

*Technologies
Web
(JavaScript)*

DA/IA2
Henallux

Module 1

Introduction à JavaScript

La place de JS dans le web

➤ Rappels de 1^{re}

- Site web, HTML, CSS

➤ Pages HTML-dynamiques

- Ou « à quoi sert JavaScript (pour le web) ? »

➤ Présentation du langage JavaScript

- Particularités du langage, premières lignes en JavaScript

➤ Utilisation de JavaScript

- Comment combiner HTML, CSS et JavaScript
De quoi pouvoir réaliser le premier laboratoire

Quelques rappels de 1^{re}

Au programme de ce chapitre...

➤ **Le fonctionnement d'un site web**

- *Navigateur et serveur*

➤ **Les standards utilisés**

- *HTML, CSS*

Ensuite : *Pages HTML-dynamiques*

Un site web

- Comment fonctionne un site web ?



- envoie les demandes,
- réceptionne les réponses et
- gère l'affichage (comprend les standards HTML et CSS).

serveur = à la fois

- machine physique accessible via internet et
- logiciel-serveur capable de répondre aux demandes de fichiers

Un site web

- Le navigateur et le serveur **communiquent** via...
 - *Envoi/réception de messages* : le **protocole** HTTP (HyperText Transfer Protocol)
 - pour envoyer une demande « je veux doc.html »
 - pour envoyer la réponse « voici le contenu de doc.html »
 - *Informations échangées* : des **standards** du web
 - **HTML** : HyperText Markup Language
 - **CSS** : Cascading Style Sheets
 - organisés par le W3C
 - conventions plus ou moins bien respectées selon les navigateurs (certains plus que d'autres...)



Le langage HTML

Un document HTML est un **fichier texte**.

```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
...
```

```
</head>
```

Partie **"en-tête"** :

- infos techniques (titre, langue, encodage, ...)
- d'autres éléments non directement visibles

```
<body>
```

```
...
```

```
</body>
```

Partie **"corps"** :

- éléments affichés sur la page web

```
</html>
```


Le langage HTML

- Le langage **HTML** (parent/descendant du XML) repose sur des **balises**.
 - Toute balise ouvrante doit être fermée : `<p>...</p>`
 - Balises sans intérieur : `
` `<meta ... />`
 - Balises imbriquées, pas de croisement ! `<p> </p>`
 - Parfois avec des attributs : `<meta charset="utf-8" />`
- « **Clean code** »
 - Balises et attributs écrits en minuscules
 - Valeurs des attributs entre guillemets (cas spécial : booléens)
 - Indentation correcte (HTML ignore les blancs multiples)
 - Respect de la syntaxe (malgré les navigateurs permissifs)

Le langage CSS

- Séparation **forme / contenu**
 - HTML s'occupe du fond/contenu – CSS s'occupe de la forme.
 - Éviter les balises `` et `<i>` en HTML
 - Pourquoi cette distinction ? En connaître les avantages...
- Trois manières possibles pour utiliser du CSS :
 - Format **inline** :

```
<h2 style="color:blue">Mon sous-titre</h2>
```
 - Format **interne** (dans l'en-tête) :

```
<style>
  h2 {color:blue}
</style>
```
 - Format **externe** (dans un fichier séparé) :

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


Le langage CSS

- Syntaxe d'une **règle CSS** :

```
h2 { color:blue ; font-size:250% }
```

Sélecteur

= cible(s) de la règle

Liste de déclarations

au format *propriété : valeur*

- Quelques sélecteurs (voir aussi les slides de 1^{re}) :
 - Balise `h2` ou identificateur `#maphoto` ou classe `.nom`
 - Combinaisons : `#contenu p.intro a:first-of-type`
- « **Clean code** »
 - Tout écrit en minuscules.
 - Bien distinguer identificateur et classe !
 - Bien choisir les noms des classes !
 - Respect de la syntaxe (malgré les navigateurs permissifs)

Pages HTML-dynamiques

Au programme de ce chapitre...

- **Page statique et page HTML-dynamique**
 - *Quelles sont les différences ?*
- **Que permet JavaScript ?**

Ensuite : *Présentation du langage JavaScript*

Statique et HTML-dynamique

- Page web **statique** :



- contenu et apparence déterminés par le code HTML/CSS envoyé par le serveur ;
- et ce contenu et cette apparence ne changent pas.

Statique et HTML-dynamique

- Page web **HTML-dynamique** :



- possède un contenu et une apparence *initiales* déterminées par le code HTML/CSS envoyé par le serveur ;
- et ce contenu et cette apparence peuvent être modifiés !
 - le navigateur gère les modifications **en local** (sans le serveur).
 - le code envoyé contient des **scripts** indiquant comment le navigateur doit (ré)agir, généralement écrits en JavaScript.

Statique et HTML-dynamique

- (Pour info) Page web **dynamique** :



- Le document envoyé est construit par le serveur et peut être différent pour chaque demande
 - selon la source de la demande (personne connectée),
 - l'heure, l'état du serveur, ou d'autres informations (base de données...)
 - ce qui nécessite du code sur le serveur (par exemple PHP, NodeJS, ASP.NET...)

Statique et HTML-dynamique

- JS permet le **DHTML** (= Dynamic HTML).

$$\text{DHTML} = \text{HTML} + \text{CSS} + \text{JavaScript}$$

HTML

Description structurée du contenu de la page

CSS

Formatage / Apparence du contenu

JavaScript

Actions à exécuter par le navigateur, pouvant modifier le contenu et l'apparence de la page web



Que permet JavaScript ?

- JavaScript permet d'écrire des **scripts clients**.
 - **Scripts** : programmes destinés à être interprétés directement
 - plutôt que compilés puis exécutés...
 - (la différence n'est plus aussi nette que ça aujourd'hui.)
 - **Clients** : tout se déroule sur l'ordinateur de l'internaute.

- Par exemple, JavaScript peut

- ajouter l'heure sur la page au chargement
- modifier l'heure affichée toutes les secondes
- afficher un message quand l'utilisateur clique sur un bouton
- afficher ou cacher un menu selon la position du curseur
- demander une information à l'utilisateur (son nom)
- manipuler le navigateur (redirection vers une autre page)
- permettre de basculer entre un thème clair et un thème foncé

*Événements déclenchant
l'exécution d'un script*



Que permet JavaScript ?

- Un exemple :

Choisissez votre année : 0

Choisissez votre groupe : ☒ A ☐ B ☐ C ☐ D ☐ E

En fin d'année, vous devriez être prêt à rentrer en IG2.

Le langage JavaScript

Au programme de ce chapitre...

- **Caractéristiques du langage JavaScript**
 - *Quel type de langage ? Et par rapport à C et à Java ?*
- **Règles syntaxiques générales de JavaScript**
- **Premières lignes en JavaScript**

Ensuite : *Utilisation de JavaScript*

Caractéristiques de JavaScript

JavaScript is the only language that I'm aware of that people feel they don't need to learn before they start using it.

Douglas Crockford

- **JavaScript** est...
 1. un langage de programmation
 2. impératif,
 3. orienté objet,
 4. événementiel et
 5. faiblement / non typé,
 6. où les fonctions sont des objets de premier ordre,
 7. qui évolue...
- **Qu'est-ce que tout cela signifie ?**

Caractéristiques de JavaScript

- (1) JS est un **langage de programmation**.
 - contrairement à HTML et à CSS, qui sont des langages informatiques **de description** (ils ne permettent pas d'écrire un programme)
- (2) ... un langage **impératif** (comme C, Java).
 - programme = séquence d'instructions
- (3) ... un langage **orienté objet** (comme Java).
 - mais JavaScript a une approche OO bien différente de Java !
 - JavaScript : **OO prototypal** <> Java : **OO de classe/classique**
 - même s'il existe une syntaxe qui cache les différences...



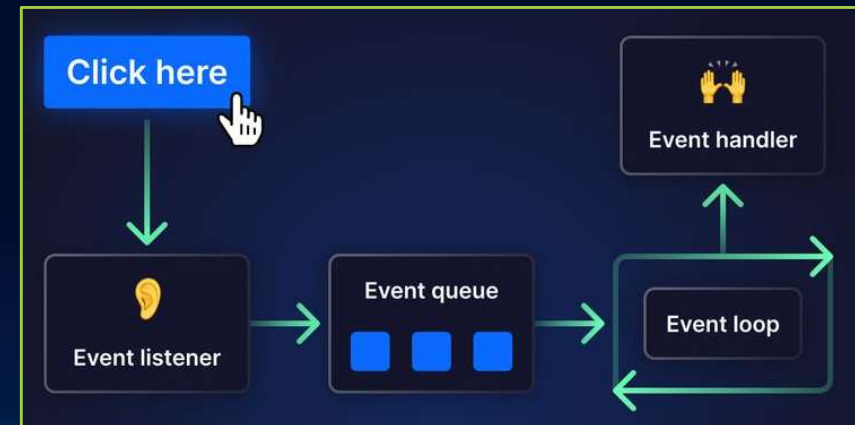
Caractéristiques de JavaScript

- (4) ... conçu pour la **programmation événementielle**.

- = certains modules s'exécutent en réponse à des déclencheurs

- Exemples d'**événements** :

- clic sur un bouton
- fin du chargement de la page
- passage du curseur sur une image ou un élément de menu



- Programmation événementielle = préciser ce qui se passe en réponse à des événements
 - ≠ programme C ou Java avec 1 module « main »

Caractéristiques de JavaScript

- (5) JavaScript est un langage **non typé**.

- Le type du contenu d'une variable peut évoluer !

```
x = 3; // x contient un nombre  
x = "One more block"; // x contient un string
```

- Quand on déclare une variable, on ne précise pas de type (idem pour les arguments d'une fonction).

```
function double(x) { return x + x; }  
// double(4) → 8  
// double("bye") → "byebye"
```

- **Danger** : certaines erreurs ne sont pas repérées automatiquement.
 - Raison de plus de programmer soigneusement et avec une réflexion préalable !

Caractéristiques de JavaScript

- (6) JS traite les fonctions comme des **objets de premier ordre**.

- Objets de premier ordre = objets qu'on peut
 - affecter à une variable,
 - passer comme argument à une fonction,
 - retourner comme résultat d'une fonction.

- Exemples :

```
function somme (x,y) { return x+y; };  
// somme(2,3) → 5
```

```
function applique (g,a,b) { return g(a,b); };  
// applique(somme,2,3) → 5
```

Caractéristiques de JavaScript

- (7) JavaScript est un **langage qui évolue**.
 - [1995] Créé chez Netscape (par Brendan Eich)
 - course pour établir des standards pour le web
 - nommé Mocha, Livescript, puis JavaScript
 - Quelques alternatives sans succès (Microsoft) : VBScript, Jscript
 - [1996] Adopté comme standard
 - soumis à l'ECMA (European Computer Manufacturer Association)
 - renommé (officiellement) en ECMAScript (ou ES)
 - [2015] ES6 = EcmaScript2015 (grosse réforme)
 - nouvelle syntaxe pour l'orienté-objet en JavaScript
 - depuis : nouvelle version chaque année
 - Évolution additive (pour assurer la rétrocompatibilité)
 - D'où beaucoup de code existant, parfois avec des approches différentes
 - D'où aussi souvent plusieurs méthodes pour accomplir certaines choses
 - Pas utilisé uniquement pour le web (NodeJS, Electron...)



Caractéristiques de JavaScript

- **JavaScript \neq Java !**

	JavaScript	Java
<i>Créateurs</i>	Netscape	Sun / Oracle
<i>Exécution</i>	interprété	compilé
<i>Objets</i>	à base d'objets	à base de classes
<i>Typage</i>	non typé, dynamique	fortement typé, statique
<i>Utilisation</i>	propre au web	langage indépendant
<i>Fichiers</i>	aucun accès (a priori)	lecture, écriture...
<i>Sécurité</i>	code public	

- Un point commun : l'utilisation d'un garbage collector.

Règles syntaxiques générales

- (Comme HTML et CSS,) JavaScript **ignore les blancs**.
 - **Blanc** = espace, tabulation, retour à la ligne...
 - autorise une indentation claire [clean code]

- En pratique (**pas pour ce cours**) : **scripts minimalisés**

- But : aussi court que possible (fichier plus petit)

- Exemple :

```
function fb(a,b,d,e){var
f,h,j,k,l,o,r,s,w,x;if((b?b.ownerDocument||b:v)!=n&&m(b),b=b||
n,d=d||[],!a||"string"!=typeof a)return
d;if(1!=(k=b.nodeType)&&9!=k)return[];if(p&&!e)
{if(f=_.exec(a))if(j=f[1]){if(9===k){if(h=b.getElementById(j),!h||!h.parentNode)re
turn d;if(h.id===j)return d.push(h),d}else
if(b.ownerDocument&&(h=b.ownerDocument.getElementById(j))&&t(b,h)&&h.id===j)return
d.push(h),d}else{if(f[2])return
I.apply(d,b.getElementsByTagName(a)),d;if((j=f[3])&&
c.getElementsByClassName&&b.getElementsByClassName)return I.apply(d,
b.getElementsByClassName(j)),d;if(c.qsa&&(!q||!q.test(a))){if(s=r=u,w=b,x=9===k&&a
,1===k&&"object"!=b.nodeName.toLowerCase()){o=g(a),(r=b.getAttribute("id"))?s=r.r
eplace(bb,"\\$&"):b.setAttribute("id",s),s="[id='"+s+"' ]",l=o.length;while(l--
)o[l]=s+qb(o[l]);
```

Règles syntaxiques générales

- Écrire des **commentaires** en JavaScript (syntaxe du C/Java)
 - `//` pour le reste de la ligne
 - `/* ... */` pour plusieurs lignes / une partie de ligne
 - [Rappels] En HTML : `<!-- ... -->` ; En CSS : `/* ... */`
- À propos des **identificateurs** en JavaScript :
 - commencent par une lettre, `$` ou `_`
 - composés de lettres, chiffres, `$` et `_`
 - (sauf les mots réservés)
 - (éviter les `$` et `_` en début : variables prédéfinies)
- JavaScript est **sensible à la casse** !
 - `mavariabLe` \neq `maVariable` \neq `MAVARIABLE`

Règles syntaxiques générales

- Liste des mots réservés en JavaScript

abstract	debugger	final	instanceof	public	transient
boolean	default	finally	int	return	true
break	delete	float	interface	short	try
byte	do	for	long	static	typeof
case	double	function	native	super	var
catch	else	goto	new	switch	void
char	enum	if	null	synchronized	volatile
class	export	implements	package	this	while
const	extends	import	private	throw	with
continue	false	in	protected	throws	

Note : certains de ces mots ne sont pas utilisés dans la version actuelle de JavaScript...

Règles syntaxiques générales

- Les instructions en JavaScript se terminent normalement par ;.
- Le ; est facultatif : un retour à la ligne peut suffire (à éviter).
 - « ASI » : Automatic Semi-colon Insertion
- Mais **attention aux ambiguïtés !**

```
function produit (x, y) {  
    let prod =  
        x * y  
    return  
        x * y  
}
```

Utilisation de Javascript

Au programme de ce chapitre...

➤ Premières lignes en JavaScript

- *Tout premier aperçu pour pouvoir réaliser les premiers exercices*

➤ Où placer le code JavaScript ?

- *Plus ou moins similaire aux options pour le CSS*

Premières lignes en JS

- Syntaxe fort proche du C ou de Java, mais sans les types.
- **Déclaration de variable** (sans / avec initialisation)
 - `let x;`
 - `let x = 42;`
- **Valeurs possibles** (premier aperçu)
 - Nombres (entiers ou réels) : `résultat = -217 * 53.17;`
 - Caractère/chaîne : `msg = "hello" + ' world';`
 - Booléens : `réussite = (dossier == "ok" && cote >= 10);`
 - Opérations : syntaxe standard

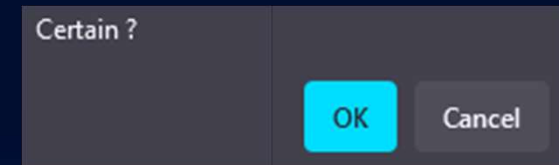
Premières lignes en JS

Note : uniquement pour les premiers exercices/tests !

- **Entrées**

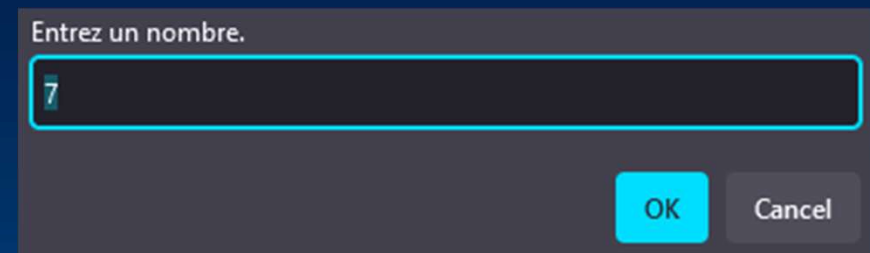
- `res = confirm(txt)` : demande une confirmation

- renvoie `true` si l'utilisateur a cliqué sur "Ok"
- renvoie `false` si l'utilisateur a cliqué sur "Cancel"



- `res = prompt(txt [,valDefault])` : demande une valeur

- propose éventuellement une valeur par défaut
- Exemple : `let x = prompt("Entrez un nombre.", 7);`




Premières lignes en JS

Note : uniquement pour les premiers exercices/tests !

- **Sorties**

- `document.write(txt)` : vers le document HTML
- `console.log(txt)` : vers la console du navigateur
- `alert(txt)` : dans une fenêtre popup

A screenshot of a JavaScript alert dialog box. It has a dark gray background with the text "Message important !" in white. In the bottom right corner, there is a red button with the text "OK" in white.

Message important !

OK

Premières lignes en JS

- **Déclaration et appel de fonction**

- ```
function alertImportant (msg) {
 let texte = "MESSAGE IMPORTANT :\n" + msg;
 alert(texte);
}
```



MESSAGE IMPORTANT:  
Carpe Diem !

`alertImportant("Carpe Diem !")`

- ```
function carré (x) {  
    return x * x;  
}
```

`let neuf = carré(3);`

Premières lignes en JS

- **Quelques instructions** (syntaxe proche du C et du Java)

- **Affectations**

```
bloc = 1;  
bloc++;
```

- **Conditionnelle**

```
if (bloc > 1) {  
    res = "plus de pitié !";  
}
```

ainsi que switch

- **Boucles**

```
while (nb > 1) {  
    nb /= 2;  
}
```

```
for (let nb = 1 ; nb < 5 ; nb++) {  
    console.log(nb);  
}
```

Où placer le code JS ?

- Option 1 : **code "interne"** (dans une balise `script`)
 - `<script> codeJS </script>`
 - Il s'agit **d'exécution synchrone** (immédiate).
 - On peut avoir plusieurs balises `script`.
 - Le navigateur exécute le script dès qu'il le rencontre lors de la lecture du fichier HTML.
 - Texte à afficher si JS est désactivé :

```
<noscript>  
  Ce browser ne supporte pas JS !  
</noscript>
```



Où placer le code JS ?

```
<html>
  <head>
    <script>
      alert("Bienvenue sur ma page web !");
    </script>
  </head>
  <body>
    <p>Ceci est le contenu du premier paragraphe.</p>
    <p>Et voici un
      <script>
        document.write("<a href='http://www.google.be'>lien</a>");
      </script>
    vers Google !
  </p>
</body>
</html>
```



Où placer le code JS ?

```
<html>
  <head>
    <script>
      alert("Script dans head !");
      function afficheHeure () {
        let now = new Date ();
        let h = now.getHours();
        let m = now.getMinutes();
        let s = now.getSeconds();
        document.write(h + ":" + m + ":" + s);
      }
    </script>
  </head>
  <body>
    <script>alert("J'entre dans body !");</script>
    <p>Il est exactement : <script>afficheHeure();</script></p>
    <p>Ceci est mon deuxième paragraphe.</p>
    <script>alert("Après 2e paragraphe.");</script>
    <p>Il est maintenant <script>afficheHeure();</script>.</p>
  </body>
</html>
```

Où placer le code JS ?

- Option 2 : **code "inline"**
 - Dans un attribut correspondant à un événement :
`<button onclick="codeJS">Cliquez moi!</button>`
 - Dans une référence href :
`text`
 - Il s'agit **d'exécution asynchrone** (reportée à plus tard, quand un événement déclencheur se produit).



Où placer le code JS ?

```
<html>
  <head>
    <script>
      function message () {
        alert("Voici un message !");
      }
    </script>
  </head>
  <body>
    <p>Pour recevoir un message, cliquez
      <a href="JavaScript:message();">sur ce lien</a>
      ou passez la souris sur ce
      <button onmouseover="message();">bouton</button>
    .</p>
  </body>
</html>
```

Où placer le code JS ?

- Option 3 : **code "externe"**
 - Dans un fichier séparé lié via une balise script :
`<script src="nomFichier.js"></script>`
 - Ici, **exécution synchrone** (immédiate).

Où placer le code JS ?

- **Résumé des 3 options**

- **inline et interne**

- peut être utile lors des tests (ou exercices)
- à éviter dans la version finale d'un site

- **externe**

- À préférer
- Lien `<script>` généralement placé dans l'en-tête `<head>`
- *Principaux avantages :*
 - code découplé de la page web (réutilisabilité, évolution...)
 - code chargé séparément du fichier HTML (peut être "caché" indépendamment)