



# Full StackJS Camp

Développer une application Web avec la solution MERN

Formation assurée par : Anis Assas & Yessine Zekri

cours réalisé par Anis Assas et Yessine Zekri

# Les fondamentaux du développement web : Plan

---

- ❑ Concepts de base
- ❑ Le protocole HTTP
- ❑ Architecture Web
- ❑ Technologies Web
- ❑ Frameworks JavaScript
- ❑ Architectures des applications web
- ❑ Développement web

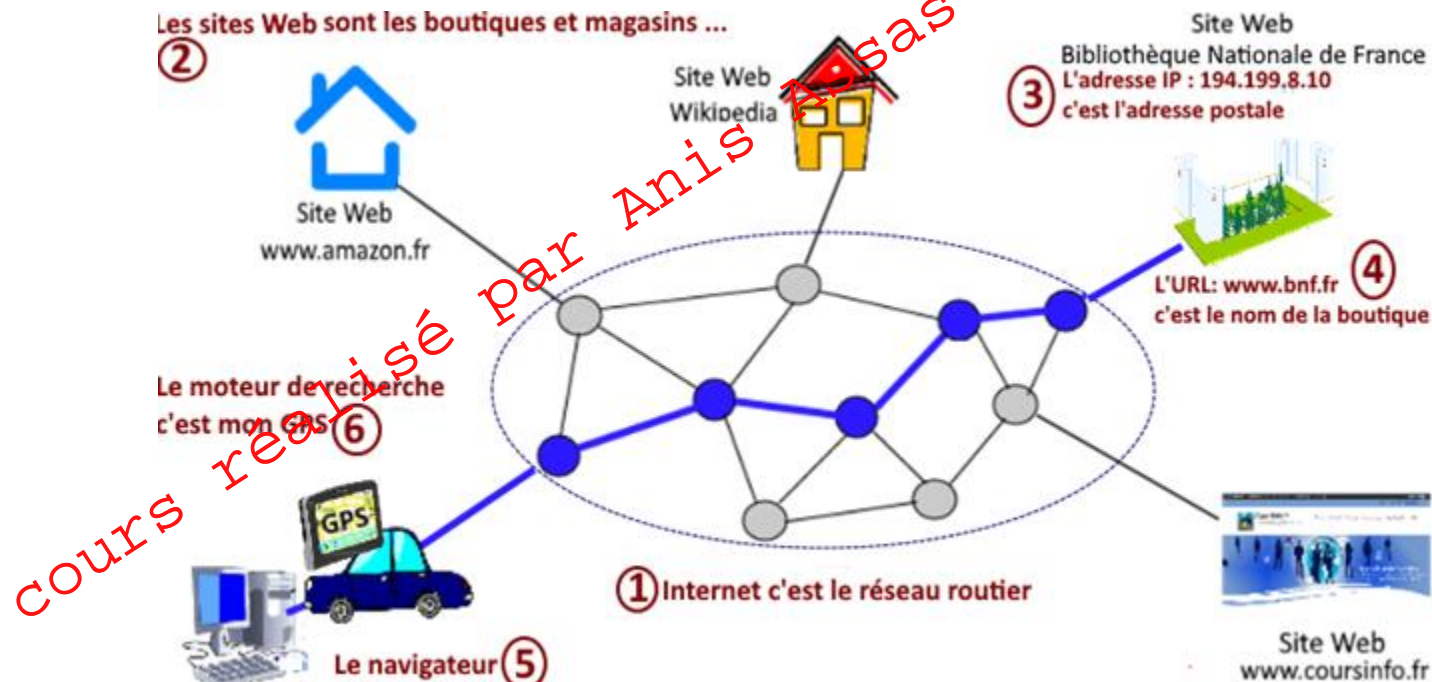


# Concepts de base

- **Internet** : le réseau informatique mondial qui rend accessibles au public des services variés :
  - Le courrier électronique,
  - La messagerie instantanée
  - ...
  - World Wide Web, ou Web
- Son architecture technique repose sur une hiérarchie de réseaux, d'où le surnom de réseau des réseaux.
- Internet est devenu populaire par l'apparition du **World Wide Web**.
- Le **Web** n'est qu'une des applications d'**Internet**.

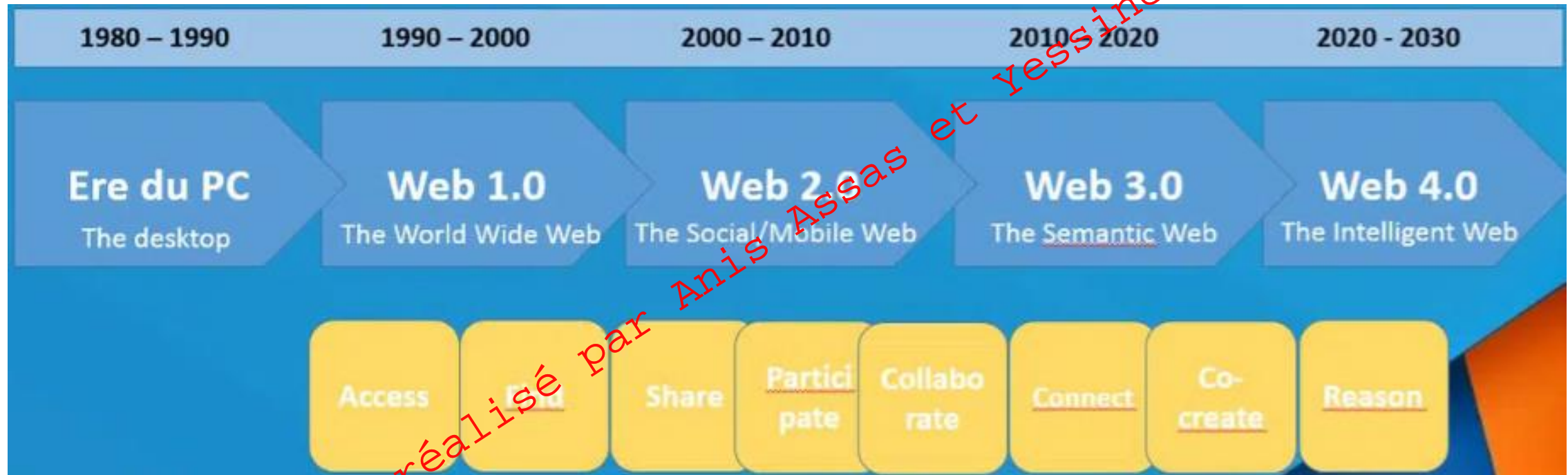
# Concepts de base : le Web

- **Le World Wide Web**, la « toile (d'araignée) mondiale », ou le Web, ou le WWW, est un système hypertexte généralement public fonctionnant sur Internet
- Il permet de consulter, avec un navigateur, des pages accessibles sur des sites. L'image de la toile d'araignée vient des **hyperliens** qui lient les pages web entre elles.



# Concepts de base : Evolution du Web

- Du web 1.0 au web 4.0 :



# Concepts de base : Evolution du Web

- **Du web 1.0 au web 4.0 :**

- ❑ **Web 1.0 :** le web statique centré sur la distribution d'informations majoritairement par les grandes entreprises.
- ❑ **Web 2.0 :** le web social qui privilégie la dimension de **partage** et d'échange d'informations et de contenus (textes, vidéos, images ou autres).
- ❑ **Web 3.0 :** le web sémantique qui vise à organiser la masse d'informations disponibles en fonction du **contexte** et des besoins de chaque utilisateur, en tenant compte de sa localisation, de ses préférences, etc.
- ❑ **Web 4.0 :** le web intelligent qui vise à immerger l'individu dans un **environnement digital intelligent** de plus en plus prégnant. Basé sur la communication sans fil reliant les personnes et les objets.
- ❑ **Web 5.0 :** le web ambiant (ou émotionnel) qui devient « une intelligence ambiante qui utilise l'IA pour relier des appareils et des services » : un internet capable de comprendre, de s'adapter aux besoins de chacun et interpréter les informations à des niveaux plus complexes, tant sur le plan émotionnel que logique.

# Concepts de base : les navigateurs

- **Logiciel de navigation :**

- Pour consulter le web, il faut disposer d'un browser fonctionnant avec le principe de client/serveur.
- Cette voie d'accès à Internet permet aux internautes de surfer sur Internet quel que soit le support utilisé : smartphone, tablette ou ordinateur.
- Offre des fonctionnalités secondaires comme l'historique de navigation, la gestion des pages favoris ou l'enregistrement des mots de passe.

- **Les navigateurs les plus populaires :** Google Chrome, Safari, Mozilla Firefox, Opera, Microsoft Edge, ...

- **Quel navigateur choisir ?** Le choix d'un navigateur web varie en fonction des besoins et des attentes de chaque internaute et surtout à sa rapidité, à la fréquence des mises à jour, l'ergonomie, l'expérience utilisateur et enfin et surtout à la sécurité.



# Concepts de base : les navigateurs

- **Versions et compatibilité des navigateurs** : <https://caniuse.com/>
- **Google Chrome** : leader mondial (le plus complet et stable avec une ergonomie minimaliste), mis à jour presque tous les 15 jours depuis sa première version en 2008.

Quel est le meilleur navigateur internet 2022 ?

Navigateur	Vitesse	Personnalisation	Ergonomie	Confidentialité	Sécurité
Chrome	□Premier	□Troisième	□Premier		□Deuxième
Safari	□Deuxième		□Deuxième	□Troisième	
Edge	□Troisième		□Troisième		□Premier
Firefox		□Deuxième		□Premier	□Troisième
Opera		□Premier		□Deuxième	

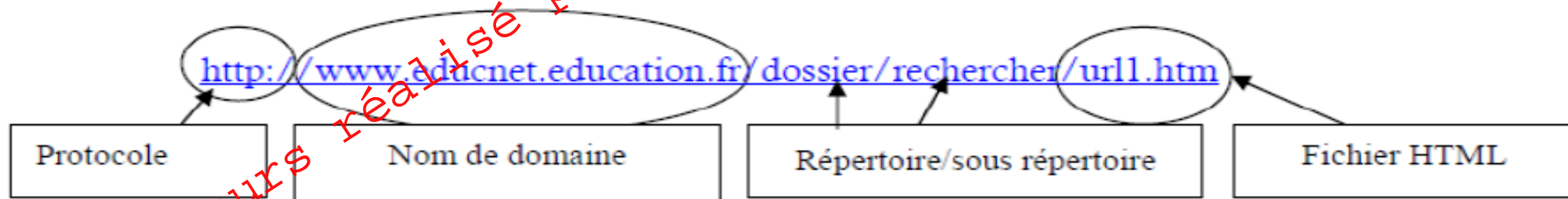
- **Parts de marché par navigateur web dans le monde ?** <https://gs.statcounter.com>

Chrome	Safari	Edge	Firefox	Samsung Internet	Opera
65.27%	18.34%	3.4%	3.29%	3.18%	2.19%
Browser Market Share Worldwide - June 2021					
Chrome for Android	Chrome 91.0	Safari iPhone	Chrome 90.0	Edge 91	Firefox 89.0
37.41%	19.54%	13.15%	3.48%	3.18%	2.05%
Browser Version Market Share Worldwide - June 2021					



# Concepts de base : une adresse URL

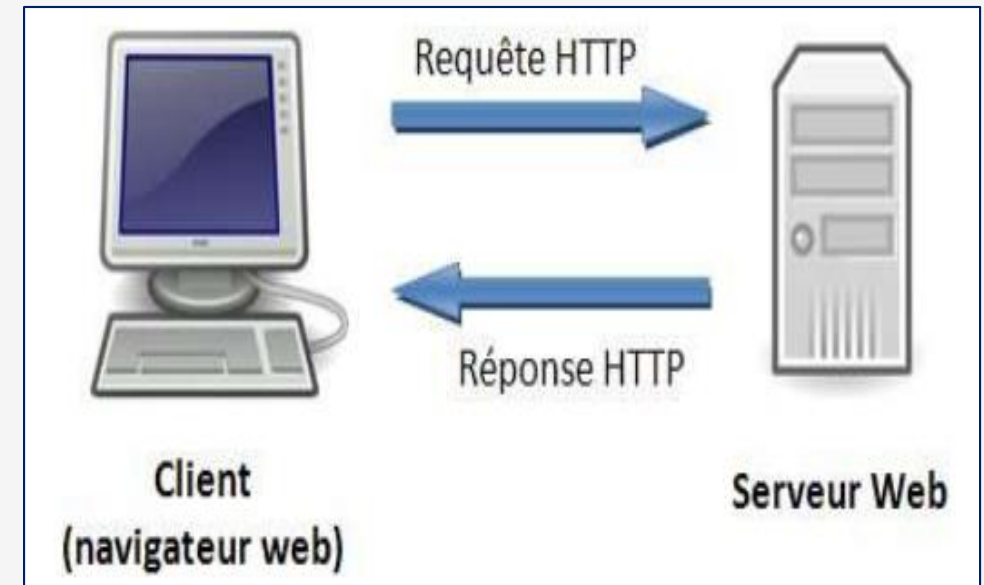
- **Une adresse URL** (Uniform Resource Locator) : localise les pages HTML qui existent dans des serveurs. Elle reflète la structuration des informations et leur emplacement sur le disque dur du serveur.
- L'URL peut être une adresse simple (généralement l'adresse d'une page d'accueil) ou une adresse compliquée.
  - **Exemple d'adresse URL simple** : <http://www.yahoo.fr>
  - **Exemple d'adresse URL compliquée** :



# Le protocole HTTP

- **HTTP : Hypertext Transfer Protocol (HTTP)** est un protocole de la couche réseau application, orienté client-serveur, qui permet le transfert de documents web tels que des documents HTML, XML et JSON.
- Une requête HTTP peut être envoyée en utilisant les méthodes suivantes :

- **GET:** Pour récupérer le contenu d'un document.
- **POST:** Pour soumissionner des formulaires (Envoyer, dans la requête, des données saisies par l'utilisateur).
- **PUT:** Pour envoyer un fichier du client vers le serveur.
- **DELETE:** Permet de demander au serveur de supprimer un document.
- **HEAD:** Permet de récupérer les informations sur un document (Type, Capacité, Date de dernière modification etc...)



# Le protocole HTTP

- Lorsque le serveur renvoie un document, il lui associe **un code de statut** renseignant ainsi le client sur le résultat de la requête.
- Principaux codes du statut HTTP :
  - **Information 1xx :**
    - 100 (Continue) : Utilisé dans le cas où la requête possède un corps(body).
    - 101 (Switching protocol) : Demander au client de changer de protocole.
  - **Succès 2xx :**
    - 200 (OK) : Le document a été trouvé et son contenu suit.
    - 201 (Created) : Le document a été créé en réponse à un PUT.
    - 202 (Accepted) : Requête acceptée, mais traitement non terminé.
    - 204 (No response) : Le serveur n'a aucune information à renvoyer.
    - 206 (Partial content) : Une partie du document suit.

# Le protocole HTTP

- **Redirection 3xx :**

- 301 (Moved) : Le document a changé d'adresse de façon permanente
- 302 (Found ) : Le document a changé d'adresse temporairement
- 304 (Not modified) : Le document demandé n'a pas été modifié

- **Erreurs du client 4xx :**

- 400 (Bad request) : La syntaxe de la requête est incorrecte
- 401 (Unauthorized) : Le client n'a pas les privilèges d'accès au document
- 403 (Forbidden) : L'accès au document est interdit
- 404 (Not found) : Le document demandé n'est pas trouvé
- 405 (Method not allowed) : La méthode de la requête n'est pas autorisée

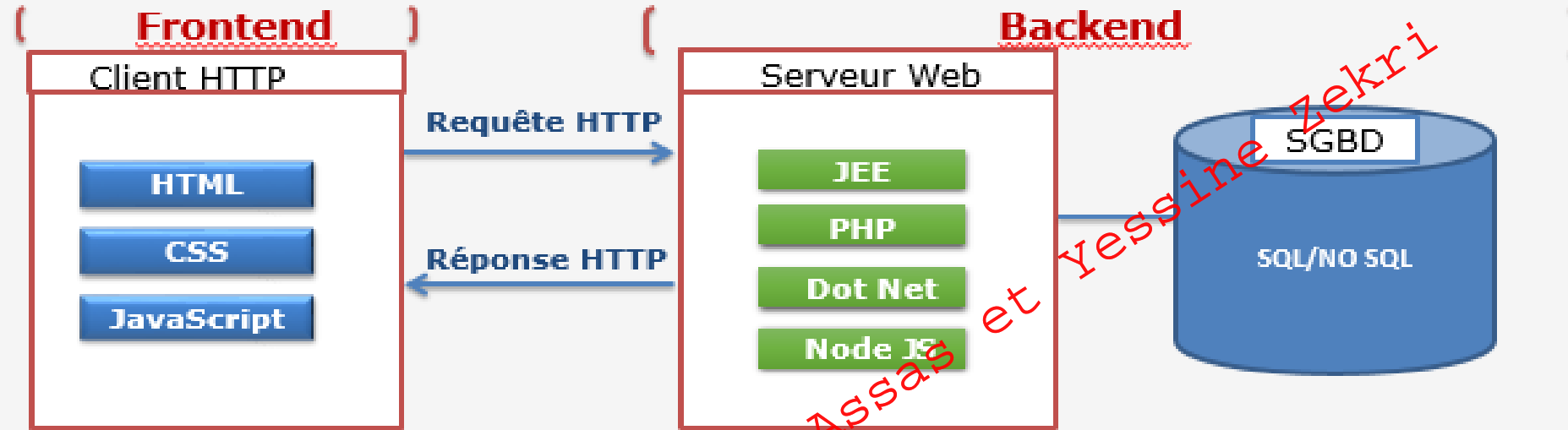
- **Erreurs du serveur 5xx :**

- 500 (Internal error) : Une erreur inattendue est survenue au niveau du serveur
- 501 ( Not implemented ) : La méthode utilisée n'est pas implémentée
- 502 ( Bad gateway) : Erreur de serveur distant lors d'une requête proxy

# Le protocole HTTP



# Architecture Web



- Un client web (Browser) communique avec le serveur web en utilisant le protocole HTTP
- Une application web se compose de deux parties:
  - La partie **Frontend** : S'occupe de la présentation des IHM côté Client :
    - Langages utilisés : HTML, CSS, JavaScript
  - La partie **Backend** : S'occupe des traitements effectués côté serveur :
    - Technologies utilisées : PHP, JEE, .Net, Node JS
- La communication entre partie Frontend et partie Backend se fait en utilisant le protocole HTTP

# Technologies web

- **HTML : HyperText Markup Language** (langage de balisage hypertexte ou **HTML**) est le langage utilisé pour décrire et définir le *contenu* d'une page web.
  - **Dernière version** : 5.2 (21 décembre 2017)
  - **Standardisé par** : WHATWG (Web Hypertext Application Technology Working Group) et W3C (World Wide Web Consortium)
- **CSS : Cascading Style Sheets** (feuilles de style en cascade ou **CSS**) est utilisé pour décrire l'apparence du contenu d'une page web.
  - **Dernière version** : 3
  - **Standardisé par** : CSS Working Group
- **JavaScript**: C'est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs avec l'utilisation de **Node.js**.
  - **Dernière version** : ES8 abréviation de « ECMAScript8 » (Juin 2017)
  - **Standardisé par** : ECMA





# Technologies web

- **Notion de framework :**

- Plusieurs traductions : cadriciel, environnement de développement, cadre d'applications, ...
- Ensemble de composants logiciels
- Facilitant la réalisation d'une (ou partie de l') application
- Imposant une certaine structure, logique, syntaxe...

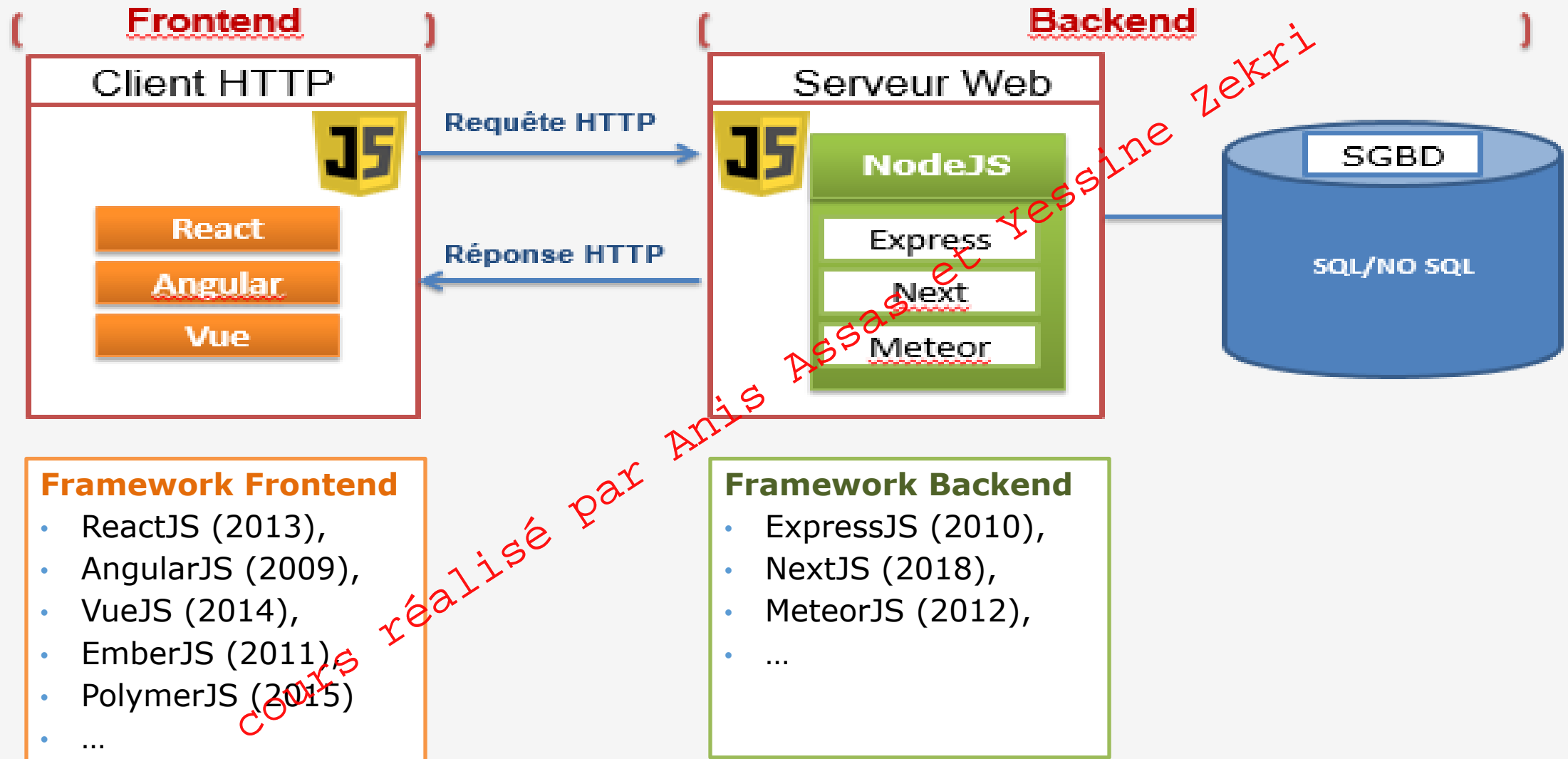
- **Frameworks applicatifs pour le développement d'applications web :**

- **Angular , React** pour JavaScript, ...
- **JSF, Spring** pour Java,
- **Laravel, Symfony** pour PHP, ...

- **Frameworks de présentation de contenu web : Bootstrap, tailwind** pour CSS, ...

- **Frameworks de persistance de données : hibernate, ...**

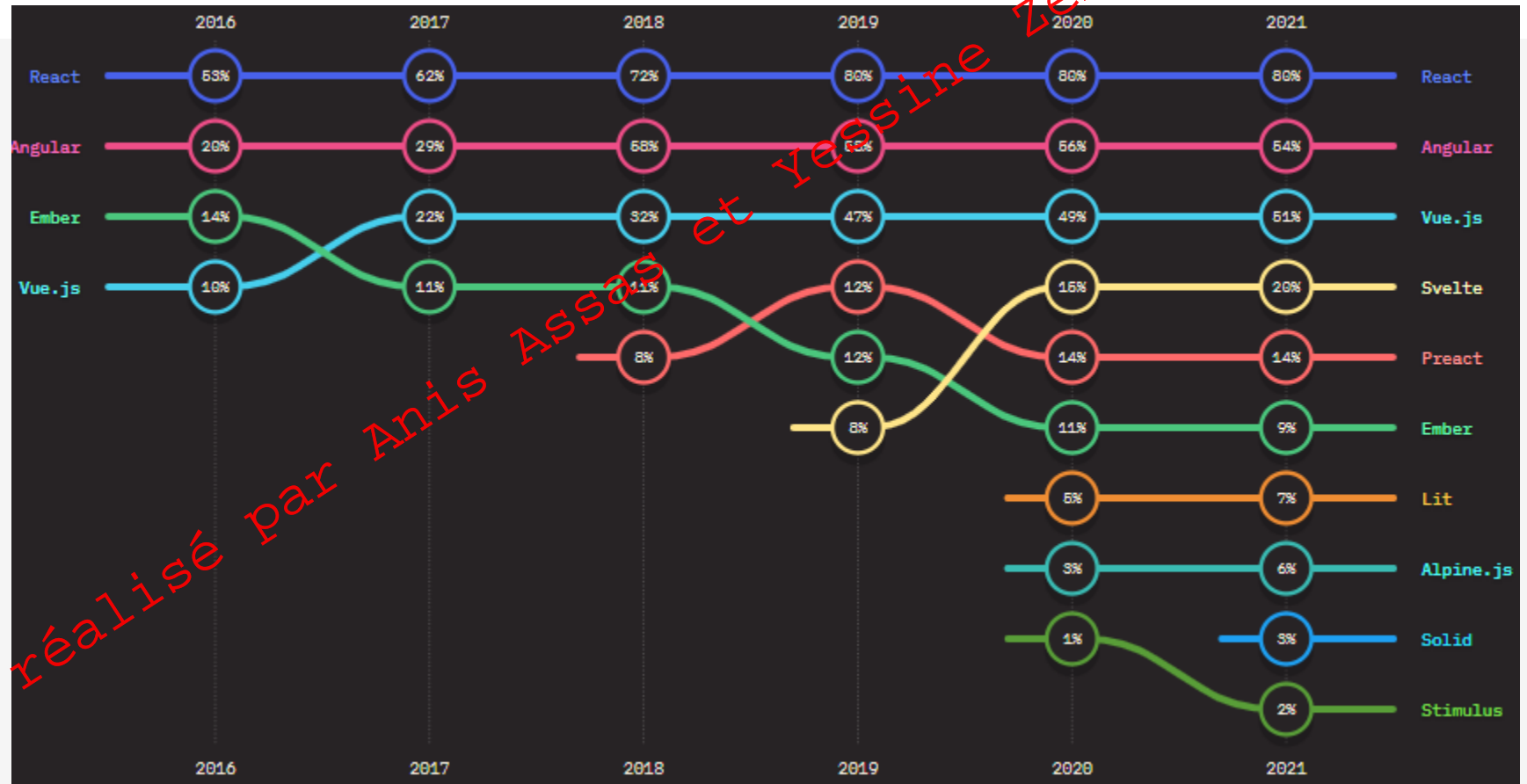
# Frameworks JavaScript



# Frameworks JavaScript

JavaScript Developer Survey - Edition 2021 (<https://2021.stateofjs.com/fr-FR/libraries/front-end-frameworks/>)

Classement des frameworks Front-End les plus populaires suivant les ratios d'utilisation



# Application web : architecture MPA

- **Multi-Page Application (MPA) :**

## Avantages

- Compatibilités avec les navigateurs
- SEO (Optimisation pour les moteurs de recherche)
- Facilité de mise en œuvre
- Navigation standard : Les utilisateurs sont familiers avec la navigation basée sur des liens

## Inconvénients

- Génération de toutes les pages auprès du serveur.
- Temps de chargement plus long
- Chaque clic sur un lien aboutit à une nouvelle requête au serveur, pour récupérer la nouvelle page, et provoque donc un rafraîchissement
- Complexité de la gestion : À mesure que l'application grandit et que le nombre de pages augmente

# Application web : architecture SPA

- **Single-Page Application (SPA) :**

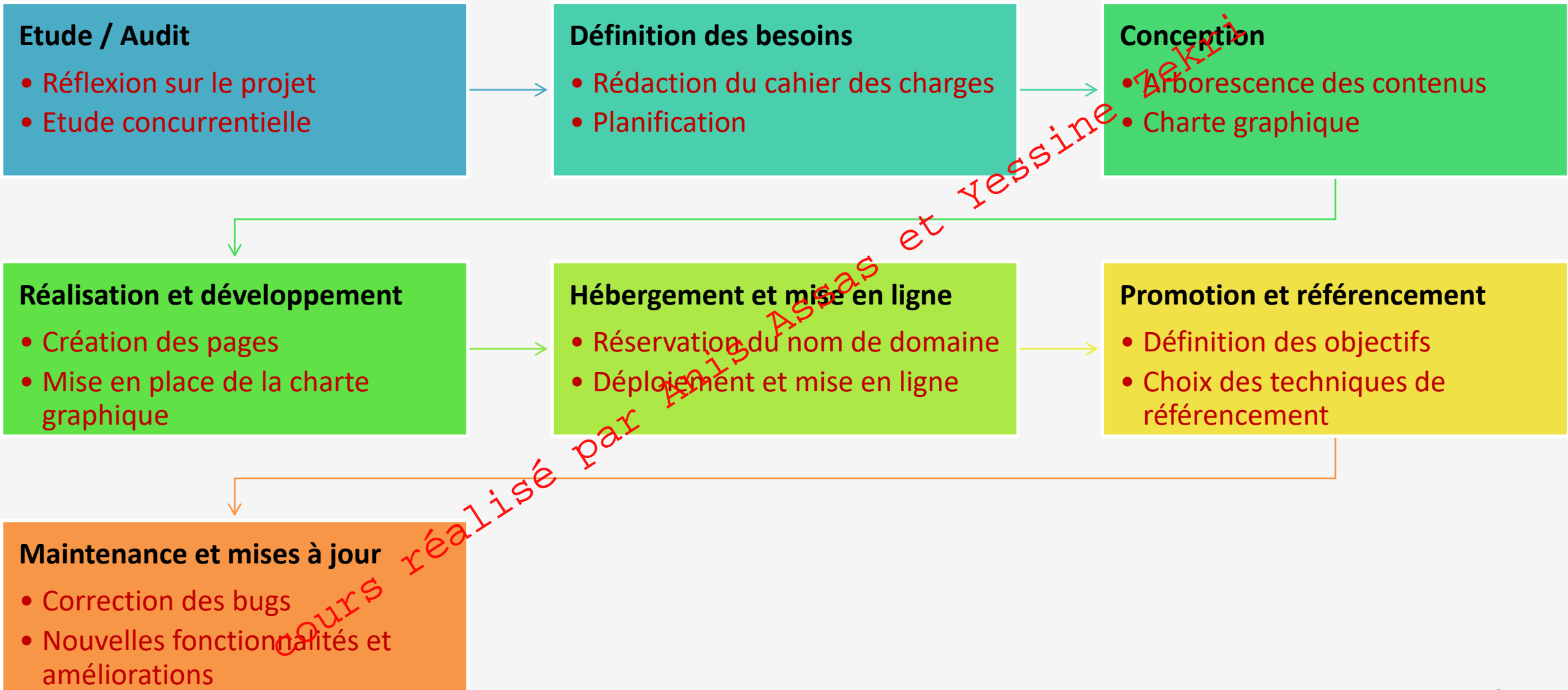
## Avantages

- Tout le contenu est chargé en une seule fois
- Temps de chargement réduit
- La maintenance de l'application est plus simple
- Réduction du trafic server

## Inconvénients

- Le site est plus long à charger (au début)
- Complexité de développement
- Une forte dépendance au JavaScript
- Mauvais référencement (SEO) car les moteurs de recherche ont du mal à indexer correctement le contenu

# Développement web : Processus



# Développement web : métier

- **Un développeur web :**

- réalise l'ensemble des fonctionnalités techniques d'un site ou d'une application web.
- conçoit des sites sur mesure ou adapte des solutions techniques existantes en fonction du projet et de la demande du client.

- **Un Développeur web full stack : "développeur à tout faire"**

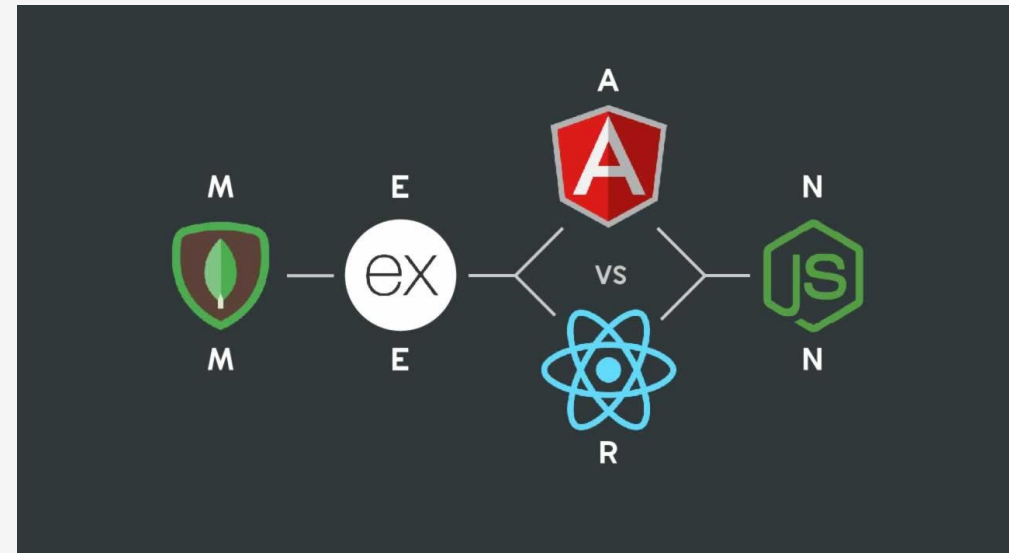
- codeur généraliste capable de réaliser la programmation d'un site ou d'une application web à la fois en front-end et back-end.
- connaît toute la chaîne de développement d'une application web et est à l'aise avec plusieurs langages et technologies.
- profil très recherché des start-ups et des jeunes projets innovants car il a un large spectre de compétences. Il peut également travailler en free-lance ou comme consultant.



# Développement web : MEAN & MERN Stack

- **MEAN & MERN Stack** : alliance de technologies (frameworks) Javascript open source utilisée pour :
  - rendre le processus de développement fluide et facile.
  - donner la capacité aux **développeurs full stack** de développer un site de A à Z sans avoir à faire intervenir une autre compétence.

- **M** : Mongo DB
- **E** : Express
  - **A** : Angular
  - **R** : React
- **N** : Node JS



# Le langage HTML 5

cours réalisé par Anis Assas et Yessine Zekri

# Le langage HTML : Plan

---

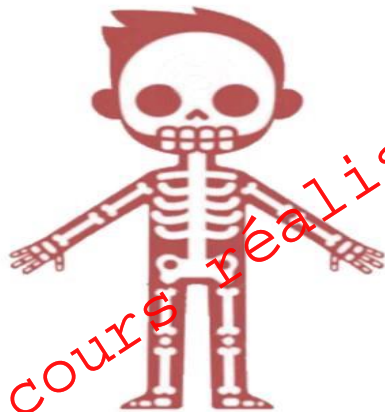
- ☐ Introduction
- ☐ Concept de balises
- ☐ Structure d'une page HTML 5
- ☐ Attributs d'une balise
- ☐ Organisation du texte et mise en forme
- ☐ Titres
- ☐ Listes
- ☐ Liens
- ☐ Tableaux
- ☐ Formulaires
- ☐ Les balises multimédia
- ☐ Les balises structurelles
- ☐ Conventions et bonnes pratiques

HTML 5



# HTML : Introduction

- **HTML : Hyper Text Markup Language**
  - Ce n'est pas un langage de programmation
    - C'est plutôt un langage de description
    - Composé de plusieurs balises (tags)
  - Interprété par le navigateur
  - Il est créé en 1991 par *Tim Berners-Lee* et il est standardisé depuis 1994 par W3C



**HTML**



**JavaScript**



**CSS**

# Standardisation

- **W3C** (World Wide Web Consortium)

- organisme de standardisation fondé par Tim Berners-Lee
- chargé de promouvoir la compatibilité des technologies web (HTML, XML, CSS, SOAP...)

- **Historique**

- Les langages de balisage sont issus du langage **SGML** (Standard Generalized Markup Language) créé en 1986 pour structurer des contenus très divers
- En 1992, le CERN (Centre Européen de Recherche Nucléaire) rend public le projet World Wide Web définissant un langage de présentation et de structuration de documents hypertextes, dérivé de SGML

## **HTML = HyperText Markup Language**

- En 1998, le langage XML (eXtensible Markup Language) a vu le jour
- Son succès dans de multiples domaines d'application a conduit le W3C (World Wide Web Consortium) à créer le langage **XHTML** (eXtensible HyperText Markup Language) en 2000, non plus comme une nouvelle version de HTML, mais comme une reformulation de HTML en tant qu'une application XML.

# HTML : évolution

- **HTML1** : première version créée par Tim Berners-Lee en 1991.
- **HTML2** : deuxième version, apparue en 1994. On commence à parler de W3C.
- **HTML3** : apparue en 1996 avec plusieurs nouveautés comme les tableaux, les scripts, le positionnement du texte autour des images, etc.
- **HTML4** : apparue en 1998 avec la possibilité :
  - d'utiliser de frames (découpage d'une page en plusieurs parties),
  - des améliorations sur les formulaires,
  - d'utiliser des feuilles de style (CSS).
- **HTML5** : finalisée en octobre 2014 et permet de :
  - simplifier les déclarations et intégrer de nouveaux médias
  - ajouter plus de précisions sur les champs d'un formulaire
  - améliorer la structuration du document (intégration de balises sémantiques)

# Quel IDE pour HTML?

- **Un éditeur HTML** (ou éditeur Web) : logiciel conçu pour faciliter la préparation et la modification de documents écrits en HTML. Il existe deux catégories d'éditeurs :
  - Les éditeurs visuels : Visual Studio Code, GoLive, DreamWeaver, frontpage, Komodo ...
  - Les éditeurs textuels ou classiques : Notepad++, Sublime Text, Atom, Bluefish, HTMLEdit ...
- **Visual Studio Code:** <https://code.visualstudio.com/download>
  - Gratuit
  - Pouvant s'adapter selon le langage de programmation
  - Extensible via l'installation de quelques centaines d'extensions
- **NB :** Pour activer la sauvegarde automatique : aller dans **File > Auto save**




# Création d'un projet

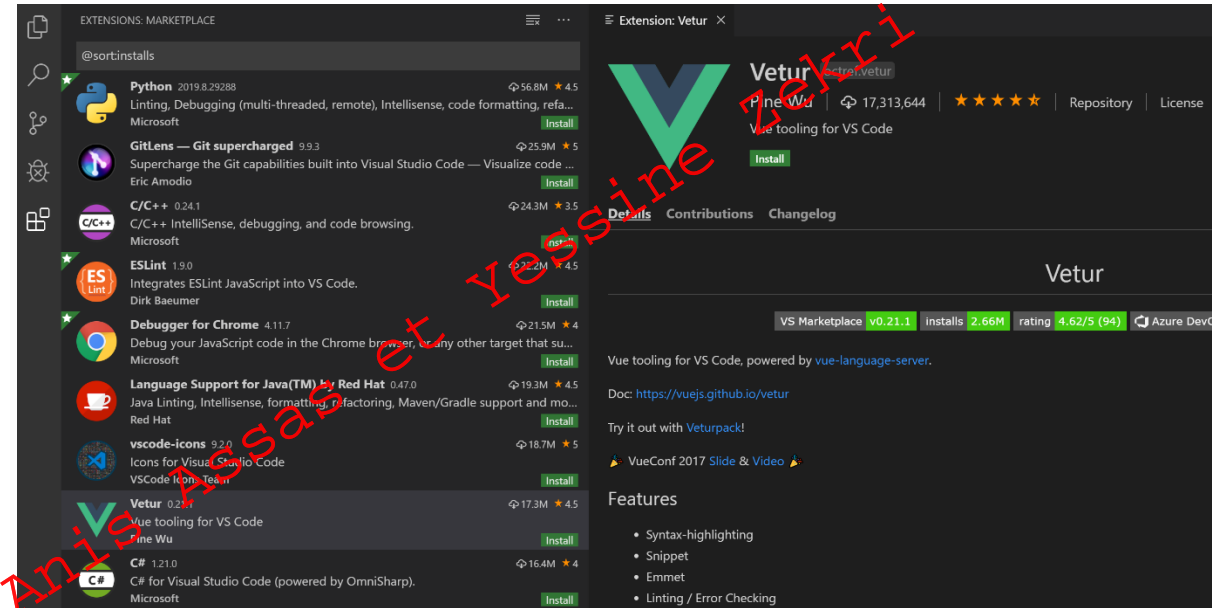
- Pour créer un projet sous **VSC**
  - Allez dans **Fichier > Ouvrir le dossier...**
  - Cliquez sur **Nouveau dossier** et saisissez **formation-html**
  - Cliquez sur le dossier **formation-html** puis sur le dossier **Sélectionner un dossier**
  - Créez un fichier **index.html** dans **formation-html**
  - Dans **index.html**, saisissez **html:5** puis cliquez sur **Entrée**

- **Code généré :**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
</body>
</html>
```

# Extension Live Server

- Installez l'extension **Live Server**
  - Cliquer sur l'icone  ou **Ctrl+Shift+X**
  - Tapez par la suite **Live Server**
  - Cliquez sur le bouton **Install**
- Faites un clic droit sur **index.html**
- Cliquez sur **Open with Live Server**
- Vous pouvez aussi utiliser l'icone **Go Live**



# Concepts de balise : Syntaxe

- Il existe deux types de balises.
  - **balise vide** : balise sans contenu. Ses informations apparaissent dans des attributs. Chaque attribut définit une propriété de la balise.
    - **Syntaxe** : `<nom-balise attribut1 = "valeur1" attribut2="valeur2" ... />`  
`<br/>`  
``
    - **balise non vide** : balise avec contenu, qui apparaît entre `<nom-balise>` et `</nom-balise>`
      - **Syntaxe** : `<nom-balise attribut1 = "valeur1" attribut2 = "valeur2" ... > Contenu </nom-balise>`  
`<h1 align = "center"> Le langage HTML</h1>`  
`<p>Ceci est un paragraphe</p>`
  - **Commentaire : balise particulière** `<!-- ceci est un commentaire -->`

# Concepts de balise : Propriétés

- Les balises doivent être ouvertes puis fermées récursivement, comme des parenthèses ([...]{(...)})
- Tout ce qui est entouré par deux balises s'appelle : **élément**
- Un élément peut éventuellement contenir du texte, liens, balises...
- Un texte en clair (non-entouré par une balise) est affiché
- Les commentaires en HTML sont ignorés par le navigateur
- Les commentaires ne sont pas affichés mais restent visibles dans le code source de la page (NB : ne jamais contenir une information confidentielle).

# Structure d'une page HTML : Composition

- Le contenu d'une page HTML est compris entre deux balises **html**
- Une page HTML est composée de deux parties :
  - une entête : **head**
  - un corps : **body**
- **DOCTYPE**
  - Ce n'est pas une balise
  - C'est facultatif
  - C'est une directive permettant de préciser qu'il s'agit d'un document HTML et indiquant sa version

```
<!DOCTYPE html>
<html>
  <head>

  </head>
  <body>

  </body>
</html>
```

```
<!-- HTML4.01 transitional -->
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

```
<!-- HTML5 -->
```

```
<!DOCTYPE html>
```

# HEAD

- Que peut contenir **<HEAD>** ?

```
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Cours HTML 5</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
  <script src="script.js" type="text/javascript"></script>
</head>
```

- **<title>** titre du document (affiché par le navigateur en haut de la page)
- **<link>** pour référencer un fichier (CSS par exemple)
- **<style>** pour inclure du code CSS
- **<meta>** peut contenir :
  - des informations sur le codage
  - des informations pour les navigateurs et les moteurs de recherche

# BODY

- Que peut contenir **<BODY>** ?

**<body>**

Les informations qui seront affichées dans le navigateur

**</body>**

- Dans le <Body>

- texte
- tableau
- image/vidéo/document
- menu
- lien
- formulaire
- liste
- ...

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Cours HTML 5</title>
</head>
<body>
  <h1 align="center">Le langage HTML</h1>
  
</body>
</html>
```



# Les attributs d'une balise

- Les balises, dans certains cas, ne suffisent pas.
- On leur associe donc des attributs

- Balise avec un seul attribut
- Balise avec plusieurs attributs
- Balise avec attribut ne prennent pas de valeur

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

```
<input type="text" placeholder="saisir votre nom">
```

```
<input type="text" placeholder="saisir votre nom" readonly>
```

## Remarque :

Avec HTML5, la valeur d'un attribut peut être entourée par des guillemets, des apostrophes comme elle peut ne pas être entourée si elle ne contient pas d'espace.

# Quelques attributs standards

- **class** : nom de classe pour cibler plusieurs éléments ( à voir dans la partie CSS)

```
<p class="paragraphe">Paragraphe 1</p>
```

- **id** : identifiant unique dans la page pour cibler un seul élément ( à voir dans la partie CSS)

```
<p class="paragraphe" id="para1">Paragraphe 1</p>
```

```
<p class="paragraphe" id="para2">Paragraphe 2</p>
```

- **style** : style CSS de l'élément

```
<p style="color: blue;">Paragraphe avec du texte en bleu.</p>
```

- **width** et **height** peuvent être utilisés comme attributs pour les balises suivantes :

- img , canvas, iframe, object, Video

# Quelques attributs standards

- Exemples avec width et height :

```
<canvas width="350" height="350" style="background-color: yellow;"></canvas>
```

```
<object data="fichier.pdf" type="application/pdf" width="350" height="350"></object>
```

```
<video width="350" height="350" controls>  
  <source src="video.mp4" type="video/mp4">  
</video>
```

cours réalisé par Anis Assas et Yessine Zekri

# Organisation du texte : Les paragraphes

`<p>Un paragraphe</p>`

`<i>Un texte en italic</i>`

`<b>un texte en gras</b>`

`<em>Un texte un peu en valeur</em>`

`<strong>Un texte bien en valeur</strong>`

`<small>Un texte à une taille inférieure</small>`

`<del>Un texte barré</del>`

`<ins>Un texte souligné</ins>`

`<sub>Un texte en indice</sub>`

`<sup>Un texte en exposant</sup>`

`<mark>Un texte marqué</mark>`

`<i>Italic</i>`

*Italic*

`<b>Bold</b>`

**Bold**

`<em>Emphasized</em>`

*Emphasized*

`<strong>Strong</strong>`

**Strong**

`<small>small</small>`

small

`<del>Deleted</del>`

~~Deleted~~

`<ins>Inserted</ins>`

Inserted

`v<sub>f</sub>`

$v_f$

`a<sup>2</sup>`

$a^2$

`<mark>Marked</mark>`

**Marked**

# Autres balises de formatage

- pour souligner un texte

```
<u>Un texte souligné</u>
```

- pour afficher une ligne horizontale

```
<hr>
```

- abréviation

```
<abbr title="World Wide Web">WWW</abbr>
```

- adresse formatée (en italique)

```
<address><p>Adresse : 123, Rue de l'Exemple</p></address>
```

- citation (en italique)

```
<p>Mon livre préféré est <cite>Les misérables</cite> de Vitor Hugo</p>
```

- élément de code informatique

```
<code> select * from client;</code>
```

- <pre> : texte pré-formaté

```
<pre>function Hello(name){ return "Bonjour " + name + " !";}</pre>
```

# Exemples

- Avec **del** et **ins**

```
<p>La dernière version de HTML est <del>quatre</del> <ins>cinq</ins></p>
```

- Avec **mark**

```
<p>HTML est très <mark> facile </mark>!.</p>
```

- Avec **address**, **strong**, **br** et **abbr**

```
<address>  
  <strong>Twitter, Inc.</strong><br>  
  1355 Market Street, Suite 900<br>  
  San Francisco, CA 94103<br>  
  <abbr title="Phone">Ph:</abbr>(123)456-7890<br>  
</address>
```

- Avec **pre**

```
<pre>  
  Le texte dans une balise "pre" est affiché.  
  Les sauts de lignes et les espaces sont respectés.  
</pre>
```

# Les titres : les différents niveaux

- De plus grand au plus petit

1. `<h1>...</h1>`
2. `<h2>...</h2>`
3. `<h3>...</h3>`
4. `<h4>...</h4>`
5. `<h5>...</h5>`
6. `<h6>...</h6>`

```
<h1>Titre de niveau 1</h1>
<h2>Titre de niveau 2</h2>
<h3>Titre de niveau 3</h3>
<h4>Titre de niveau 4</h4>
<h5>Titre de niveau 5</h5>
<h6>Titre de niveau 6</h6>
```

Titre de niveau 1

Titre de niveau 2

Titre de niveau 3

Titre de niveau 4

Titre de niveau 5

Titre de niveau 6

# Les règles à respecter

- Avoir **une seule balise h1** par page
- Rédiger des **h1 différents** pour chaque page : Chaque page devrait avoir son propre titre principal (balise <h1>) qui est spécifique au contenu de cette page. Le titre principal doit être unique pour chaque page.
- Utiliser les **niveaux inférieurs** (du **h2 au h6**) pour structurer le contenu d'une page.
- Les titres ne doivent pas être utilisés uniquement pour **définir** la taille de la police d'un texte : Leur objectif principal est de structurer le contenu de la page et de faciliter la navigation pour les utilisateurs et les moteurs de recherche.
- Les niveaux de titres peuvent être **imbriqués** afin de créer des sous-sections qui reflètent l'organisation. Leur objectif principal est de structurer le contenu de la page et de faciliter la navigation pour les utilisateurs et les moteurs de recherche.



# Les listes : les différents niveaux

- `<ul>...</ul>` : une liste non-ordonnée (non-numérotée)
- `<ol>...</ol>` : une liste ordonnée
- `<li>...</li>` : un élément d'une liste
- `<dl>...</dl>` : une liste de description
- `<dt>...</dt>` : un terme d'une liste
- `<dd>...</dd>` : la description d'un terme

# Exemples de listes

- Liste non-numérotée

```
<ul>  
  <li>Barcelone</li>  
  <li>Marseille</li>  
  <li>Manchester</li>  
</ul>
```

- Barcelone
- Marseille
- Manchester

- Liste numérotée

```
<ol>  
  <li>Barcelone</li>  
  <li>Marseille</li>  
  <li>Manchester</li>  
</ol>
```

1. Barcelone
2. Marseille
3. Manchester

cours réalisé par Anis Assas et Yessine Zekri

# Exemples : Listes numérotées

- Si on veut remplacer les chiffres par des lettres

```
<ol type="A">  
  <li>Barcelone</li>  
  <li>Marseille</li>  
  <li>Manchester</li>  
</ol>
```

A. Barcelone  
B. Marseille  
C. Manchester

- Pour commencer à partir d'un nombre autre que 1

```
<ol start="5">  
  <li>Barcelone</li>  
  <li>Marseille</li>  
  <li>Manchester</li>  
</ol>
```

5. Barcelone  
6. Marseille  
7. Manchester

cours réalisé par Anis Assas et Yessine Zekri

# Exemples

- Il est possible d'imbriquer les listes

```
<ul>
  <li>France</li>
  <li>Italie
    <ul>
      <li>Milan</li>
      <li>Turin</li>
    </ul>
  </li>
</ul>
```

- France
- Italie
  - Milan
  - Turin

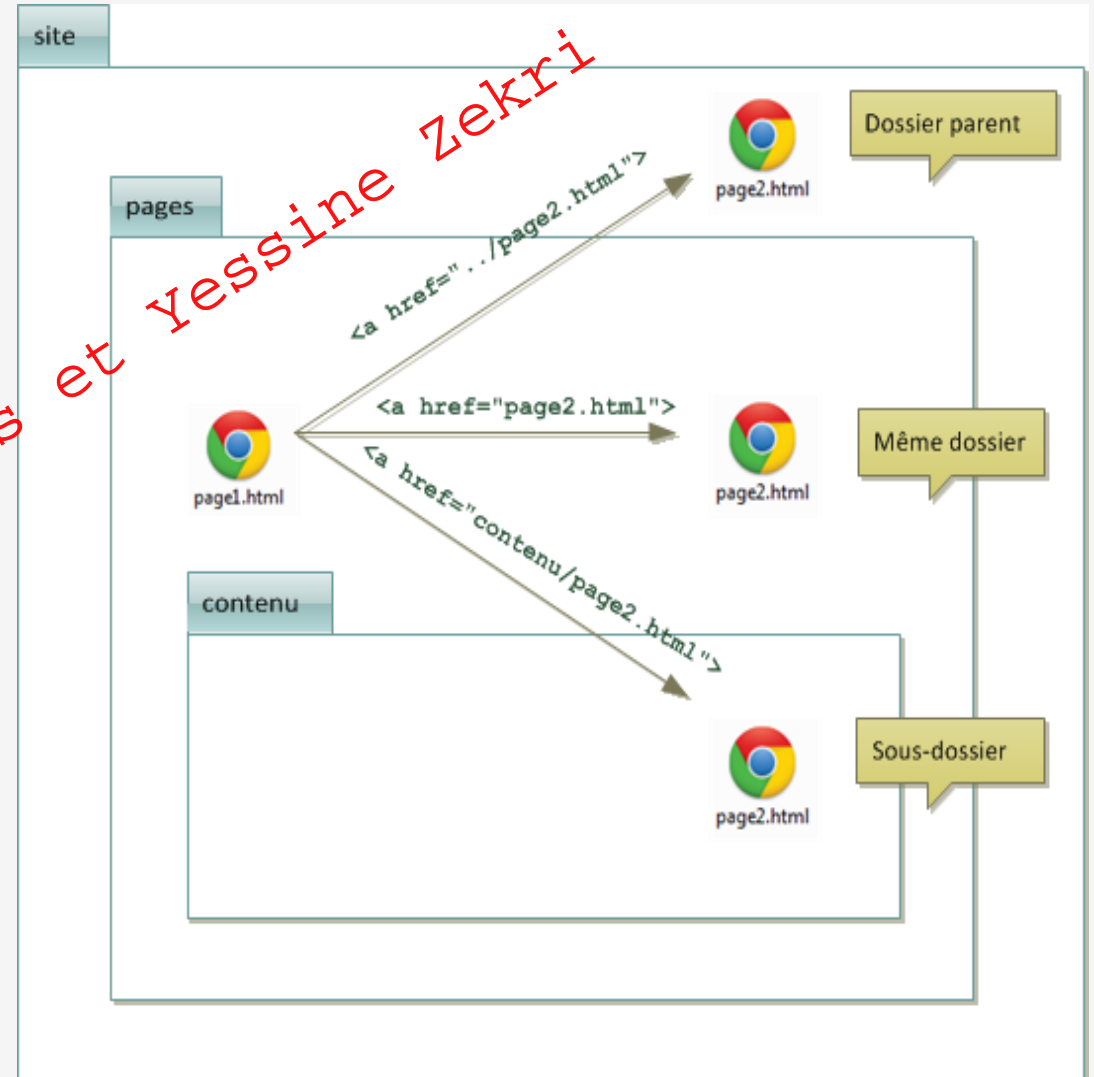
- Exemple d'une liste de description

```
<dl>
  <dt>France</dt>
  <dd>Bleu blanc rouge</dd>
  <dt>Allemagne</dt>
  <dd>Noir jaune rouge</dd>
</dl>
```

France  
Bleu blanc rouge  
Allemagne  
Noir jaune rouge

# Les liens

- `<a>...</a>` : insérer un lien (interne ou externe)
- Pour créer un lien vers un autre document local ou distant
  - `<a href="http://monsite.fr">monsite</a>`
  - `<a href="page2.html">Page 2</a>`
- **href** est l'attribut de l'élément `<a>` le plus important puisqu'il indique la destination du lien.
- Utiliser un ancre pour pointer vers un signet ('name' ou 'id') dans un même document.



# Exemples

- Lien pour télécharger un fichier

```
<a href="https://www.7-zip.org/a/7z1900-x64.exe">Download 7-zip</a>
```

- Le lien sera ouvert dans une nouvelle fenêtre

```
<a href="https://www.unicef.org/fr" target="_blank">UNICEF</a>
```

- Pour définir une base pour tous les liens de la page pour simplifier la création de liens

```
<base href="https://www.unicef.org/fr/">  
<a href="agir">Agir & Donner</a>
```

- Créer une ancre dans une même page vers l'id d'un élément

```
<a href="#ancree">Ancree</a>  
<p id="ancree">Les ancres se définissent grâce à l'attribut NAME ou ID</p>
```

# Les tableaux

- Les tables constituent un mode privilégié de présentation d'information structurée.
- Une table (balise **<table>**) est divisée en lignes et colonnes
- Une bordure délimite ou non les cellules de la table (Balise **<td>**), organisée en lignes (balise **<tr>**)
- La table préserve en permanence l'aspect visuel de colonnes dont toutes les cellules ont la même largeur.
- En revanche, les lignes peuvent avoir des hauteurs différentes
- Les différentes balises d'un tableau :
  - **<table>** : tableau
  - **<caption>** : légende du tableau (texte associé)
  - **<th>** : cellule d'en-tête dans un tableau
  - **<tr>** : ligne dans un tableau
  - **<td>** : cellule dans un tableau

# Exemple : tableau avec une bordure

- Tableau avec une bordure

```
<table border="1">  
  <caption>Moyenne par matière</caption>  
  <tr>  
    <th>Matiere</th>  
    <th>Moyenne</th>  
  </tr>  
  <tr>  
    <td>PHP</td>  
    <td>10</td>  
  </tr>  
  <tr>  
    <td>Java</td>  
    <td>8</td>  
  </tr>  
</table>
```

Moyenne par  
matière

Matiere	Moyenne
PHP	10
Java	8



# Exemple : Fusion des cellules

- Fusionner des cellules d'un tableau

```
<table border="2">
  <tr>
    <td rowspan="2">Sites</td>
    <td colspan="2">Performances</td>
  </tr>
  <tr>
    <td>Nombre visiteurs/jour </td>
    <td>PageRank</td>
  </tr>
  <tr>
    <td>www.monsite.com</td>
    <td>3000</td>
    <td>7/10</td>
  </tr>
</table>
```

- rowspan** : fusion verticale de 2 ou plusieurs lignes
- colspan** : fusion horizontale de 2 ou plusieurs colonnes

Sites	Performances	
	Nombre visiteurs/jour	PageRank
www.monsite.com	3000	7/10

# Exemple : Regroupement des éléments

- Regrouper les éléments d'un tableau

Mois	Loyer
January	600\$
February	700\$
Total	1 300\$

```
<table border="1">
  <thead>
    <tr>
      <th>Mois</th>
      <th>Loyer</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>January</td>
      <td>600$</td>
    </tr>
    <tr>
      <td>February</td>
      <td>700$</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td>Total</td>
      <td>1 300$</td>
    </tr>
  </tfoot>
</table>
```

# Les formulaires : Déclaration

- Déclaration d'un formulaire

```
<form method="POST ou GET" action="page web destination"></form>
```

- Les attributs d'un formulaire

- **method** : concerne l'envoi de données et peut prendre deux valeurs :

- ❖ **GET** : (utilisé par défaut) non fréquemment utilisée car limitée à 255 caractères.

En plus, les informations envoyées seront visibles dans la zone d'adresse (URL).

- ❖ **POST** : plus utilisée que GET car elle permet d'envoyer un grand nombre d'informations et les données saisies dans le formulaire ne transitent pas par la barre d'adresse (URL).

# Les formulaires : Déclaration

- Les attributs d'un formulaire

- **action** : indique l'adresse (absolue ou relative) de la page ou du programme qui va traiter les informations du formulaire (généralement avec un langage autre que HTML).

```
<form action="traitement.php"></form>  
<form action="http://www.monsite.com/traitement.php"></form>  
<form action="mailto:yessine.zekri.isetjb@gmail.com"></form>
```

- **name** : indique le nom interne du formulaire
- **onsubmit** : gère un événement quand l'utilisateur clique sur le bouton d'envoi des données (pour valider les données par exemple).
- **onreset** : gère un événement quand l'utilisateur efface en bloc toutes les saisies réalisées (pour annuler la saisie des données par exemple).

- Que peut-on avoir dans un formulaire?

- Des zones de saisie, des zones de choix, des boutons...

# Zones de saisie

- Une zone de saisie mono-ligne

```
<input type="text" name="nom-input">
```

- Une zone de saisie multi-lignes

```
<textarea name="nom-textarea" rows="5"></textarea>
```

- Un libellé associé à une zone de saisie. Généralement on associe la même valeur aux attributs **for**, **id** et **name**

```
<label for="nom">Nom :</label>  
<input type="text" name="nom" id="nom" value="John Wick">
```

- Avec HTML 5, les 3 écritures suivantes sont équivalentes :

```
<input type=text value=John>  
<input type="text" value="John Wick">  
<input type='text' value='John Wick'>
```

- **email** : pour les adresses emails

```
<input type="email" name="user_email">
```

# Zones de saisie

- **password** : pour les mots de passe

```
<input type="password" id="pass" name="password">
```

- **color** : pour les couleurs

```
<input type="color" id="head" name="couleur" value="#00ff00">
```

- **number** : pour les nombres (attributs min, max et step)

```
<input type="number" name="age" min="10" max="100" step="5" >
```

- **date** : pour les calendriers (on pourrait également utiliser le type : month, week, time, datetime-local, ...)

```
<input type="date" id="date" value="2023-07-30">  
<input type="month" id="month" value="2023-12">  
<input type="week" id="week" value="2023-W01">  
<input type="time" id="time" value="12:30">  
<input type="datetime-local" id="time" value="2023-07-30T12:30">
```

- **url** : pour les adresses URL

```
<input type="url" name="url" placeholder="https://example.com" size="30">
```

# Zones de saisie

- **pattern** : attribut pour l'utilisation des expressions régulières (regex) pour le filtrage (validité de données)

```
<input type="text" pattern="[a-z]{3,5}"/>
<style>
    input:invalid {
        border: red solid 3px;
    }
</style>
```

Texte :

Texte :

Le champ de saisie doit contenir entre 3 et 5 lettres minuscules

- **Autres exemples :**

```
<input type="text" pattern="[A-Z][0-9]{3,}"/>
<style>
    input:invalid {
        border: red solid 3px;
    }
</style>
```

le champ de saisie doit commencer par une lettre majuscule suivie d'au moins 3 chiffres consécutifs.

# Zones de saisie

- Autres exemples :

```
<input type="text" pattern="[a-zA-Z]+[0-9]{4}[A-Z]?" />
<style>
    input:invalid {
        border: red solid 3px;
    }
</style>
```

Le champ de saisie doit commencer par une ou plusieurs lettres (minuscules ou majuscules), suivies de 4 chiffres et éventuellement d'une lettre majuscule à la fin.

```
<input type="email"
pattern="[a-zA-Z0-9.-_]+@[a-zA-Z]+\.[a-zA-Z]{2,3}" />
<style>
    input:invalid {
        border: red solid 3px;
    }
</style>
```

Le champ de saisie doit contenir une adresse e-mail valide, avec un nom d'utilisateur pouvant inclure des lettres, des chiffres, des points (.), des tirets (-) et des traits de soulignement (\_), suivi du symbole "@" et d'un nom de domaine contenant des lettres, suivi d'un point (.) et d'une extension de domaine de 2 à 3 lettres.



# Zones de saisie

- Autres exemples :

```
<input type="url" pattern="http://www\..+\..+"/>
<style>
    input:invalid {
        border: red solid 3px;
    }
</style>
```

Le champ de saisie doit contenir une URL commençant par "<http://www>", suivie de n'importe quel contenu avant un autre point (.) et une extension de domaine.

(? ⇔ {,1}) 0 ou 1 répétition)  
(\* ⇔ {0,}) 0 ou plusieurs répétitions)  
(+ ⇔ {1,}) 1 ou plusieurs répétitions)  
(. ⇔ absolument n'importe quel caractère)

# Zones de choix

- Trois types de zones de choix : Les cases à cocher, Les boutons radio, Les listes déroulantes

- Cases à cocher

```
<input type="checkbox" name="nom_du_choix" checked>  
<input type="checkbox" name="nom_du_choix">
```

- Boutons radios

```
<input type="radio" name="nom_du_choix" checked>  
<input type="radio" name="nom_du_choix">
```

- Listes déroulantes

```
<select name="liste">  
  <option value="valeur1" selected>valeur1</option>  
  <option value="valeur2">valeur2</option>  
  <option value="valeur3">valeur3</option>  
</select>
```

# Exemple : cases à cocher

- Case à cocher dans un fieldset (délimiter les groupes dans un formulaire)

```
<fieldset>
  <legend> Veuillez sélectionner vos intérêts :</legend>
  <input type="checkbox" id="coding" name="interest" value="coding">
  <label for="coding">Développement</label>
  <input type="checkbox" id="music" name="interest" value="music">
  <label for="music">Musique</label>
</fieldset>
```

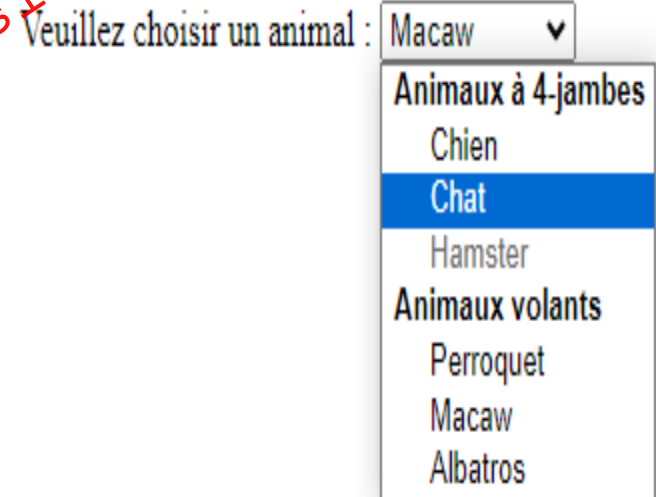
Veuillez sélectionner vos intérêts :

☐ Développement ☐ Musique

# Exemple : Liste déroulante

- Les listes déroulantes (le regroupement)

```
<label> Veuillez choisir un animal :  
  <select name="pets">  
    <optgroup label="Animaux à 4-jambes">  
      <option value="Chien">Chien</option>  
      <option value="chat">Chat</option>  
      <option value="hamster" disabled>Hamster</option>  
    </optgroup>  
    <optgroup label="Animaux volants">  
      <option value="perroquet">Perroquet</option>  
      <option value="macaw">Macaw</option>  
      <option value="albatros">Albatros</option>  
    </optgroup>  
  </select>  
</label>
```



# Exemple : Liste de données

- Les listes de données (datalist) = liste déroulante + zone de saisie

```
<label>Veuillez choisir votre sport préféré  
:</label>  
<input list="sports" name="sport">  
<datalist id="sports">  
  <option value="football">  
  <option value="handball">  
  <option value="tennis">  
  <option value="hockey">  
</datalist>
```

Veuillez choisir votre sport préféré :

- football
- handball
- tennis
- hockey

cours réalisé par Anis Assas et Yessine Zekri

# Boutons (1)

- Types de boutons

- button** : bouton générique qui permet de réaliser plusieurs tâches pour HTML (quitter une page,...) ou de déclencher un code **JavaScript**

- Pour déclarer un bouton simple (pas de type submit) dans une balise **form**

```
<input type="button" value="Cliquer">
```

Ou

```
<button type=button>Cliquer</button>
```

- Exemple

```
<form>
  <input type="button" value="Cliquer" onclick="alert('Vous avez cliqué.');">
  <button type=button onclick="alert('Vous avez cliqué.');">Cliquer</button>
</form>
```

# Boutons (2)

- Types de boutons

- submit** : envoie le contenu d'un formulaire à la page indiquée dans l'action sachant que tout bouton déclaré dans un formulaire avec la balise **<button>** est de type **submit** et un bouton de type **submit** peut être aussi déclaré avec la balise **input**

```
<input type="submit" value="Envoyer">
```

Ou

```
<button>Envoyer</button>
```

- Exemple

```
<form method="post" action="page_web_destination.html">  
  <input type="submit" value="Envoyer">  
  <button>Envoyer</button>  
</form>
```

# Boutons (3)

- Types de boutons

- reset** : remet à zéro (efface) le contenu d'un formulaire

```
<input type="reset" value="Annuler">
```

- Exemple**

```
<form method="post" action="page_web_destination.html">  
  <input type="text">  
  <input type="reset" value="Annuler">  
</form>
```

cours réalisé par anis assas et yessine zekri



# Autres attributs pour les formulaires (1)

- Quelques autres attributs :

- **required** : pour indiquer qu'un champ est obligatoire

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

- **Exemple**

```
<form method="post" action="page_web_destination.html">  
  <input type="text" required>  
  <input type="submit" value="Envoyer">  
</form>
```

cours réalisé par Anis ASSAS et Yessine Zekri

# Autres attributs pour les formulaires (2)

- Quelques autres attributs :

- **autofocus** : pour placer le curseur dans cet élément dès le chargement de la page

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

- Exemple

```
<form method="post" action="page_web_destination.html">  
  <input type="text" autofocus>  
  <input type="submit" value="Envoyer">  
</form>
```

cours réalisé par Anis ASSAS et Yessine Zekri

# Autres attributs pour les formulaires (3)

- Quelques autres attributs :

- **autocomplete** : pour indiquer si on autorise auto-complétion (**on** ou **off**)

```
<input type="text" autocomplete="on">
```

- Exemple

```
<form method="GET" action="page_web_destination.html">  
  <input type="text" autocomplete="on" name="nom">  
  <input type="submit" value="Envoyer">  
</form>
```

cours réalisé par Anis Assas et Yessine Zekri

# Autres attributs pour les formulaires (4)

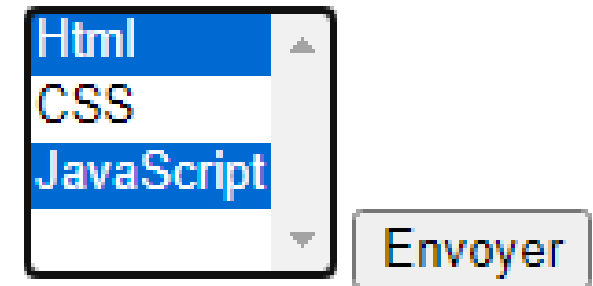
- Quelques autres attributs :

- **multiple** : pour indiquer qu'il est possible de choisir plusieurs éléments d'une liste

```
<select name="option" multiple></select>
```

- Exemple

```
<form method="GET" action="page_web_destination.html">  
  <select name="module" multiple>  
    <option value="Html">Html</option>  
    <option value="CSS">CSS</option>  
    <option value="JavaScript">JavaScript</option>  
  </select>  
  <input type="submit" value="Envoyer">  
</form>
```



# Autres attributs pour les formulaires (5)

- Quelques autres attributs :

- **placeholder** : pour afficher un message indicatif dans un champ

```
<input type="text" placeholder="Mon prénom" name="prenom">
```

- **Exemple**

```
<form method="GET" action="page_web_destination.html">  
  <input type="text" placeholder="Mon prénom" name="prenom">  
  <input type="submit" value="Envoyer">  
</form>
```

cours réalisé par Anis Assas et Yessine Zekri

# Autres attributs pour les formulaires (6)

- Quelques autres attributs :

- **readonly** : pour rendre le champ en lecture seule

```
<input type="text" value="Mon prénom" name="prenom" readonly>
```

- **Exemple**

```
<form method="GET" action="page_web_destination.html">  
  <input type="text" value="Mon prénom" name="prenom" readonly>  
  <input type="submit" value="Envoyer">  
</form>
```

cours réalisé par Anis Assas et Yessine Zekri

# Autres attributs pour les formulaires (7)

- Quelques autres attributs :

- **range** : retourne une valeur numérique qui correspond à la position du curseur.

```
<label>Volume : </label>  
<input type="range" id="r" value="80" name="volume"/>
```

- Exemple

```
<form method="GET" action="page_web_destination.html">  
  <label>Volume : </label>  
  <input type="range" id="r" value="80" name="volume"/>  
  <input type="submit" value="Envoyer">  
</form>
```

# La balise <meter>

- **Balise <meter>** : représente une jauge mesurant une valeur comprise entre un min et un max.

```
<label> Progression : </label>  
<meter min="10" max="100" value="40" name="progression"></meter>
```

- **Exemple**

```
<form method="GET" action="page_web_destination.html">  
  <label> Progression : </label>  
  <meter min="10" max="100" value="40" name="progression"></meter>  
  <input type="hidden" name="progression" value="40">  
  <input type="submit" value="Envoyer">  
</form>
```



# La balise <progress>

- **Balise <progress>** : représente un champ qui indique la progression en pourcentage d'une tâche au cours du temps.

```
<label> Installation : </label>  
<progress id="p" min="10" max="100" value="40" name="progression"></progress>
```

- **Exemple**

```
<form method="GET" action="page_web_destination.html">  
  <label> Installation : </label>  
  <progress id="p" min="10" max="100" value="40" name="progression"></progress>  
  <input type="hidden" name="progression" value="40">  
  <input type="submit" value="Envoyer">  
</form>
```

# Les balises multimédia : Les images

- La balise `<img>` permet d'insérer soit

- une image locale :

```

```

- ou distante :

```

```

- Les attributs recommandés pour garantir un affichage correct

- **src** : URL de l'image
- **alt** : Texte à afficher si l'image ne peut pas être chargée (indispensable pour être **W3C Valid**)
- **height** : hauteur
- **width** : largeur

# Exemples

- Une image locale d'hauteur de 128 pixels \* largeur de 128 pixels

```

```

- Une image à distance d'hauteur de 132 pixels \* largeur de 100 pixels

```

```

- Pour ajouter une légende à une image, on ajoute les deux balises **figure** et **figcaption**

```
<figure>
  
  <figcaption>superbus.com</figcaption>
</figure>
```

# Les balises multimédia : audio

- **Attention** : les navigateurs ne supportent pas tous les formats **audio** et **vidéo** existants

- Insérer un élément audio

```
<audio src="/audio.mp3" controls></audio>
```

```
<audio controls>  
  <source src="audio.mp3">  
</audio>
```

- Les attributs possibles :

- **controls** : pour afficher les boutons lecture et pause et la barre de défilement
- **loop** : pour jouer le fichier audio en boucle
- **autoplay** : pour lire le contenu du fichier dès le chargement de la page
- **muted** : pour activer le mode silencieux

```
<audio controls loop autoplay muted>  
  <source src="audio.mp3" type="audio/mpeg">  
</audio>
```

# Les balises multimédia : Vidéo

- Insérer un élément Vidéo

```
<video src="video.mp4" controls height="500" width="650"></video>
```

```
<video controls height="500" width="650">  
  <source src="video.mp4">  
  <source src="video.webm">  
</video>
```

- Remarque : Les balises <audio> et <video> pourraient inclure plusieurs formats de sources pour que le navigateur puisse choisir le format qu'il supporte,
- Les attributs possibles :
  - **controls, loop, autoplay et muted** ont le même rôle que ceux de l'élément <audio>
  - **L'attribut poster** : précise l'URL de l'image à afficher à la place de la vidéo tant que celle-ci n'est pas lancée.

```
<video controls poster = "image.png" height="500" width="650">  
  <source src="video.mp4">  
</video>
```

assas\_anis@yahoo.fr & yessine.zekri.isetjb@gmail.com

# Les balises multimédia : « iframe »

- Une deuxième solution consiste à
  - héberger la vidéo sur **YouTube**
  - copier l'identifiant de la vidéo
  - utiliser ce dernier dans la balise **iframe**

```
<iframe height="500" width="650" src="https://www.youtube.com/embed/8cm1x4bC610"></iframe>
```

cours réalisé par Anis Assas et Yessine Zekri

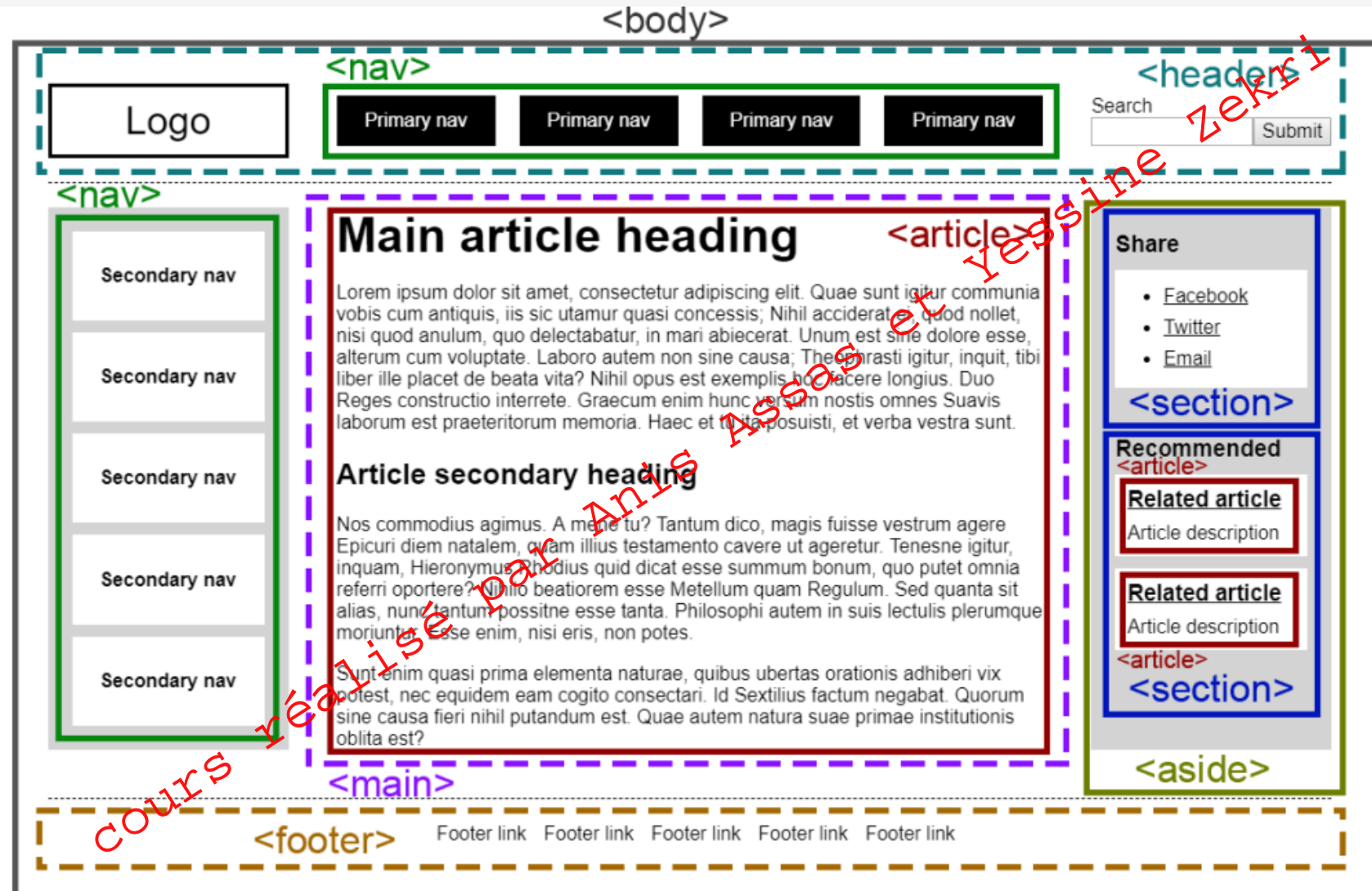
# Les balises structurales

**Balises sémantiques** : Simplifient le codage et la lecture d'un site par les moteurs de recherche, permettant aux robots de repérer plus facilement les éléments du site.

- **<div>...</div>** : conteneur d'éléments
- **<header>...</header>** : l'entête de la page
- **<footer>...</footer>** : le pied de la page
- **<main>...</main>** : tout le reste de la page
- **<nav>...</nav>** : l'emplacement du menu
- **<section>...</section>** : le main peut être composé de plusieurs sections
- **<article>...</article>** : dans une section on peut définir un ou plusieurs articles
- **<aside>...</aside>** : Lorsque la balise **<aside>** est placée dans un **<article>**, son contenu se rapporte spécifiquement à cet article et non au site Web dans son ensemble. Cela peut inclure des notes de bas de page, un glossaire ou tout élément lié à l'article lui-même.

- **Remarque** : Ne pas confondre les deux balises **<head>** et **<header>**

# Les balises structurelles





# Balises de compléments d'information

- **<details> et <summary>** : Éléments qui balisent un contenu masqué par défaut et déployé à la demande de l'utilisateur
  - **<details>...</details>** : permet de révéler une information : l'élément de divulgation des détails
  - **<summary>...</summary>** : permet de révéler le contenu d'un résumé ou d'une légende pour le contenu d'un élément <details>. En cliquant sur l'élément <summary>, on passe de l'état affiché à l'état masqué (et vice versa) de l'élément <details> parent.

```
<details>
  <summary> Cliquer pour voir les détails ! </summary>
  <dl>
    <dt>CSS</dt> <dd>Définition CSS ...</dd>
    <dt>HTML</dt> <dd>Définition HTML ...</dd>
    <dt>JavaScript</dt> <dd>Définition JavaScript ...</dd>
  </dl>
</details>
```

▶ Cliquer pour voir les détails !

▼ Cliquer pour voir les détails !

CSS  
Définition CSS ...  
HTML  
Définition HTML ...  
JavaScript  
Définition JavaScript ...

# Le dessin vectoriel SVG

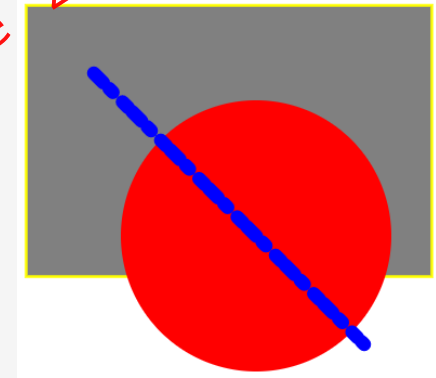
- **SVG (Scalable Vector Graphics)** : format d'images vectorielles basé sur le langage de balisage XML.
  - ❑ répond à des besoins graphiques légers qu'ils soient statiques, dynamiques ou interactifs.
  - ❑ seules les informations décrivant ces formes sont stockées (coordonnées, couleurs, effets) contrairement aux images bitmap (JPG, PNG, GIF) qui doivent mémoriser le contenu pixel par pixel.
- **La balise <svg>...</svg>** : nouvel élément introduit par HTML5 afin d'embarquer du contenu SVG dans une page web.
- **Les formes** : SVG propose plusieurs moyens de créer des formes, simples ou plus complexes.
  - ❑ **Rectangle** : **<rect>** en spécifiant sa longueur et sa largeur avec les attributs **width** et **height**. Les attributs **x** et **y** indiquent la position (horizontale et verticale).

```
<svg width="500" height="500">  
  <rect x="25px" y="25px" width="150" height="100" fill="grey" stroke="yellow"/>  
</svg>
```



# Le dessin vectoriel SVG

- ❑ **Cerle ou ellipse** : **<circle>** L'attribut **r** spécifie le rayon et les attributs **cx** et **cy** les coordonnées du centre.
- ❑ **Ligne** : **<line>** nécessitant les quatre paramètres :
  - **x1** : spécifie la position du premier point sur l'axe des abscisses ;
  - **x2** : spécifie la position du second point sur l'axe des abscisses ;
  - **y1** : spécifie la position du premier point sur l'axe des ordonnées ;
  - **y2** : spécifie la position du second point sur l'axe des ordonnées ;



```
<svg width="500" height="500">  
  <rect x="25px" y="25px" width="150" height="100" fill="grey" stroke="yellow"/>  
  <circle cx="110" cy="110" r="50" fill="red"/>  
  <line x1="50" y1="50" x2="150" y2="150" stroke="blue" stroke-width="5"  
    stroke-dasharray="5,3,2" stroke-linecap="round"/>  
</svg>
```

# Conventions et bonnes pratiques

- HTML n'est pas sensible à la casse mais il est recommandé d'écrire le nom des balises et attributs en **minuscule**
- En HTML5, il n'est plus nécessaire de fermer les balises orphelines . Donc, ces deux écritures sont équivalentes : **<input .... / >** et **<input .... >**
- Contrairement aux langages de programmation, il est recommandé en HTML5 de ne pas mettre d'espace avant et après l'opérateur
- En HTML5, la balise **<title>** est recommandée

# Conventions et bonnes pratiques

- En HTML5, il est recommandé d'indiquer la langue utilisée dans la page `<html lang="en-US">` pour aider les moteurs de recherche
- En HTML5, il est recommandé de donner un nom en minuscule aux fichiers.
- En HTML5, une nouvelle balise a été introduite afin d'ajuster l'affichage en fonction de l'écran  
`<meta name="viewport" content="width=device-width, initial-scale=1.0">`
- Pour tester la validité d'un code HTML5 : <https://validator.w3.org/>

# Atelier 1

cours réalisé par Anis Assas et Yessine Zekri