

**Web**

**Bruno BEAUFILS**

**2023/2024**

# 1. Introduction

## 2. Documents numériques

JavaScript

## 3. Internet (aspect technique)

## 4. DNS

## 5. World Wide Web

## 1. Introduction

## 2. Documents numériques JavaScript

## 3. Internet (aspect technique)

## 4. DNS

## 5. World Wide Web

# Création documentaire

- ▶ 2 aspects importants
  - ▶ **fond** : informations, données
  - ▶ **forme** : présentation
- ▶ Exemples
  - ▶ nombre : quantité vs notation/représentation (indo-arabes, romains, etc.)
  - ▶ roman : auteur vs typographe
- ▶ Informatique a permis de **séparer fond et forme** facilement
  - ▶ Avantages
    - ▶ **accessibilité** (handicap, automatisation, etc.)
    - ▶ **universalité** = pas lié à un seul mode de représentation
    - ▶ seules les **données** sont importantes dans les documents
  - ▶ Problèmes
    - ▶ Outils mal conçus (Word) et éducation/formation déficiente
    - ▶ Utilisateurs plus souvent habitués au **WYSIWYG** qu'au **WYSIWYM**

Web conçu pour être un système **universel** d'échange de documents

- ▶ 2 outils pour décrire les documents
  - ▶ HTML : description des informations
  - ▶ CSS : description de la mise en forme
- ▶ 1 langage de programmation pour manipuler le tout
  - ▶ Javascript

*fond*  
*forme*

# Création documentaire

- ▶ 2 aspects importants
  - ▶ **fond** : informations, données
  - ▶ **forme** : présentation
- ▶ Exemples
  - ▶ nombre : quantité vs notation/représentation (indo-arabes, romains, etc.)
  - ▶ roman : auteur vs typographe
- ▶ Informatique a permis de **séparer fond et forme** facilement
  - ▶ Avantages
    - ▶ **accessibilité** (handicap, automatisation, etc.)
    - ▶ **universalité** = pas lié à un seul mode de représentation
    - ▶ seules les **données** sont importantes dans les documents
  - ▶ Problèmes
    - ▶ Outils mal conçus (Word) et éducation/formation déficiente
    - ▶ Utilisateurs plus souvent habitués au **WYSIWYG** qu'au **WYSIWYM**

Web conçu pour être un système **universel** d'échange de documents

- ▶ 2 outils pour décrire les documents
  - ▶ HTML : description des informations
  - ▶ CSS : description de la mise en forme
- ▶ 1 langage de programmation pour manipuler le tout
  - ▶ Javascript

*fond*  
*forme*

# Création documentaire

- ▶ 2 aspects importants
  - ▶ **fond** : informations, données
  - ▶ **forme** : présentation
- ▶ Exemples
  - ▶ nombre : quantité vs notation/représentation (indo-arabes, romains, etc.)
  - ▶ roman : auteur vs typographe
- ▶ Informatique a permis de **séparer fond et forme** facilement
  - ▶ Avantages
    - ▶ **accessibilité** (handicap, automatisation, etc.)
    - ▶ **universalité** = pas lié à un seul mode de représentation
    - ▶ seules les **données** sont importantes dans les documents
  - ▶ Problèmes
    - ▶ Outils mal conçus (Word) et éducation/formation déficiente
    - ▶ Utilisateurs plus souvent habitués au **WYSIWYG** qu'au **WYSIWYM**

Web conçu pour être un système **universel** d'échange de documents

- ▶ 2 outils pour décrire les documents
  - ▶ HTML : description des informations
  - ▶ CSS : description de la mise en forme
- ▶ 1 langage de programmation pour manipuler le tout
  - ▶ Javascript

*fond*  
*forme*

## 1. Introduction

## 2. Documents numériques JavaScript

## 3. Internet (aspect technique)

## 4. DNS

## 5. World Wide Web

# JavaScript : un langage de programmation

## ▶ Langage de programmation complet

- ▶ Variables, Séquences, Alternatives, Répétitions, Fonctions
- ▶ Normalisé par [ECMAScript](#)
  - ▶ langage interprété, à objets et prototypes, avec typage faible dynamique

## ▶ Beaucoup de navigateurs web **intègrent** un interpréteur [JavaScript](#)

- ▶ permet d'exécuter des programmes pour rendre un fichier HTML **dynamique**
  - ▶ modification de l'arbre HTML et des styles appliqués
  - ▶ fonction de connexion directe au serveur (XMLHttpRequest)
  - ▶ interfaces directes au moteur de rendu
  - ▶ nombreuses librairies (stockage/multimédia/etc.)
- ▶ permet de faire de véritables **applications web**

## ▶ Les applications mobiles sont souvent de *simples* applications web

- ▶ HTML + CSS + JS
- ▶ nombreux *Frameworks* (CSS+JavaScript) pour faciliter la mise en forme
  - ▶ sémantique : [SimpleCSS](#), [PicoCSS](#), etc.
  - ▶ non sémantique : [Bootstrap](#), [PureCSS](#), [W3.CSS](#), [KnaCSS](#), etc.

## ▶ Aucun rapport avec le langage Java sinon l'aspect propriétaire du nom



# JavaScript : un langage de programmation

## ▶ Langage de programmation complet

- ▶ Variables, Séquences, Alternatives, Répétitions, Fonctions
- ▶ Normalisé par [ECMAScript](#)
  - ▶ langage interprété, à objets et prototypes, avec typage faible dynamique

## ▶ Beaucoup de navigateurs web **intègrent** un interpréteur [JavaScript](#)

- ▶ permet d'exécuter des programmes pour rendre un fichier HTML **dynamique**
  - ▶ modification de l'arbre HTML et des styles appliqués
  - ▶ fonction de connexion directe au serveur (XMLHttpRequest)
  - ▶ interfaces directes au moteur de rendu
  - ▶ nombreuses librairies (stockage/multimédia/etc.)
- ▶ permet de faire de véritables **applications web**

## ▶ Les applications mobiles sont souvent de *simples* applications web

- ▶ HTML + CSS + JS
- ▶ nombreux *Frameworks* (CSS+JavaScript) pour faciliter la mise en forme
  - ▶ sémantique : [SimpleCSS](#), [PicoCSS](#), etc.
  - ▶ non sémantique : [Bootstrap](#), [PureCSS](#), [W3.CSS](#), [KnaCSS](#), etc.

## ▶ Aucun rapport avec le langage Java sinon l'aspect propriétaire du nom

# JavaScript : un langage de programmation

- ▶ **Langage de programmation complet**
  - ▶ Variables, Séquences, Alternatives, Répétitions, Fonctions
  - ▶ Normalisé par [ECMAScript](#)
    - ▶ langage interprété, à objets et prototypes, avec typage faible dynamique
- ▶ Beaucoup de navigateurs web **intègrent** un interpréteur [JavaScript](#)
  - ▶ permet d'exécuter des programmes pour rendre un fichier HTML **dynamique**
    - ▶ modification de l'arbre HTML et des styles appliqués
    - ▶ fonction de connexion directe au serveur (XMLHttpRequest)
    - ▶ interfaces directes au moteur de rendu
    - ▶ nombreuses librairies (stockage/multimédia/etc.)
  - ▶ permet de faire de véritables **applications web**
- ▶ Les applications mobiles sont souvent de *simples* applications web
  - ▶ HTML + CSS + JS
  - ▶ nombreux *Frameworks* (CSS+JavaScript) pour faciliter la mise en forme
    - ▶ sémantique : [SimpleCSS](#), [PicoCSS](#), etc.
    - ▶ non sémantique : [Bootstrap](#), [PureCSS](#), [W3.CSS](#), [KnaCSS](#), etc.
- ▶ Aucun rapport avec le langage Java sinon l'aspect propriétaire du nom

# Un accès à l'arbre du document

- ▶ des fonctions qui permettent d'accéder à l'arbre du document
  - ▶ lecture et modification de la structure et des propriétés
  - ▶ accès (réaction) aux événements gérés par le navigateur
  - ▶ accès via le **Modèle Objet du Document (DOM)**
    - ▶ types de variables (Element, HTMLDocument, etc.)
    - ▶ variables (document, window, etc.)
- ▶ insérables dans le code HTML via l'élément `<script>`
  - ▶ directement

```
<script>  
  alert("Une fenêtre dans la page");  
</script>
```

- ▶ via un lien

```
<script src="fichier.js"></script>
```

- ▶ un exemple : <src/gotlib-js.html>

# Quelques fonctions, propriétés et instructions utiles

```
// retourne le premier élément sélectionné par XXX
document.querySelector("XXX");

// retourne tous les éléments sélectionnés par XXX dans un tableau
document.querySelectorAll("XXX");

// répète les actions AAA et BBB sur tous les objets du tableau TTT
// 0 représente successivement toutes les valeurs de ces objets
// 0 est utilisable dans AAA et BBB
for ( 0 of TTT ) { AAA; BBB; }

// le nombre d'éléments sélectionnés par XXX
document.querySelectorAll("XXX").length

// le contenu HTML d'un élément
element.innerHTML

// le contenu texte d'un élément
element.innerText

// affiche le message MMM sur la console
console.log(MMM);
```

- ▶ Console web
  - ▶ Accès rapide : Shift + Control + K
  - ▶ Permet d'agir dynamiquement sur le contenu affiché
- ▶ Utilisation des **sélecteurs CSS** pour interroger le contenu d'une page
  - ▶ accès aux propriétés du DOM

```
document.querySelector(XXX);
```

- ▶ beaucoup de raccourcis

- ▶ \$("XXX")

pareil que : document.querySelector("XXX")

- ▶ \$\$("XXX")

pareil que : document.querySelectorAll("XXX")

- ▶ exemples :

- ▶ afficher le **contenu** de **tous** les noeuds sélectionnés par XXX

```
for ( s of $$("XXX") ) { console.log(s.textContent); }
```

- ▶ afficher le **nombre** de noeuds sélectionnés par XXX

```
$$("XXX").length
```





# Exercice

- ▶ Intéressez-vous à la page :  
<https://www.univ-lyon2.fr/formation/m2-droit-economie-gestion>
- ▶ Faites afficher dans la console web
  - ▶ le nombre de masters 2
  - ▶ la liste de tous les masters 2
  - ▶ la liste de tous les masters 2 de l'UFR des *Sciences économiques et de gestion*



# Web Scraping

- ▶ **web scraping** = extraire des données dans le contenu d'une page web
  - ▶ moissonnage (*harvesting*)
  - ▶ utilisation des sélecteurs CSS (ou de [XPath](#)) pour choisir quoi récupérer
- ▶ un domaine où les sélecteurs CSS prennent leur sens
  - ▶ permettent de construire vos jeux de données

- ▶ **rvest** : une bibliothèque R pour faciliter l'extraction
- ▶ quelques fonctions intéressantes
  - ▶ **read\_html** : récupère un source HTML (à partir d'une URL)
  - ▶ **html\_element** : récupère le **premier** noeud sélectionné par un sélecteur CSS
    - ▶ correspond à `document.querySelector()`
  - ▶ **html\_elements** : récupère **tous** les éléments sélectionnés par un sélecteur CSS
    - ▶ renvoie une liste au sens R
    - ▶ correspond à `document.querySelectorAll()`
  - ▶ **html\_text** : ne conserve que le contenu textuel dans un source HTML
    - ▶ correspond à `.innerText`
  - ▶ **html\_attr** : récupère la valeur d'un attribut dans un source HTML
  - ▶ **html\_table** : récupère le contenu d'une table
    - ▶ renvoie un *data frame* au sens R
- ▶ beaucoup d'autres fonctions disponibles

## Scrapons Gotlib

```
# installation du paquet
install.packages("rvest")

# utilisation du paquet
library(rvest)

# récupère la page
page <- read_html("https://m2.ape-cee.fr/src/gotlib-js.html")

# récupère dans page le **premier** noeud h2
premier <- html_element(page, "h2")

# récupère dans page *tous* les éléments h2
tous <- html_elements(page, "h2")

# ne conserve que le contenu textuel
html_text(tous)

# récupère la valeur de l'attribut id
html_attr(tous, "id")

# Exemples courts
html_text(html_elements(read_html("https://m2.ape-cee.fr/src/gotlib.html"), "h2"))
html_attr(html_elements(read_html("https://m2.ape-cee.fr/src/gotlib.html"), "img"), "alt")
```

# Devoir : sélection d'éléments

## 1. Regardez la page <src/etudiants.html>

- ▶ elle contient les listes des étudiants du master depuis 2016
- ▶ elle vous permet de sélectionner des informations
  - ▶ remplir le champ *Sélecteur*
  - ▶ puis valider ou cliquer sur le bouton *Montrer*

## 2. Déterminez quel sélecteur CSS permet d'obtenir :

**2.1** les années universitaires disponibles

**2.2** le **nom** de tous les étudiants de 2018-2019

**2.3** le **prénom** de toutes les filles de toutes les années

**2.4** le **nom** de tous les seconds étudiants de chaque année

## 3. Étudiez en détail le contenu de la page

<http://ww2.assemblee-nationale.fr/deputes/liste/departements>

## 4. Déterminez quel sélecteur CSS permet d'y obtenir :

- ▶ la liste des députés du Rhône (bas et haut)

# Devoir : sélection d'éléments (suite)

## 5. Envoyez-moi vos 5 **sélecteurs** dans un fichier nommé `selecteurs`

- ▶ 1 sélecteur par question, 1 question par ligne
  - ▶ votre fichier doit faire exactement 5 lignes
- ▶ adresse : **bruno.beaufils@univ-lille.fr**
- ▶ objet du mail : **[M2APE] devoir 3 Prénom NOM**
- ▶ le **strict** respect de ces contraintes est important

# Références

- ▶ Mozilla Developer Network (JavaScript)
  - ▶ La console web
  - ▶ L'ardoise JavaScript
  - ▶ (Références JavaScript)
- ▶ Modèle Objet du Document
- ▶ W3schools
  - ▶ JavaScript Tutorial

## 1. Introduction

## 2. Documents numériques

JavaScript

## 3. Internet (aspect technique)

## 4. DNS

## 5. World Wide Web

## 1. Introduction

## 2. Documents numériques

JavaScript

## 3. Internet (aspect technique)

## 4. DNS

## 5. World Wide Web



## 1. Introduction

## 2. Documents numériques

JavaScript

## 3. Internet (aspect technique)

## 4. DNS

## 5. World Wide Web