HTML

Where the magic begins

SOMMAIRE

LE WEB

C’est où, c’est quand, c’est comment ?

PRÉSENTATION

- Avant toute chose, il est important de comprendre comment fonctionne le web

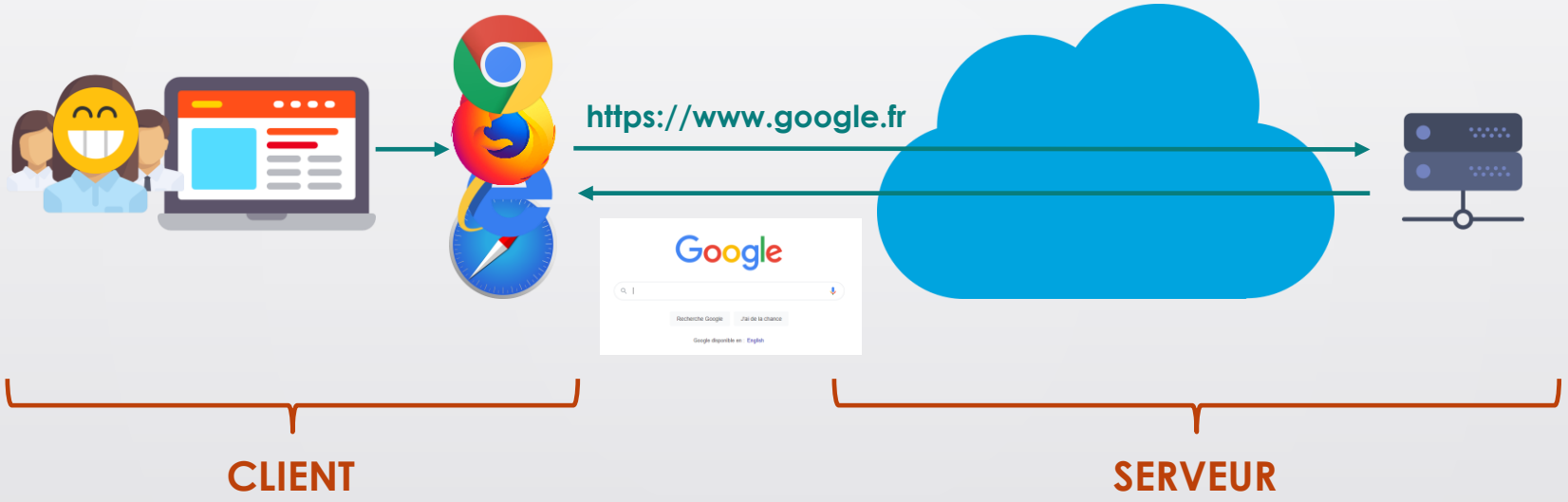
- **WWW** = World Wide Web = la toile d'araignée mondiale (si, si)

- Il s'agit de l'ensemble des sites accessibles par tout un chacun

- Ne pas confondre le web et internet !
  - **Web** : l'ensemble des sites navigables
  - **Internet** : infrastructure de communication (inclut le web, les mails, les chats, etc.)

COMMENT ÇA MARCHE ?

- Comment fonctionne un site web ?



- On appelle cette architecture « **client/serveur** » :
  - On a d'un côté un **client** (*vous*) qui demande un site web
  - On a d'un autre côté un **serveur** (*Google*) qui reçoit la requête et retourne le site web en question

LES TECHNOLOGIES

- Tout site web repose sur 3 technologies (langages) :

HTML
Version 5
Sert à **structurer l'information**

CSS
Version 3
Sert à **styliser l'affichage**

JavaScript
Sert aux **interactions** avec l'utilisateur
Pas au programme de ce module

LES NAVIGATEURS

- Selon votre système d'exploitation (OS), vous avez un navigateur par défaut :
  - Windows → **Edge**
  - MacOS, iOS → **Safari**
  - Unix → **Firefox**
  - Android → **Chrome**
- Et vous avez la possibilité d'en installer d'autres (*Chrome*, *Opera*, *Brave*, etc.)

- Le **navigateur** est le logiciel qui va :
  - **Transmettre** au serveur votre demande (requête)
  - Vous **afficher** le résultat de ladite requête (ou une erreur : 500, 404, etc.)

LES NAVIGATEURS

- Certains navigateurs ont des réputations sulfureuses

- **Internet Explorer**, par exemple, avait la réputation d'être bourré de bugs, d'une lenteur terrible, et il ne respectait pas les conventions du W3C (voir ci-après)

- Aujourd'hui, chaque navigateur a ses qualités et ses défauts

LES NAVIGATEURS

Chrome
58%

Navigateur de Google
Respect des normes ++
Intégré sur les smartphones récents (Android)

Safari
21%

Navigateur d'Apple
A eu des périodes difficiles
Natif sur MacOS et iOS

Firefox
8%

Navigateur open-source (géré par une association)
A failli être leader, avant d'être détrôné par Chrome
Aujourd'hui un peu lourd et boudé

Edge
5%

Descendant d'Internet Explorer
Né avec le même moteur que Chrome pour pallier les défauts d'IE et nettoyer sa réputation
Par défaut sur Windows et un peu insistant

On fait en général en sorte que son site web fonctionne correctement sur ces 4 navigateurs

HTML : PRÉSENTATION

Origines et balises de base

C'EST QUOI, HTML ?

- HyperText Markup Language
- Aujourd'hui en version 5 (HTML5)
- C'est un langage de balisage,
  c'est-à-dire qu'on ne lui donne pas d'instructions
- A la place, on structure sémantiquement la page

C'EST QUOI, HTML ?

- **HTML** est une déclinaison du **XML**, un langage de balisage qui permet de **structurer** des informations à l'aide de **balises**

- On peut ainsi **décrire** des informations, comme dans une base de données

```
<xml>
    <videotheque>
        <film vu="non">
            <titre>Mourir peut attendre</titre>
            <realisateur>Cary Joji Fukunaga</realisateur>
            <acteur>Daniel Craig</acteur>
            <annee>2021</annee>
        </film>
        <film vu="non">
            <titre>Dune</titre>
            <realisateur>Damien Chazelle</realisateur>
            <acteur>Timothée Chalamet</acteur>
            <annee>2021</annee>
        </film>
        <film vu="oui" date-sortie="03/06/2021">
            <titre>Kaamelott, premier volet</titre>
            <realisateur>Alexandre Astier</realisateur>
            <acteur>Alexandre Astier</acteur>
            <annee>2021</annee>
        </film>
    </videotheque>
</xml>
```

C'EST QUOI, HTML ?

Ma vidéothèque est composée de **films**, eux-mêmes composés de certaines informations

Les balises sont toujours inscrites entre **chevrons**, et sont ouvertes (<film>) et fermées (</film>)

Des balises peuvent **contenir d'autres balises**, mais aussi des **attributs**

En XML, vous créez toutes les balises que vous voulez ;
En HTML, on vous met à disposition des **balises spécifiques**

```
<xml>
    <videotheque>
        <film vu="non">
            <titre>Mourir peut attendre</titre>
            <realisateur>Cary Joji Fukunaga</realisateur>
            <acteur>Daniel Craig</acteur>
            <annee>2021</annee>
        </film>
        <film vu="non">
            <titre>Dune</titre>
            <realisateur>Damien Chazelle</realisateur>
            <acteur>Timothée Chalamet</acteur>
            <annee>2021</annee>
        </film>
        <film vu="oui" date-sortie="03/06/2021">
            <titre>Kaamelott, premier volet</titre>
            <realisateur>Alexandre Astier</realisateur>
            <acteur>Alexandre Astier</acteur>
            <annee>2021</annee>
        </film>
    </videotheque>
</xml>
```

C'EST QUOI, HTML ?

- HTML vous met à disposition une **centaine de balises** destinées à **structurer le contenu** de votre page

- Concrètement, vous allez « ranger » l'ensemble du contenu de votre page dans des balises qui **décrivent sémantiquement** ce que c'est (corps de la page, titre, lien, liste, etc.)

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML
2  <html>
3      <head>
4          <title>Example</title>
5          <link href="screen.css" rel="sty
6      </head>
7      <body>
8          <h1>
9              <a href="/">Header</a>
10         </h1>
11         <ul id="nav">
12             <li>
13                 <a href="one/">One</a>
14             </li>
15             <li>
16                 <a href="two/">Two</a>
17             </li>
```

STRUCTURE D'UN FICHIER HTML

- Tandis qu'un fichier XML peut avoir quasiment la structure que vous voulez, un fichier HTML doit respecter certaines règles :

```
<!DOCTYPE html>
<html>
    <head>
        ... informations sur le site (titre, données SEO, etc.)
        ... chargement des styles (CSS)
        ... chargement des scripts (JavaScript)
    </head>
    <body>
        ... contenu de votre page (ce qui sera visible à l'écran)
    </body>
</html>
```

- Il existe d'autres règles spécifiques à chaque balise… on a ici au moins la structure obligatoire du fichier

STRUCTURE D'UN FICHIER HTML

- On parle ici d'une arborescence avec des nœuds

- Chaque élément a :
  - Des enfants
  - Un parent (sauf `html`)

- L'arbre d'un document HTML est appelé le **DOM**
  → *Document Object Model*

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML
2  <html>
3      <head>
4          <title>Example</title>
5          <link href="screen.css" rel="sty
6      </head>
7      <body>
8          <h1>
9              <a href="/">Header</a>
10         </h1>
11         <ul id="nav">
12             <li>
13                 <a href="one/">One</a>
14             </li>
15             <li>
16                 <a href="two/">Two</a>
17             </li>
```

HTML : BALISES DE BASE

Structurons ensemble nos pages web

STRUCTURE D'UN FICHIER HTML

```
<!DOCTYPE html>
<html>
    <head>
        ... informations sur le site
        ... chargement des styles (CS
        ... chargement des scripts (
    </head>
    <body>
        ... contenu de votre page (ce
    </body>
</html>
```

Cette ligne est une balise un peu particulière qui précise au navigateur qu'il s'agit d'un fichier HTML
Elle est obligatoire et n'a pas de fermeture

La balise `html` est la balise qui contient tout le reste

La balise `head` n'est pas affichée
Elle sert à « configurer » la page

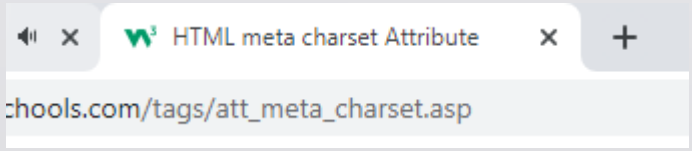
C'est la balise `body` qui est affichée à l'écran

Pour être valide, un fichier HTML doit toujours avoir cette structure

LES BASES DU <HEAD>

- La balise head contient en général au minimum deux balises :

```
<!DOCTYPE html>
<html>
    <head>
        <title>Le titre de ma page (qui apparaît sur l'onglet du navigateur)</title>
        <meta charset="UTF-8"/>
    </head>
    <body>
        ... contenu de votre page (ce qui sera visible à l'écran)
    </body>
</html>
```



meta est une balise auto-fermante,
c'est-à-dire qu'au lieu d'écrire <meta …></meta>,
on peut écrire <meta …/>

C'EST À VOUS !

- Nous savons désormais comment créer une page web !
- Nous allons donc créer notre premier fichier HTML
  et l'afficher dans notre navigateur

- Téléchargez Visual Studio Code : https://code.visualstudio.com/
  *VSCode (pour les intimes) est un **IDE** (Integrated Development Environment), c'est-à-dire un logiciel pour faire du développement*

- **Créez un répertoire** pour votre nouveau projet,
  puis **créez un fichier** `index.html` valide avec du texte dans le body,
  pour ensuite l'**ouvrir avec votre navigateur**

LES BALISES LES PLUS COURANTES

- Félicitations, vous avez créé votre première page web !

- Maintenant, on va vouloir enrichir un peu son contenu, notamment à l'aide de deux balises (enfin six…) :

`p`

Les paragraphes

Des blocs de « texte riche »

`h1, h2, h3, h4, h5, h6`

Les titres

Le numéro renseigne sur l'arborescence

LES PARAGRAPHES

p

Les paragraphes

Des blocs de « texte riche »

- Les paragraphes contiennent du texte et éventuellement des images
- Ils sont un bloc indivisible dont les éléments voisins viennent se positionner au-dessus ou en-dessous
- On peut sauter une ligne dans un paragraphe grâce à la balise `<br/>`
- **A vous de jouer :** ajoutez des paragraphes à votre document

LES TITRES

- Vous pouvez utiliser 6 niveaux de titre allant de h1 à h6

h1, h2, h3, h4, h5, h6
Les titres
Le numéro renseigne sur l'arborescence

- Ces titres sont particulièrement importants pour la structure de votre page

```
<h1>Bienvenue sur mon site</h1>
<h2>Le développement</h2>
<p>Des explications sur le développement :)</p>
<h3>Le HTML</h3>
<p>Le HTML, c'est un langage trop cool</p>
<h4>Les balises</h4>
...
<h4>Les attributs</h4>
...
<h3>Le CSS</h3>
<p>Ca aussi, c'est plutôt cool.</p>
<h2>La recette du cheesecake</h2>
...
```

On a alors la structure suivante :

1. *Bienvenue sur mon site*
    1. *Le développement*
        1. *Le HTML*
            1. *Les balises*
            2. *Les attributs*
        2. *Le CSS*
    2. *La recette du cheesecake*

PARENTHÈSE : TITRES ET RÉFÉRENCEMENT

- Pour le **référencement** de votre site,
  Google (comme ses concurrents) analyse le contenu
  de votre site, c'est-à-dire le HTML

`h1, h2, h3, h4, h5, h6`
Les titres
Le numéro renseigne sur l'arborescence

- Google ne voit pas le CSS, ce n'est pas un humain capable de distinguer,
  à l'œil, quelles parties sont importantes, quelle est la **structure du contenu**

- Il est donc important d'avoir un document sémantiquement cohérent
  (c'est-à-dire avec des niveaux de titre correctement agencés)
  de façon à optimiser la compréhension des robots **(SEO)**

PARENTHÈSE : VALIDATION W3C

- Comme nous le disions plus tôt, le HTML, c'est du XML avec des contraintes
- Le **W3C** (World Wide Web Consortium) est un organisme de standardisation qui décide donc de ce qui est standard et ce qui ne l'est pas
- Il met à disposition différents **validateurs** afin de voir si son code respecte les normes établies
- Le validateur HTML est accessible au bout de lien suivant : https://validator.w3.org/
- Testez donc votre code !

LES IMAGES

- Dans un paragraphe, ou en-dehors, on peut intégrer des images

- On utilise alors la balise **img** avec l'attribut **src**

```
<img src="./images/pikachu.png" />
```

*Chemin **relatif***

```
<img src="http://www.google.fr/une-image.jpg" />
```

*Chemin **absolu***

- Ajoutez des images à votre page en allant chercher
  tantôt des images sur le web, tantôt des images sur votre machine

LES IMAGES

- Sur une `img`, on peut préciser les attributs `width` et `height`
- Ceux-ci permettent de spécifier une taille en pixels
- Si on n'en renseigne qu'un des deux, l'autre est calculé automatiquement de façon à conserver les proportions de l'image
- Si on renseigne les deux, l'image peut être déformée
- A vous de jouer

LES IMAGES

- Enfin, une `img` peut avoir l'attribut `alt` qui va contenir le **texte alternatif** de l'image (description du contenu de l'image)
- Ce texte sera affiché si l'image ne charge pas…
- … mais pas seulement : il est également lu par les **moteurs de recherche** ainsi que les **outils d'accessibilité**
- C'est donc une bonne pratique que de le renseigner ; à vous de jouer

LES LIENS

- Le principe du web, et du HTML en particulier, c'est qu'on peut naviguer entre les contenus via des liens hypertextes

- Pour créer un lien, au sein d'un paragraphe ou en-dehors, on utilise la balise **a** avec l'attribut **href**

```
<a href="http://www.google.fr">Vers Google</a>
```

L'adresse vers laquelle on va mener l'utilisateur

Le texte qui sera affiché

PARENTHÈSE : LES DIFFÉRENTS TYPES D'ADRESSE

- Il existe 3 types d'adresse que l'ont peut renseigner dans un lien (ou le src d'une image…) :
  - Les adresses absolues : elles pointent vers un autre site
    - **http://www.google.fr**
  - Les adresses relatives : elles pointent vers un document sur le même serveur, relativement à l'emplacement de la page actuelle
    - **./mon-sous-dossier/mon-autre-page.html**
    - **../images/truc.png**
  - Les adresses relatives à la racine : elles pointent vers un document sur le même serveur, mais en partant de la racine
    - **/pages/page.html**
    - **/projet/images/mon-image.png**

LES LIENS

- On peut, au clic sur un lien, **forcer l'ouverture d'un nouvel onglet** avec l'attribut target :

```
<a href="http://www.google.fr" target="_blank">Vers Google dans un nouvel onglet</a>
```

- Cependant, ce n'est pas une bonne pratique : le W3C dit que l'utilisateur doit pouvoir **naviguer comme il veut**, qu'on doit lui laisser le choix (clic molette s'il le souhaite)

- A vous de jouer : ajoutez des liens à votre contenu et créez plusieurs pages liées entre elles

LES LISTES

- Toujours en vue d'organiser correctement le contenu, on peut faire des listes :

```
<ul>
    <li>ABC</li>
    <li>DEF</li>
    <li>GHI</li>
</ul>
```

- ABC
- DEF
- GHI

```
<ol>
    <li>Un deux trois</li>
    <li>Quatre cinq six</li>
    <li>Sept huit neuf</li>
</ol>
```

1. Un deux trois
2. Quatre cinq six
3. Sept huit neuf

ul = Unordered List

ol = Ordered List

li = List Item

Les listes ne peuvent pas être insérées dans un paragraphe, ça n'aurait pas de sens

A vous de jouer

LES TABLEAUX

- On peut avoir besoin de créer des tableaux de données à deux dimensions

- **Attention :** les tableaux ne sont pas faits pour faire de la mise en forme !

- Pour créer un tableau, on commence par la balise table

```
<table>

</table>
```

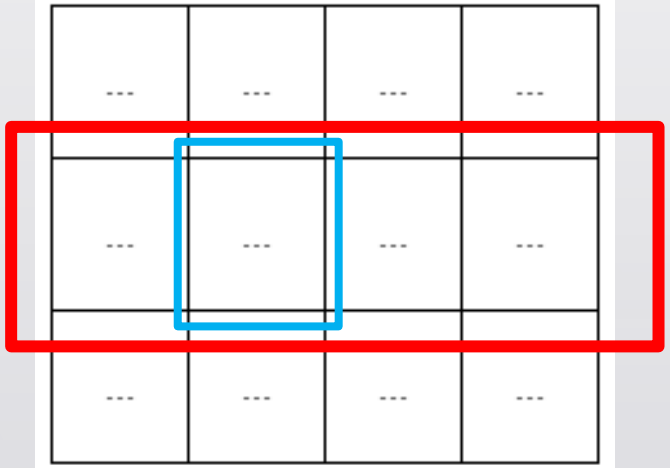
LES TABLEAUX

```
<table>

</table>
```

- Un tableau ne peut pas contenir directement du texte

- Il doit contenir a minima des lignes, qui elles-mêmes vont contenir des cellules



Ce tableau contient 3 lignes,
qui contiennent chacune 4 cellules

LES TABLEAUX

```
<table>
   <tr>
     <td>...</td>
     <td>...</td>
     <td>...</td>
     <td>...</td>
  </tr>
   <tr>
     <td>...</td>
     <td>...</td>
     <td>...</td>
     <td>...</td>
  </tr>
   <tr>
     <td>...</td>
     <td>...</td>
     <td>...</td>
     <td>...</td>
  </tr>
</table>
```

1ère ligne

2ème ligne

3ème ligne

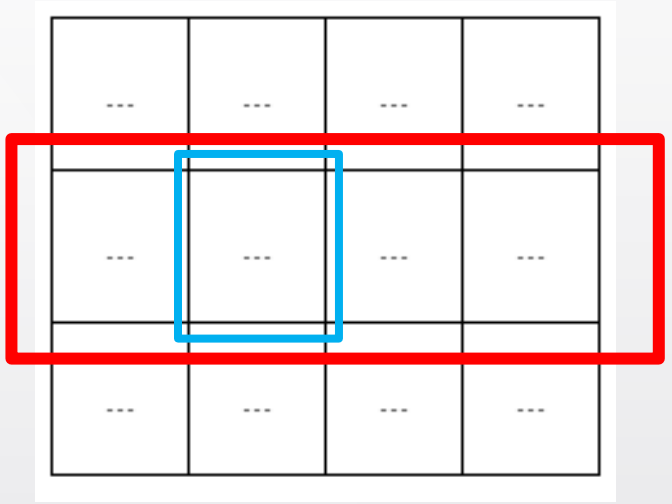


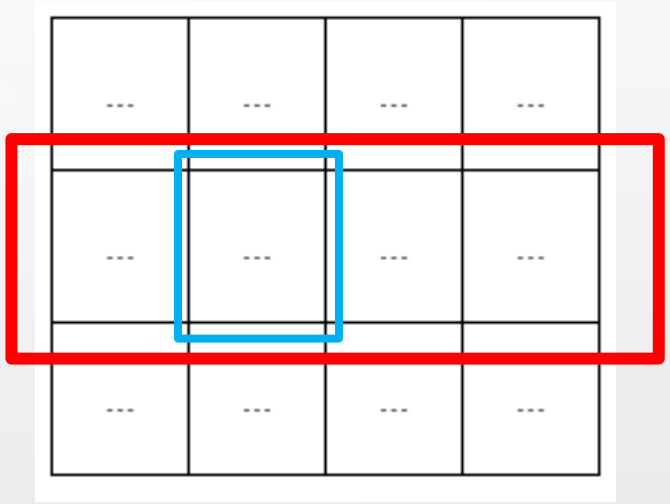
table = tableau

tr = « table row » = ligne

td = « table data cell » = cellule

th = « table header » = cellule d'en-tête

LES TABLEAUX



- A vous de jouer : créez un tableau à deux dimensions contenant les données suivantes :

Génération	Année	Pokémon
1ère génération	1996	151
2ème génération	1999	100
3ème génération	2002	135
4ème génération	2006	107
5ème génération	2010	156
6ème génération	2013	72
7ème génération	2016	88
8ème génération	2019	92
Total		901

Pour mettre en exposant, utiliser la balise `<sup>texte</sup>`

table = tableau
tr = « table row » = ligne
td = « table data cell » = cellule
th = « table header » = cellule d'en-tête

PARENTHÈSE : L'INSPECTEUR DU NAVIGATEUR

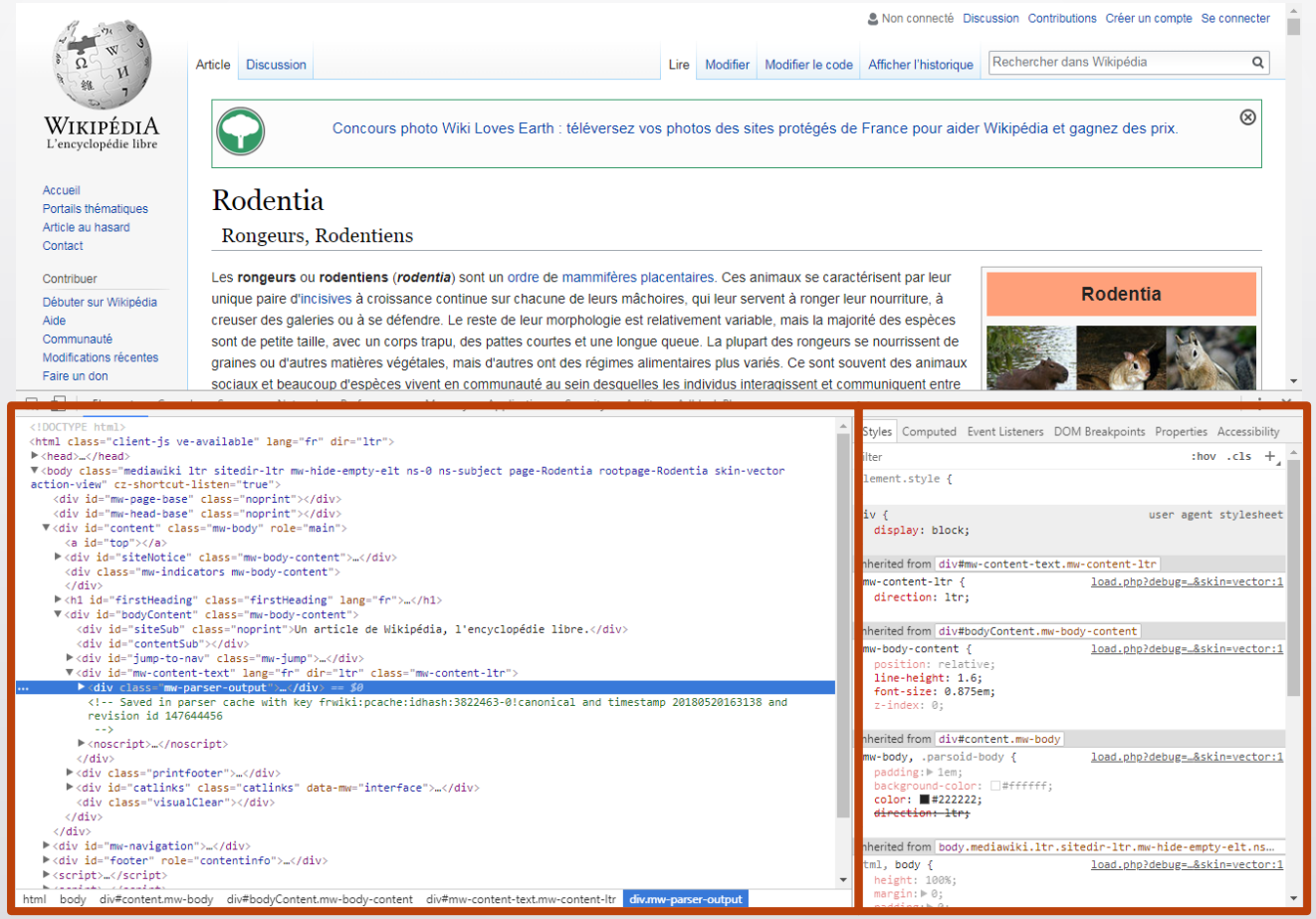
- Sur tous les navigateurs actuels, il existe un outil d'aide au développement tantôt appelé « Outils de développement », « Dev tools », « Inspecteur », etc.

- Pour l'ouvrir, appuyez sur **F12** ou faites un clic droit dans la page, puis « Inspecter l'élément »

HTML : BALISES DE BASE

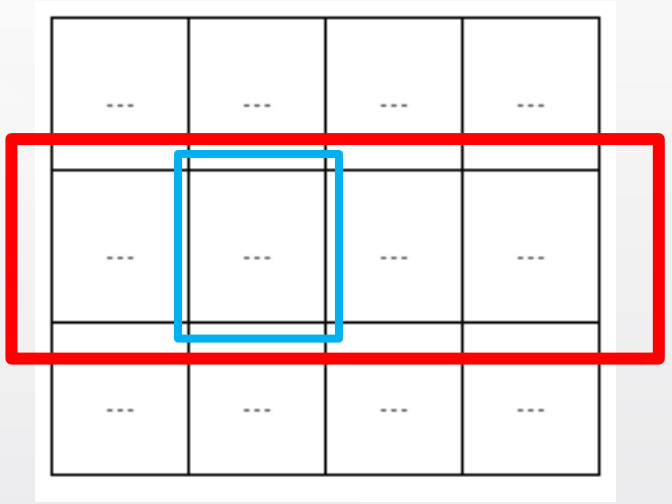
PARENTHÈSE : L'INSPECTEUR DU NAVIGATEUR

HTML de la page



CSS appliqué sur l'élément sélectionné

LES TABLEAUX



- Inspectez votre tableau ; qu'observez-vous ? (Chrome)

```
<html>
▶ <head>…</head>
▼ <body>
  ▼ <table>
    ▼ <tbody>
      ▼ <tr>
          <th>Génération</th>
          <th>Année</th>
          <th>Pokémon</th>
        </tr>
```

- Le navigateur a rajouté une balise `tbody`, dans laquelle il a mis toutes les lignes

Génération	Année	Pokémon
1ère génération	1996	151
2ème génération	1999	100
3ème génération	2002	135
4ème génération	2006	107
5ème génération	2010	156
6ème génération	2013	72
7ème génération	2016	88
8ème génération	2019	92
Total		901

LES TABLEAUX

- En fait, toujours dans un souci de structuration des données
  (car ça n'a aucun impact sur l'affichage… mais ça peut aider le développeur CSS),
  il existe 3 balises pour identifier les sections d'un tableau :

`<thead>`

`<tbody>`

`<tfoot>`

Génération	Année	Pokémon
1ère génération	1996	151
2ème génération	1999	100
3ème génération	2002	135
4ème génération	2006	107
5ème génération	2010	156
6ème génération	2013	72
7ème génération	2016	88
8ème génération	2019	92
Total		901

A vous de jouer : corrigez votre tableau

LES TABLEAUX

```
1  <!doctype html>
2  <html>
3      <head>
4          <title>Liste des générations de Pokémon</title>
5      </head>
6      <body>
7          <table>
8              <thead>
9                  <tr>
10                     <th>Génération</th>
11                     <th>Année</th>
12                     <th>Pokémon</th>
13                 </tr>
14             </thead>
15             <tbody>
16                 <tr>
17                     <td>1<sup>ère</sup> génération</td>
18                     <td>1996</td>
19                     <td>151</td>
20                 </tr>
21                 <tr>
22                     <td>2<sup>ème</sup> génération</td>
23                     <td>1999</td>
24                     <td>100</td>
25                 </tr>
26                 ...
27             </tbody>
28             <tfoot>
29                 <tr>
30                     <td>Total</td>
31                     <td></td>
32                     <td>901</td>
33                 </tr>
34             </tfoot>
35         </table>
36     </body>
37 </html>
```


Le `thead` n'étant en général constitué que d'une ligne, il est courant d'oublier le `tr`

LES TABLEAUX

- Dernière étape : on souhaite fusionner des cellules

Génération	Année	Pokémon
1ère génération	1996	151
2ème génération	1999	100
3ème génération	2002	135
4ème génération	2006	107
5ème génération	2010	156
6ème génération	2013	72
7ème génération	2016	88
8ème génération	2019	92
Total		901

Cette cellule doit occuper 2 colonnes

Pour ce faire, on utilise les attributs de `td`/`th`

- `colspan="N"` : la cellule occupe N colonnes
- `rowspan="N"` : la cellule occupe N lignes

A vous de jouer : modifiez votre cellule « Total »

Afin de mieux voir vos modifications, vous pouvez ajouter dans votre balise **head** la balise suivante :

```
<style>
    table { border-collapse: collapse; }
    td, th { border: 1px solid black; }
</style>
```

LES TABLEAUX

- Exercice : reproduisez ce tableau

State of Health	Fasting Value		After Eating
	Minimum	Maximum	2 hours after eating
Healthy	70	100	Less than 140
Pre-Diabetes	101	126	140 to 200
Diabetes	More than 126	N/A	More than 200

LES SECTIONS

- Avec tout ce que nous venons de voir, vous avez de quoi structurer du contenu simple

- Il existe des balises qui permettent d'agencer les sections de contenu :
  - `<header>` : En-tête de la page
  - `<nav>` : Navigation de la page
  - `<footer>` : Pied de page
  - `<section>` : Section de page
  - `<article>` : Article (contenu autonome)
  - `<aside>` : Informations complémentaires

- Ces balises n'ont aucun impact sur le design du site (sauf si vous leur appliquez du CSS) ; leur but est uniquement sémantique (**SEO**)

LES BALISES GÉNÉRIQUES

- Toutes les balises que nous avons vues jusqu'ici ont un sens sémantique, c'est-à-dire qu'elles signifient quelque chose

- Lorsqu'on souhaite créer des divisions dans le contenu sans sens sémantique, on peut utiliser les balises `<div>` et `<span>`
  - `<div>` : « division » générique, c'est un bloc d'éléments qui contient en général d'autres balises
  - `<span>` : c'est un élément « en ligne » qui ne contient en général que du texte (on peut en trouver au sein des paragraphes, par exemple)

- Ces deux balises sont souvent utilisées à tort **absolument partout**, voire **mal**
  (on ne met pas une `div` dans un `span` !)
  On ne doit les utiliser **que lorsqu'aucune autre ne semble pertinente** !

LES BALISES GÉNÉRIQUES

- `div`, `span` : On ne doit les utiliser **que lorsqu'aucune autre ne semble pertinente** !

- **Intérêt 1 : Le SEO**
  - Les moteurs de recherche **interprètent** l'en-tête, le pied de page, les sections, etc.
  - Ils en déduisent les parties importantes de votre page, celles qui contiennent le **cadre du site** (menus, bannière, etc.) et celles qui contiennent le **contenu textuel** (corps d'un article, etc.)

- **Intérêt 2 : L'accessibilité**
  - Les malvoyants et les aveugles utilisent des **outils qui interprètent et lisent à voix haute ou affichent en braille** les sites web ; un site bien construit sera donc mieux compris par vos visiteurs handicapés

- **Intérêt 3 : La maintenabilité**
  - « *Toujours développer comme si la personne qui passera derrière vous était un psychopathe qui connaît votre adresse* »

LES BALISES GÉNÉRIQUES

- **Intérêt 3 : La maintenabilité**
  - « *Toujours développer comme si la personne qui passera derrière vous était un psychopathe qui connaît votre adresse* »

```
<div>
    <img src="mabanniere.jpg" />
    <div>
        <div>
            <div><a href="/menu1">Menu 1</a></div>
            <div><a href="/menu2">Menu 2</a></div>
        </div>
    </div>
</div>
<div>
    <span>Titre du site</span>
    <div>
        <span>Ceci est un article</span>
        <div>Par Nicolas Lethuillier</div>
        <div>Ceci est le contenu textuel</div>
    </div>
</div>
```

```
<header>
    <img src="mabanniere.jpg" />
    <nav>
        <ul>
            <li><a href="/menu1">Menu 1</a></li>
            <li><a href="/menu2">Menu 2</a></li>
        </ul>
    </nav>
</header>
<section>
    <h1>Titre du site</h1>
    <article>
        <h2>Ceci est un article</h2>
        <div>Par Nicolas Lethuillier</div>
        <p>Ceci est le contenu textuel</p>
    </article>
</section>
```

*Maintenir un code avec des **`div`** et des **`span`** partout, c'est comme devoir retrouver une voiture spécifique sur un parking avec pour seul indice : « c'est une voiture »*

A RETENIR

Contenu

`<p>` : paragraphe
`<h1>…<h6>` : titre
`<img src="…" alt="…" />` : image
`<a href="…">` : lien

Listes

`<ul>` : liste non ordonnée
`<ol>` : liste ordonnée
`<li>` : item de liste

Tableaux

`<table>` : tableau
`<thead>,<tbody>,<tfoot>` : structure du tableau
`<tr>` : ligne
`<th>` : cellule d'en-tête
`<td>` : cellule

Structure

`<header>` : en-tête de la page
`<nav>` : navigation de la page
`<footer>` : pied de page
`<section>` : section de page
`<article>` : article (contenu autonome)
`<aside>` : informations complémentaires

Et quand il n'y a pas de balise plus adéquate :
`<div>` : bloc de division du contenu
`<span>` : « bout » de contenu textuel

HTML : NOTIONS AVANCÉES

Pour faire des trucs encore plus poussés

- Maintenant que nous avons vu une liste certes **non exhaustive** mais déjà bien suffisante de balises usuelles, nous allons nous intéresser à des détails un peu plus techniques

- Nous aborderons notamment :
  - Le contenu de la balise `<head>`
  - Les identifiants
  - Les formulaires
  - Les balises méconnues
  - Les « faux-amis » du style (`<b>`, `<strong>`, `<i>`, `<em>`, etc.)
  - Le dilemme du `<br/>`
  - Les entités HTML

LA BALISE <HEAD>

- Rappel : la balise `head` contient en général au minimum deux balises

```
<!DOCTYPE html>
<html>
    <head>
        <title>Le titre de ma page (qui apparaît sur l'onglet du navigateur)</title>
        <meta charset="UTF-8"/>
    </head>
    <body>
        ... contenu de votre page (ce qui sera visible à l'écran)
    </body>
</html>
```

Indique l'encodage de la page, c'est-à-dire le jeu de caractères à utiliser pour l'afficher (UTF-8 dans 99,9% des cas)

Les balises `meta` servent à apporter tout un tas d'informations sur le document
On peut mettre autant de balises `meta` qu'on le souhaite

LA BALISE <HEAD>

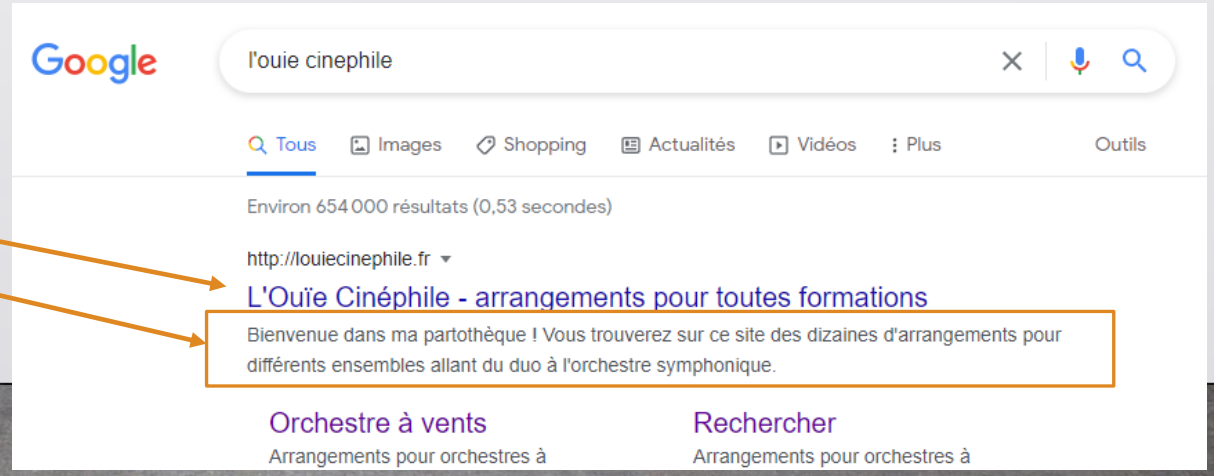
- La meta d'encodage (charset) est un peu particulière
- En général, une balise meta a un attribut name et un attribut content

```
<meta name="description" content="Bienvenue dans ma partothèque ! Vous trouverez sur ce site des dizaines
                                  d'arrangements pour différents ensembles allant du duo à l'orchestre symphonique." />
```

- Par exemple, on peut définir une « description » qui sera affichée par les moteurs de recherche

Balise `<title>`

Balise `<meta name="description">`



LA BALISE <HEAD>

- Vous croiserez sans doute d'autres balises `meta`, comme celles inventées par Facebook (préfixées par « `og:` », pour Open Graph)

```
<meta property="og:image" content="https://developer.mozilla.org/static/img/opengraph-logo.png">
<meta property="og:description" content="MDN Web Docs fournit des
                                         informations sur les technologies web ouvertes comme HTML, CSS et les API
                                         utilisées pour les sites web et les applications web. On y trouve également
                                         de la documentation destinées aux développeurs à propos des produits
                                         Mozilla tels que les outils de développement de Firefox.">
<meta property="og:title" content="Mozilla Developer Network">
```

Attention : Open Graph utilise `property` au lieu de `name`

LA BALISE <HEAD>

- C'est aussi dans le head que l'on va intégrer :
  - Nos feuilles de style CSS (cf. cours suivant)
  - Nos scripts JavaScript (cf. bien plus tard)

Obligatoire

Adresse de la feuille de style

```
<link rel="stylesheet" href="styles/mon-fichier.css" />

<script type="text/javascript" src="scripts/mon-script.js"></script>
```

Obligatoire

Adresse du script

Attention
<script> n'est pas auto-fermante !

LA BALISE <HEAD>

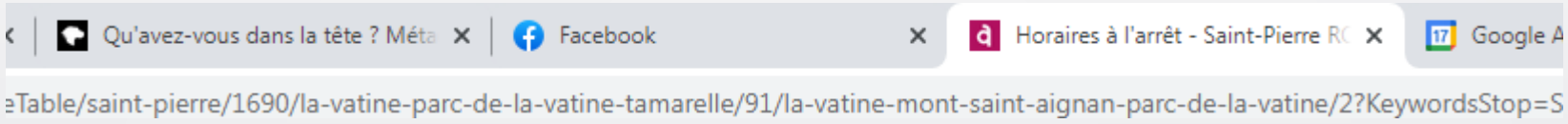
- Et puisqu'on parle de CSS, on peut aussi en intégrer dans une balise style :

```
<style>
    table { border-collapse: collapse; }
    td, th { border: 1px solid black; }
</style>
```

- Et on peut faire la même chose avec JavaScript...
  mais on verra ça lors du cours JavaScript

LA BALISE <HEAD>

- Dernière métadonnée que l'on peut définir dans le head : la **favicon**



- C'est un détail… mais pas des moindres,
  quand comme moi vous avez 47 onglets ouverts

```
<link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
```

- On retrouve la même balise que pour le CSS,
  mais avec valeurs d'attribut différentes

PARENTHÈSE : LA LANGUE DU DOCUMENT

- Toutes les balises peuvent avoir un attribut `lang` qui indique la langue dans laquelle est son contenu

- Il est très important (toujours SEO, accessibilité, etc.) de l'indiquer dans la balise `html` :

```
<!DOCTYPE html>
<html lang="fr">
▼<head>
```

Ainsi, Google et tous les outils d'interprétation savent que votre contenu est en français

Vous pouvez également mettre des parties de contenu dans une langue différente :

```
<p>C'est alors que DeNiro, s'entraînant devant son miroir, lui assène des <span lang="en">you're talking to me?</span> sur différents tons.</p>
```

LES IDENTIFIANTS

- Tout élément HTML peut avoir un attribut id

```
<body id="page-775">
```

```
<img src="mon-image.jpg" id="cette-image-la" />
```

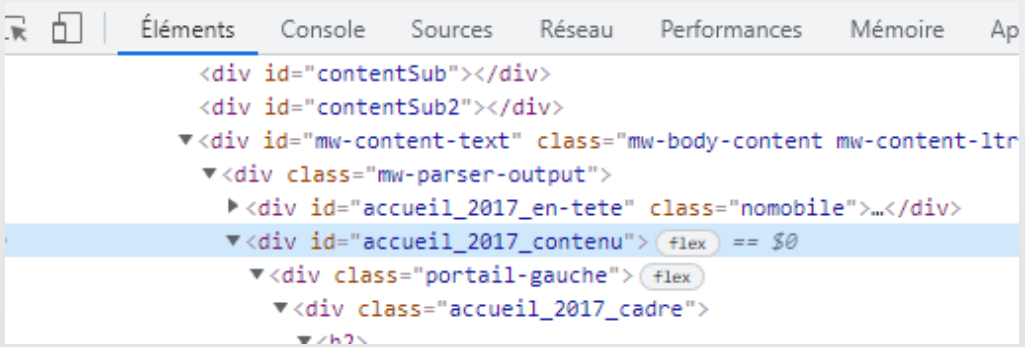
```
<p>J'ai <span id="nombre-sous">10</span> sous dans ma poche</p>
```

- C'est un identifiant **unique** qui peut servir :
  - Au CSS, pour cibler cet élément en particulier
  - Au JavaScript, pour cibler cet élément
  - À la **navigation**
  - Aux **formulaires**

UNIQUE !

LES IDENTIFIANTS

- C'est un identifiant **unique** qui peut servir :
  - …
  - À la **navigation**
  - …



Si je vais à l'adresse https://fr.wikipedia.org/#accueil_2017_contenu,
la fenêtre scrolle automatiquement au niveau de l'élément

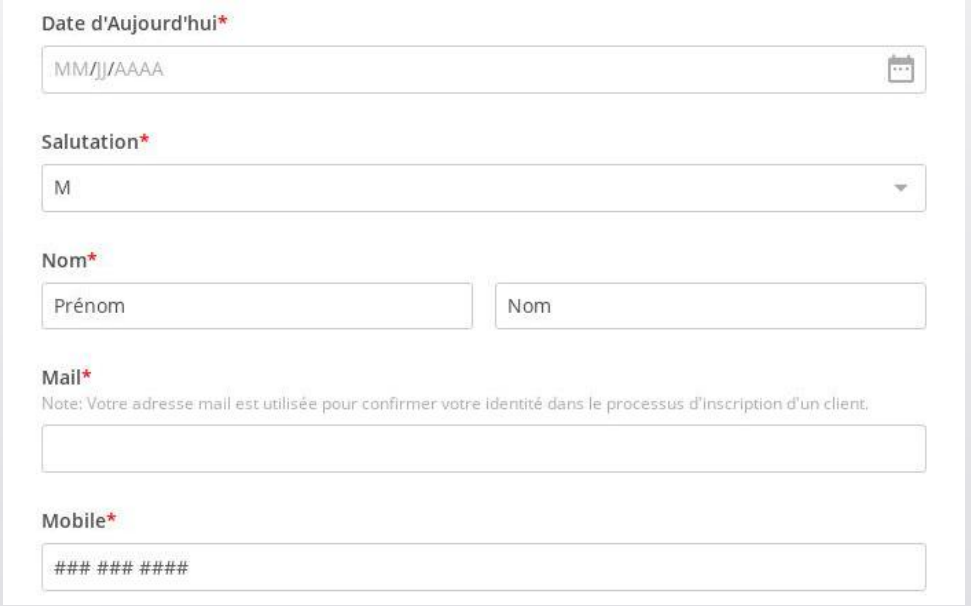
Le # indique qu'il y a un identifiant à cibler

LES FORMULAIRES

- Un formulaire est un élément HTML qui contient des champs, lesquels permettent de transmettre des informations au serveur

- Les champs peuvent être :
  - Du texte libre
  - Un menu déroulant
  - Une date
  - Une case à cocher
  - Etc.

- Lorsqu'on envoie les données d'un formulaire, on dit qu'on le « soumet »



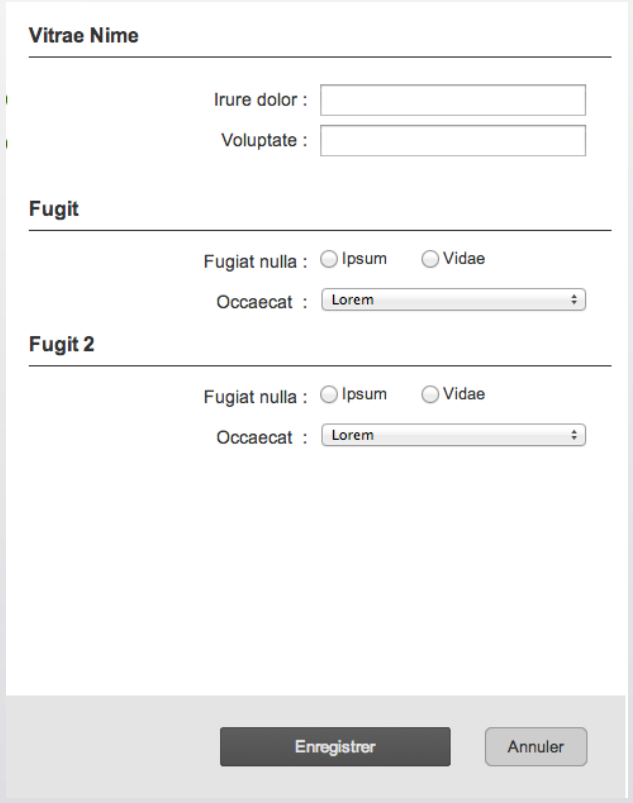
LES FORMULAIRES

- La première étape pour construire un formulaire est d'utiliser la balise `form`

- Elle nécessite deux attributs :
  - `method` : `get` ou `post` (cf. back-end)
  - `action` : la page vers laquelle envoyer les données

```
<form method="post" action="subscribe.php">
    ... contenu du formulaire ...
</form>
```

- Lorsque le formulaire est soumis, l'utilisateur est redirigé vers la page du `action`



LES FORMULAIRES

- On peut remplir le formulaire de champs à l'aide des balises :
  - `<input />`
  - `<textarea></textarea>`
  - `<select>`
    `<option></option>`
    `</select>`
  - `<button></button>`

- On pourra également, optionnellement, utiliser les balises :
  - `<label></label>`
  - `<fieldset>`
    `<legend></legend>`
    `</fieldset>`

LES FORMULAIRES : <INPUT>

- La balise `<input>` est la plus courante, car elle couvre la plupart des besoins grâce à son attribut `type` :
  - button
  - **checkbox**
  - color
  - **date**
  - datetime-local
  - email
  - **file**
  - **hidden**
  - image
  - month
  - **number**
  - **password**
  - **radio**
  - range
  - reset
  - search
  - **submit**
  - tel
  - **text**
  - time
  - url
  - week
- Dans tous les cas, elle est toujours auto-fermante
- Explorons ensemble les types les plus courants

LES FORMULAIRES : <INPUT>

- **checkbox**
- date
- file
- hidden
- number
- password
- radio
- submit
- text

```
<input type="checkbox" />
```

- L'input de type checkbox crée une case à cocher

- On peut s'en servir pour un choix unique ou pour une liste à choix multiples :

Avec l'attribut checked, on lui spécifie qu'elle doit être cochée par défaut

```
<input type="checkbox" checked />
```

LES FORMULAIRES : <INPUT>

- checkbox
- **date**
- file
- hidden
- number
- password
- radio
- submit
- text

```
<input type="date" />
```

- L'`input` de type `date` crée un sélecteur de date
- **Attention** : son style dépend du navigateur et il est difficile à personnaliser (CSS)
- On utilise alors souvent des bibliothèques dédiées (Google → « *datepicker js* »)

LES FORMULAIRES : <INPUT>

- checkbox
- date
- **file**
- hidden
- number
- password
- radio
- submit
- text

```
<input type="file" />
```

- L'`input` de type `file` crée un sélecteur de fichier
- **Attention** : comme les dates, son style dépend du navigateur et il est difficile à personnaliser (CSS)
- Avec l'attribut `multiple`, on peut sélectionner plusieurs fichiers

LES FORMULAIRES : <INPUT>

- checkbox
- date
- file
- **hidden**
- number
- password
- radio
- submit
- text

```
<input type="hidden" name="id" value="42" />
```

- L'input de type hidden est invisible pour l'utilisateur mais passe une valeur quand même
- Il est utilisé par les développeurs pour transmettre une information qui ne regarde pas l'utilisateur
- On lui assigne toujours une value qui est la valeur qui sera transmise à la soumission du formulaire

LES FORMULAIRES : <INPUT>

- checkbox
- date
- file
- hidden
- **number**
- password
- radio
- submit
- text

```
<input type="number" />
```

- L'input de type number n'accepte que les nombres
- Les navigateurs proposent souvent des flèches pour incrémenter et décrémenter la valeur (+1/-1)
- Les navigateurs mobiles ouvrent le clavier des chiffres, et pas le clavier AZERTY

LES FORMULAIRES : <INPUT>

- checkbox
- date
- file
- hidden
- number
- **password**
- radio
- submit
- text

```
<input type="password" />
```

- L'input de type password accepte tous les caractères, mais les masque à l'utilisateur
- On assure ainsi la sécurité de la saisie vis-à-vis des regards indiscrets
- **Attention** : ça ne chiffre rien ! L'information est quand même envoyée « en clair » au serveur

LES FORMULAIRES : <INPUT>

- checkbox
- date
- file
- hidden
- number
- password
- **radio**
- submit
- text

```
<input type="radio" />
```

- L'input de type radio crée une case à cocher unique

- Contrairement aux checkbox, on choisit parmi n options :

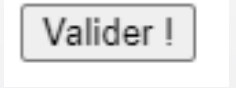
Avec l'attribut checked, on lui spécifie qu'il doit être coché par défaut

```
<input type="radio" checked />
```

LES FORMULAIRES : <INPUT>

- checkbox
- date
- file
- hidden
- number
- password
- radio
- **submit**
- text

```
<input type="submit" value="Valider !" />
```



- L'`input` de type `submit` crée un bouton de soumission
- Au clic dessus, le formulaire est envoyé avec l'ensemble des données saisies, et l'utilisateur est redirigé vers la page indiquée dans `action`
- Le texte à afficher doit être mis dans son attribut `value`

LES FORMULAIRES : <INPUT>

- checkbox
- date
- file
- hidden
- number
- password
- radio
- submit
- **text**

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

- L'`input` de type `text` est le plus courant, car il permet de saisir *n'importe quoi*
- Pourquoi ne pas mettre des `text` partout, plutôt que `date`, `number`, `password`… ?
- Parce que les navigateurs les considèrent différemment ! (expérience utilisateur ++)

LES FORMULAIRES : <INPUT>

- `checkbox`
- `date`
- `file`
- `hidden`
- `number`
- `password`
- `radio`
- `submit`
- `text`

- Quel que soit le type de l'`input`, on doit lui renseigner un attribut `name`, qui est le nom via lequel le serveur pourra récupérer sa valeur
- Ce `name` doit être unique, sauf dans le cas des boutons `radio`, pour les « *grouper* »
- On peut également renseigner l'attribut `value` pour indiquer une valeur par défaut dans le champ
- Il existe plusieurs autres attributs utile, en fonction du type… vous verrez à l'usage !

LES FORMULAIRES : <TEXTAREA>

- Les `textarea` sont comme des `input` de type `text`, mais qui accepte des sauts de ligne

```
<textarea>ABC</textarea>
```

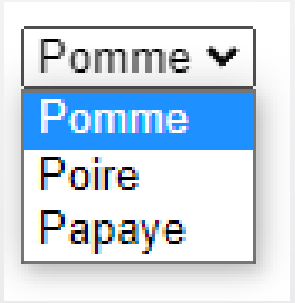
- Cette balise n'est pas auto-fermante, car au lieu de prendre une `value`, son contenu est entre les balises (pas ma faute…)

- Avec les attributs `cols` et `rows`, on spécifie le nombre de colonnes (caractères) et de lignes *(→ CSS + options navigateur)*

LES FORMULAIRES : <SELECT>

- Les `select` permettent de créer des menus déroulants, avec autant d'options que vous le souhaitez

```
<select>
  <option>Pomme</option>
  <option>Poire</option>
  <option>Papaye</option>
</select>
```



- On peut attribuer une `value` à chaque option
  Dans ce cas, la valeur envoyée sera le contenu de `value`, et pas le texte affiché

- Avec l'attribut multiple, on peut sélectionner plusieurs valeurs

LES FORMULAIRES : <BUTTON>

- Les `button` sont comme des `input` de type `button`,
  sauf qu'on peut mettre tout le HTML qu'on veut à l'intérieur

```
<button>
    <img src="une-icone.jpg" />
    Ceci est un <strong>contenu</strong> !
</button>
```

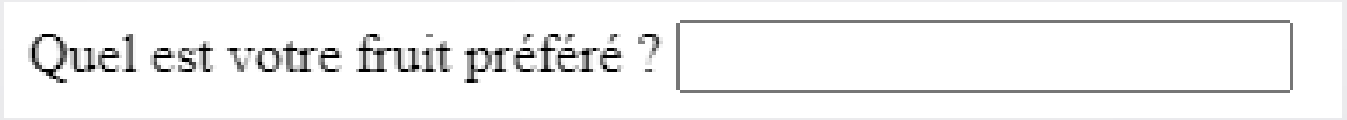
- On peut mettre toutes les balises imaginables dans un `button`

- Un `button` sans `type` se comporte comme un `input` de type `submit`
  Si on ne souhaite pas soumettre le formulaire (mais éventuellement faire une action JavaScript au clic, par exemple), on doit lui ajouter `type="button"`

LES FORMULAIRES : <LABEL>

- Les labels sont importants, tout comme leur bon usage

```
<label for="fruit-prefere">Quel est votre fruit préféré ?</label>
<input type="text" id="fruit-prefere" name="fruit" />
```



- On cible dans `for` l'`id` du champ associé

- Non seulement il y a un bénéfice sémantique, mais lorsque l'utilisateur clique sur la question, le focus se met automatiquement dans le champ !

LES « FAUX-AMIS » DU STYLE

- Il existe plusieurs balises pour mettre en forme du contenu

- Il existe plusieurs balises pour faire ressortir du contenu spécifique :
  - `<em>` : indique du texte en emphase
  - `<i>` : met en italique
  - `<strong>` : met du texte en exergue
  - `<b>` : met du texte en gras

- Les balises `em` et `strong` ont un **sens sémantique** : il s'avère que les navigateurs les affichent en italique/gras par défaut (car c'est commode), mais ils pourraient demain changer d'avis et afficher les `strong` en plus gros et les `em` en rouge souligné

- Les balises `i` et `b` ont été inventées pour de la mise en forme ; elles n'ont aucun sens sémantique

LE DILEMME DU

- La balise `<br/>` (*break*) permet d'aller à la ligne

- Il y a trois types de développeur :
  - Ceux qui l'utilise à outrance
  - Ceux qui la bannissent à jamais, persuadés qu'elle est dépréciée
  - Ceux qui réfléchissent deux minutes

- `<br/>` sert à aller à la ligne au sein d'un même paragraphe
  Dans ce cas, elle est parfaitement adaptée et son usage louable

- `<br/>` **ne doit pas** servir à déplacer un élément HTML à la ligne du dessous,
  ou pire : faire de l'espace entre deux blocs

LES BALISES MÉCONNUES

- Nous avons découvert ensemble **45** balises HTML sur environ **115**

- Il en existe donc de nombreuses autres avec des fonctionnalités et des sens sémantiques particuliers

- Vous n'êtes pas obligés de toutes les connaître…
  mais ayez la curiosité de les consulter au moins une fois,
  pour vous faire une idée de ce qui existe

https://developer.mozilla.org/fr/docs/Web/HTML/Element

LES ENTITÉS HTML

- Avant d'avoir à disposition l'UTF-8, on utilisait, pour afficher les accents, des entités HTML :

```
<p>Apr&eagrave;s cette s&eacute;ance de cin&eacute;ma, nous irons au th&eacute;&acirc;tre !</p>
```

- Les entités commencent toujours par & et finissent par ;

- On ne les utilise quasiment plus aujourd'hui…
  mais ne soyez pas surpris si vous en croisez

https://dev.w3.org/html5/html-author/charref

LES COMMENTAIRES

- Tous les langages de programmation permettent de mettre des commentaires dans son code, afin de renseigner les futurs développeurs sur l'intérêt de telle ou telle section

```
<div>
    <!-- Menu de sélection du fruit -->
    <select>
      <option>Pomme</option>
      <option>Poire</option>
      <option>Papaye</option>
    </select>
</div>
```

- Si votre HTML est bien construit, ils ne sont en général pas nécessaires
- **Attention** : les visiteurs de votre site peuvent les voir !

VERS CSS

Introduction au cours d'après…

LES CLASSES

- Au même titre que l'`id`, tout élément HTML peut avoir un attribut `class`

- La ou les classes d'un élément sont des mots-clés qui le distingue des autres éléments

- Les classes sont utilisées en CSS pour sélectionner cet élément et pas les autres

```
<p>Ceci est un paragraphe</p>
<p class="highlight">Ceci est un paragraphe un peu particulier</p>
<p>Ceci est un autre paragraphe</p>
```

```
<body class="no-js high-contrast theme2">
```

L'ATTRIBUT STYLE

- Tout élément HTML peut également avoir un attribut style
  dans lequel on peut mettre du CSS qui sera appliqué uniquement sur cet élément

- Attention : ceci n'est pas une bonne pratique pour plusieurs raisons
  - Les responsabilités de chaque technologie :
    HTML est fait pour structurer l'information, pas lui appliquer du style
  - La maintenance :
    il est toujours plus propre de ne pas mélanger les langages dans le même fichier
  - La maintenance, le retour :
    le style défini dans l'attribut prend la priorité sur le fichier CSS, donc difficile de s'y retrouver !

LES CONVENTIONS DE NOMMAGE

- Tous les langages ont des conventions de nommage

- Elles sont parfois méconnues…

- En l'occurrence, en HTML, on écrit les balises en minuscules
  et les noms de fichiers, les identifiants et les classes en kebab-case :

```
<img src="images/mon-image.jpg" />
```

```
<div class="featured-content">
```

```
<input type="text" id="first-name" />
```

CONCLUSION

CONCLUSION

- Vous avez maintenant tous les éléments pour faire des pages HTML…
- … mais elles ne ressemblent pas à grand-chose, une fois dans le navigateur
- Nous allons donc voir, dans la suite, le langage qui va nous permettre de styliser tout ça : **CSS**
- Dans un futur plus ou moins proche, vous découvrirez également **JavaScript**, qui permet de rendre vos pages interactives !

CONCLUSION

- Pour mettre tout ça en œuvre, vous réaliserez un **blog** !

- Ce blog contiendra des articles, des informations sur l'utilisateur connecté, ainsi qu'un formulaire d'ajout de contenu

- Tout cela ne sera naturellement pas utilisable en tant que tel, puisque vous ne connaissez pas encore de langage back-end (ou *serveur*)