans un formulaire.

```
<!DOCTYPE html>
<html>
<head>
  <title>Boutons dans un formulaire</title>
</head>
<body>
  <form>
    <input type="radio" id="homme" name="sexe" value="homme">
    <label for="homme">Homme</label>
    <input type="radio" id="femme" name="sexe" value="femme">
    <label for="femme">Femme</label><br>

    <input type="checkbox" id="newsletter" name="newsletter" value="oui">
    <label for="newsletter">S'abonner à la newsletter</label><br>

    <input type="submit" value="Envoyer">
    <input type="reset" value="Annuler">
    <input type="button" value="Cliquez ici">
  </form>
</body>
```

Les formulaires : cachés, de transfert de fichier, de mot de passe

Utilisez les balises `<input>` avec différents types d'attributs pour créer des éléments spéciaux dans un formulaire.

```
<!DOCTYPE html>
<html>
<head>
    <title>Formulaires spéciaux</title>
</head>
<body>
    <form>
        <input type="hidden" name="id_utilisateur" value="123">

        <input type="file" id="fichier" name="fichier">

        <label for="motdepasse">Mot de passe :</label>
        <input type="password" id="motdepasse" name="motdepasse">
    </form>
</body>
</html>
```

5. CSS3



A) Cascading Style Sheets

- 👉 « Feuilles de style en cascade »
- 👉 Présentation des document HTML et XML
- 👉 Introduit en 1990, réellement pris en charge en 2000
- 👉 HTML ne décrit que la structure logique (le fond) des documents,
- 👉 la structure physique (la forme) est spécifiée par une feuille de style en CSS.

B) Déclarer une feuille de style

👉 Ajout dans l'élément **<head>** d'une balise **<link>**

```
<head>
  <link rel="stylesheet" type="text/css"
        href="mystyle.css" />
</head>
```

👉 On peut avoir plusieurs feuilles de style (leurs effets se cumulent).

C) Règles

- 👉 En CSS, la mise en forme est spécifiée par un ensemble de règles.
- 👉 Une règle typique est composée de trois parties :
 - un sélecteur
 - une propriété
 - une valeur

```
em { font-style: italic }
```

- 👉 Plusieurs règles similaires peuvent coexister :

```
em { font-style: italic }
em { color: blue }
cite { font-style: italic }
cite { color: blue }
```

C) Règles - exemples

```
em { font-style: italic }  
em { color: blue }  
cite { font-style: italic }  
cite { color: blue }
```

```
em { font-style: italic ; color: blue }  
cite { font-style: italic ; color: blue }
```

```
em, cite { font-style: italic ; color: blue }
```

D) Propriétés du texte

font-style	normal, italic, oblique
font-weight	normal bold bolder lighter ou une valeur entre 100 et 900 (400 = normal)
font-variant	normal small-caps

👉 font-family

👉 font-size

D) Propriétés du texte - exemples

```
body { font-size: 12px; }
```

```
h1 { font-size: 150%; }
```

E) Les couleurs



```
em { color: #f00 }                      /* #rgb */
em { color: #ff0000 }                    /* #rrggbb */
em { color: rgb(255,0,0) }
em { color: rgb(100%, 0%, 0%) }
em { color: rgba(255, 0, 0, 0.5) }
```

F) Les sélecteurs complexes

```
X Y { /* s'applique à tout élément Y situé  
à l'intérieur d'un X – même indirectement */ }  
  
X > Y { /* s'applique à tout élément Y situé  
directement à l'intérieur d'un X */ }  
  
X + Y { /* s'applique à tout élément Y situé  
immédiatement après un X */ }
```

F) Les sélecteurs complexes - exemples

```
q      { font-style: italic; }
q em   { font-weight: bold; }
q strong { text-decoration: underline; }

body>h1 { text-align: center; }

h1+* { font-variant: small-caps; }

ul ul li { font-size: 80%; }
```

G) Classes et identifiant

👉 HTML autorise les attributs suivant dans n'importe quelle balise :

- id accepte comme valeur un nom unique (interdiction d'utiliser le même id dans un même document) ;
- class accepte comme valeur une liste de noms séparés par des espaces (le même nom de classe peut être présent dans plusieurs balises).

```
<ol id="contents">...</ol>  
  
<article class="post funny">...</article>
```

H) Sélecteurs associés

```
article.post { /* tout <article> de la classe 'post' */ }

.funny          { /* tout élément de la classe 'funny' */ }

ol#contents   { /* toute <ol> avec l'id 'contents' */ }

#contents      { /* tout élément avec l'id 'contents' */ }
```

I) Priorité

- 👉 La règle la plus spécifique a toujours la priorité.
- 👉 En cas de spécité égale, c'est la dernière règle (dans l'ordre du des fichier.s) qui s'applique.
- 👉 Chaque attribut CSS est traité séparément.

S'entraîner avec les sélecteurs CSS

👉 Pour s'entraîner à l'utilisation des selecteurs css, il existe le jeu CSS Diner, <https://flukeout.github.io/>

Introduction aux Media Queries

Découvrez les Media Queries en CSS, qui permettent d'appliquer des styles spécifiques en fonction de différentes caractéristiques de l'appareil ou de l'écran.

```
<!DOCTYPE html>
<html>
<head>
    <title>Introduction aux Media Queries</title>
    <style>
        body {
            font-size: 16px;
        }

        @media screen and (max-width: 600px) {
            body {
                font-size: 14px;
            }
        }
    </style>
</head>
<body>
    <p>Ceci est un exemple de Media Query.</p>
</body>
</html>
```

Anatomie d'une Media Query

@media screen (min-width: 320px) and (max-width: 768px)

AT-RULE	MEDIA TYPE	MEDIA FEATURE	OPERATOR	MEDIA FEATURE
@media	screen	(min-width: 320px)	and	(max-width: 768px)

Cibler différentes caractéristiques

Utilisez les Media Queries pour cibler des caractéristiques telles que la largeur d'écran, l'orientation, le type de périphérique, etc.

```
<!DOCTYPE html>
<html>
<head>
    <title>Cibler différentes caractéristiques</title>
    <style>
        body {
            background-color: white;
        }

        @media screen and (max-width: 600px) {
            body {
                background-color: lightgray;
            }
        }

        @media (orientation: portrait) {
            body {
                font-family: Arial, sans-serif;
            }
        }

        @media (orientation: landscape) {
            body {
                font-family: "Times New Roman", serif;
            }
        }
    </style>
</head>
<body>
    <p>Essayez de redimensionner la fenêtre ou de changer l'orientation</p>
</body>
</html>
```

Introduction aux Flexbox

Découvrez le modèle de mise en page flexible avec Flexbox, qui facilite le positionnement et l'alignement des éléments dans un conteneur.

```
<!DOCTYPE html>
<html>
<head>
  <title>Introduction aux Flexbox</title>
  <style>
    .container {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 200px;
    }

    .item {
      width: 50px;
      height: 50px;
      background-color: red;
      margin: 5px;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="item"></div>
    <div class="item"></div>
    <div class="item"></div>
  </div>
</body>
</html>
```

L'axe principal et l'axe secondaire

Comprenez les concepts d'axe principal (main axis) et d'axe secondaire (cross axis) dans Flexbox, qui déterminent le flux des éléments.

```
<!DOCTYPE html>
<html>
<head>
    <title>Axe principal et axe secondaire</title>
    <style>
        .container {
            display: flex;
            flex-direction: row; /* Par défaut, l'axe principal est horizontal */
            height: 200px;
            justify-content: center; /* Alignement horizontal des éléments */
            align-items: center; /* Alignement vertical des éléments */
        }

        .item {
            width: 50px;
            height: 50px;
            background-color: red;
            margin: 5px;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="item"></div>
        <div class="item"></div>
        <div class="item"></div>
    </div>
</body>
</html>
```

Aller plus loin

Tous les concepts de la flexbox sont résumés dans un excellent article de CSSTricks : <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Introduction aux Grids

Découvrez le modèle de mise en page en grille avec CSS Grid, qui permet de créer des mises en page complexes et réactives.

```
<!DOCTYPE html>
<html>
<head>
  <title>Introduction aux Grids</title>
  <style>
    .container {
      display: grid;
      grid-template-columns: repeat(3, 100px); /* Trois colonnes de
      grid-gap: 10px; /* Espacement entre les éléments */
    }

    .item {
      width: 100px;
      height: 100px;
      background-color: red;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="item"></div>
    <div class="item"></div>
    <div class="item"></div>
  </div>
</body>
</html>
```

Mise en page réactive avec Grids

Utilisez les Media Queries avec CSS Grid pour créer des mises en page réactives qui s'adaptent à différentes tailles d'écran.

```
<!DOCTYPE html>
<html>
<head>
    <title>Mise en page réactive avec Grids</title>
    <style>
        .container {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
            grid-gap: 10px;
        }

        .item {
            height: 100px;
            background-color: red;
        }

        @media screen and (max-width: 600px) {
            .container {
                grid-template-columns: repeat(2, 1fr);
            }
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="item"></div>
        <div class="item"></div>
        <div class="item"></div>
        <div class="item"></div>
    </div>
</body>
</html>
```

Contrôle de l'alignement et de l'espacement

Utilisez les propriétés de CSS Grid pour contrôler l'alignement et l'espacement des éléments dans la grille.

```
<!DOCTYPE html>
<html>
<head>
    <title>Contrôle de l'alignement et de l'espacement</title>
    <style>
        .container {
            display: grid;
            grid-template-columns: repeat(3, 100px);
            grid-gap: 10px;
            justify-content: center; /* Alignement horizontal */
            align-items: center; /* Alignement vertical */
            grid-auto-rows: 100px; /* Hauteur par défaut des lignes */
        }

        .item {
            background-color: red;
        }

        .tall {
            grid-row: span 2; /* Étendre l'élément sur deux lignes */
        }

        .wide {
            grid-column: span 2; /* Étendre l'élément sur deux colonnes */
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="item"></div>
        <div class="item tall"></div>
        <div class="item wide"></div>
        <div class="item"></div>
    </div>
</body>
</html>
```

Aller plus loin

Tous les concepts de la grid sont résumés dans un excellent article de CSSTricks : <https://css-tricks.com/snippets/css/complete-guide-grid/>

6. JAVASCRIPT



JS



1. Les fonctions

En JavaScript, les fonctions sont des objets de première classe.

Cela signifie qu'elles peuvent être manipulées et échangées, qu'elles peuvent avoir des propriétés et des méthodes, comme tous les autres objets JavaScript. Les fonctions sont des objets Function.

Valeurs par défaut des arguments

```
1 function multiply(a, b=1){  
2     return a * b  
3 }  
4  
5 console.log(multiply(5,2)) // 10  
6  
7 console.log(multiply(5)) //5
```

Utilisation des fonctions fléchées

```
1 const multiply = (a, b = 1) => a * b;  
2  
3 console.log(multiply(5, 2)); // 10  
4  
5 console.log(multiply(5)); //5  
6
```

Fonctions imbriquées

```
1  function ajoutCarre(a, b) {  
2      function carre(x) {  
3          return x * x;  
4      }  
5  
6      return carre(a) + carre(b);  
7  }  
8
```

Fonctions anonymes

```
1 setTimeout( () => {  
2   console.log("timeout over");  
3 }, 1000);  
  
4  
5 setTimeout( function () {  
6   console.log("timeout over");  
7 }, 1000);  
8
```

Fonction de Callback

Une fonction de callback est une fonction passée dans une autre fonction en tant qu'argument. Elle est ensuite invoquée à l'intérieur de la fonction externe pour accomplir une action.

```
1 const greetings = (name) => {
2     alert("Greetings " + name)
3 }
4
5 const processUserName = (callback) => {
6     const name = prompt("What's your name ?")
7     callback(name)
8 }
9
10 processUserName(greetings)
```

Closures

Une closure est une fonction interne qui va « se souvenir » et pouvoir continuer à accéder à des variables définies dans sa fonction parente même après la fin de l'exécution de celle-ci.

```
Z: > Formation > JS formation.js > ...
1 ~ function compteur() {
2     let count = 0;
3
4 ~     return function() {
5         |     |     return count++;
6     };
7 }
8
9 let plusUn = compteur();
10
11 |
```

Le scope (la portée)

Exemple de variable locale :

```
Z: > Formation > JS formation.js > 📄 a
1  function a() {
2      var str = "Bonjour";
3      console.log(str); // "Bonjour"
4      function b() {
5          var str = "Le monde";
6          console.log("In b : ", str); // 'In b : Le monde'
7      }
8      b();
9  }
```

Le scope (la portée)

Exemple de variable globale:

```
Z: > Formation > JS formation.js > ...
1   var str = "Hello world";
2
3   function a() {
4       console.log(str); // "Hello world"
5   }
6
7   |
```

Le scope (la portée)

Version ES6 : Let et Const

```
Z: > Formation > JS formation.js > b
1  // Avant
2  function a() {
3      var str = 'Hello';
4      if (true) {
5          var str = 'World'; // C'est la même variable !
6          console.log(str); // World
7      }
8      console.log(str); // World
9  }
10
11 function b() {
12     let str = 'Hello';
13     if (true) {
14         let str = 'World'; // c'est une variable différente
15         console.log(str); // World
16     }
17     console.log(str); // Hello
18 }
```



2. Les objets

JavaScript est conçu autour d'un paradigme simple, basé sur les objets. Un objet est un ensemble de propriétés et une propriété est une association entre un nom (aussi appelé clé) et une valeur. La valeur d'une propriété peut être une fonction, auquel cas la propriété peut être appelée « méthode ».

Création d'un objet

```
Z: > Formation > JS formation.js > ...
1  var o = new Object();
```

Un objet vide est stocké dans "o". Ainsi, nous pouvons accéder aux méthodes du constructeur Objet.

Par exemple :

- > Assign
- > Create
- > entries
- > keys

```
Z: > Formation > JS formation.js > ...
1  var o = {};
```



3. Les classes

Les classes JavaScript ont été introduites avec ECMAScript 2015. Elles fournissent uniquement une syntaxe plus simple pour créer des objets et manipuler l'héritage.

Définir une classe

```
Z: > Formation > JS formation.js > ...
```

```
1 ✓ class Rectangle {  
2   ✓ constructor(hauteur, largeur) {  
3     this.hauteur = hauteur;  
4     this.largeur = largeur;  
5   }  
6 }  
7  
8 |
```

Les expressions de classes

```
Z: > Formation > JS formation.js > ...
1  // anonyme
2  let Rectangle = class {
3      constructor(hauteur, largeur) {
4          this.hauteur = hauteur;
5          this.largeur = largeur;
6      }
7  };
8
9  // nommée
10 let Rectangle = class Rectangle {
11     constructor(hauteur, largeur) {
12         this.hauteur = hauteur;
13         this.largeur = largeur;
14     }
15 };
16
17
```

Ajout de méthodes

```
Z: > Formation > JS formation.js > ...
● 1  class Rectangle {
  2    constructor(hauteur, largeur) {
  3      this.hauteur = hauteur;
  4      this.largeur = largeur;
  5    }
  6
  7    get area() {
  8      return this.calcArea();
  9    }
 10
 11   calcArea() {
 12     return this.largeur * this.hauteur;
 13   }
 14
 15
 16   const carré = new Rectangle(10, 10);
 17
 18   console.log(carré.area);
```

Méthode "static"

```
Z: > Formation > JS formation.js > ...
1  class Point {
2    constructor(x, y) {
3      this.x = x;
4      this.y = y;
5    }
6
7    static distance(a, b) {
8      const dx = a.x - b.x;
9      const dy = a.y - b.y;
10     return Math.hypot(dx, dy);
11   }
12 }
13
14 const p1 = new Point(5, 5);
15 const p2 = new Point(10, 10);
16
17 console.log(Point.distance(p1, p2));
```

Héritage

```
Z: > Formation > JS formation.js > 🐶 Chien
  1 class Animal {
  2   constructor(nom) {
  3     this.nom = nom;
  4   }
  5
  6   parle() {
  7     console.log(`#${this.nom} fait du bruit.`);
  8   }
  9 }
10
11 class Chien extends Animal {
12   constructor(nom) {
13     super(nom); // appelle le constructeur parent avec le paramètre
14   }
15   parle() {
16     console.log(`#${this.nom} aboie.`);
17   }
18 }
```



4. Les collections

Les collections incluent les tableaux et les objets semblables à des tableaux que sont les objets `Array` et les objets `TypedArray`.

Créer un tableau

```
Z: > Formation > JS formation.js > ...
1  var arr = new Array(longueurTableau);
2  var arr = Array(longueurTableau);
3
4  // Cela aura le même effet que :
5  var arr = [];
6  arr.length = longueurTableau;
7
8  |
```

Remplir un tableau

```
Z: > Formation > JS formation.js > ...
1  var arr = [];
2  arr[3.4] = "Oranges";
3  console.log(arr.length);           // 0
4  console.log(arr.hasOwnProperty(3.4)); // true|
```

Quelques méthodes de l'objet Tableau

Le forEach

```
1 const arr = ["Orange", "Fraise", "Pêche"]
2
3 arr.forEach(el => console.log(el))
4 //affiche Orange, puis Fraise, puis Pêche
```

Quelques méthodes de l'objet Tableau

La longueur

```
Z: > Formation > JS formation.js > ...
1  var chats = ['Marie', 'Toulouse', 'Berlioz'];
2  console.log(chats.length); // 3
3
4  chats.length = 2;
5  console.log(chats); // affiche "Marie,Toulouse" - Berlioz a été retiré
6
7  chats.length = 0;
8  console.log(chats); // affiche [], le tableau est vide
9
10 chats.length = 3;
11 console.log(chats); // [ <3 empty slots> ]
```

Quelques méthodes de l'objet Tableau

```
1 let arr = ["Orange", "Fraise", "Pêche"];
2 arr.sort(); //["Fraise","Pêche"]
3
4 let arr = ["Orange", "Fraise", "Pêche"];
5 arr.reverse(); //["Pêche", "Fraise", "Orange"]
6
7 let arr = ["Orange", "Fraise", "Pêche"];
8 const first = arr.shift(); //["Fraise", "Orange","Pêche"]
9 //first = "Orange"
10
11 let arr = ["Orange", "Fraise", "Pêche"];
12 const count = arr.push("Ananas"); //["Orange","Fraise", "Pêche", "Ananas"]
13 //count = 4
14
15 let arr = ["Orange", "Fraise", "Pêche"];
16 const list = arr.join(" - "); // "Orange - Fraise - Pêche"
17 //count = 4
18
19 let arr = ["Orange", "Fraise", "Pêche"];
20 const newArr = arr.map(fruit => fruit.toUpperCase());
21 //newArr = ["ORANGE", "FRAISE", "PÊCHE"]
22
```



5. Les erreurs

La première chose qu'il y a à savoir lorsqu'on souhaite prendre en charge les erreurs est que lorsqu'une erreur d'exécution survient dans un script le JavaScript crée automatiquement un objet à partir de l'objet global Error qui est l'objet de base pour les erreurs en JavaScript.

Attraper des erreurs

```
Z: > Formation > JS formation.js > ...
1 try{
2     prenom
3     alert('Bonjour');
4 }catch(err){
5     alert('Erreur rencontrée. ' +
6         '\nNom de l\'erreur : ' + err.name +
7         '\nMessage d\'erreur : ' + err.message +
8         '\nEmplacement de l\'erreur : ' + err.stack);
9 }
10 alert('Ce message s\'affiche correctement');
11
```



6. Les modules

Les modules permettent de fournir un mécanisme pour diviser les programmes JavaScript en plusieurs modules qu'on pourrait importer les uns dans les autres. Cette fonctionnalité était présente dans Node.js depuis longtemps et plusieurs bibliothèques et frameworks JavaScript ont permis l'utilisation de modules (CommonJS, AMD, RequireJS ou, plus récemment, Webpack et Babel).

Déclarer un module

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Document</title>
7    </head>
8    <body></body>
9    <script src="index.js" type="module" />
10   </html>
11
```

Exportation de module

```
Z: > Formation > JS formation.js >  draw
  1   export const name = 'square';
  2
  3   export function draw(ctx, length, x, y, color) {
  4     ctx.fillStyle = color;
  5     ctx.fillRect(x, y, length, length);
  6
  7   return {
  8     length: length,
  9     x: x,
 10    y: y,
 11    color: color
 12  };
 13}
```

Importation de module

```
Z: > Formation > JS formation.js
1   import { name, draw, reportArea, reportPerimeter } from './modules/square.mjs';
2
3
4 |
```



7. Les évènements

Les événements sont des actions ou des occurrences qui se produisent dans le système que vous programmez et dont le système vous informe afin que vous puissiez y répondre d'une manière ou d'une autre si vous le souhaitez.

Exemple d'évènement

```
Z: > Formation > JS formation.js > onclick
1  var btn = document.querySelector('button');
2
3  ↵ function random(number) {
4      return Math.floor(Math.random()*(number+1));
5  }
6
7  ↵ btn.onclick = function() {
8      var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255) + ')';
9      document.body.style.backgroundColor = rndCol;
10 }
```

Exemple d'évènement - Listener

```
Z: > Formation > JS formation.js > ...
1  var btn = document.querySelector('button');
2
3  ↘ function bgChange() {
4      var rndCol = 'rgb(' + random(255) + ', ' + random(255) + ', ' + random(255) + ')';
5      document.body.style.backgroundColor = rndCol;
6  }
7
8  btn.addEventListener('click', bgChange);|
```



8. Les promesses

Une promesse est un objet (`Promise`) qui représente la complétion ou l'échec d'une opération asynchrone. La plupart du temps, on « consomme » des promesses.

Promesse :

```
1 new Promise((resolve, reject) => {
2   try {
3     //une opération asynchrone
4     resolve("Terminé");
5   } catch (error) {
6     reject(error);
7   }
8 })
9 .then((resolvedValue) => {
10   //...traitement de la valeur
11 })
12 .catch((error) => {
13   //traitement de l'erreur
14 });
15
```



9. Fetch

L'API Fetch fournit une interface JavaScript pour l'accès et la manipulation des parties de la pipeline HTTP, comme les requêtes et les réponses. Cela fournit aussi une méthode globale `fetch()` qui procure un moyen facile et logique de récupérer des ressources à travers le réseau de manière asynchrone.

Exemple :

```
1 const init = {  
2   method : "GET",  
3   headers : new Headers(),  
4   mode : "cors",  
5   cache : "default"  
6 }  
7  
8 fetch("flowers.jpg", init)  
  .then((response) => response.blob())  
  .then((blob) => {  
11     const objectURL = URL.createObjectURL(blob)  
12     myImage.src = objectURL  
13   })
```