

Module 1

Introduction à Javascript

Technologies web
HENALLUX — IG2

La place de JS dans le web

➤ **Rappels d'IG1**

- Site web, HTML, CSS

➤ **Pages HTML-dynamiques**

- Ou « à quoi sert Javascript ? »

➤ **Présentation du langage Javascript**

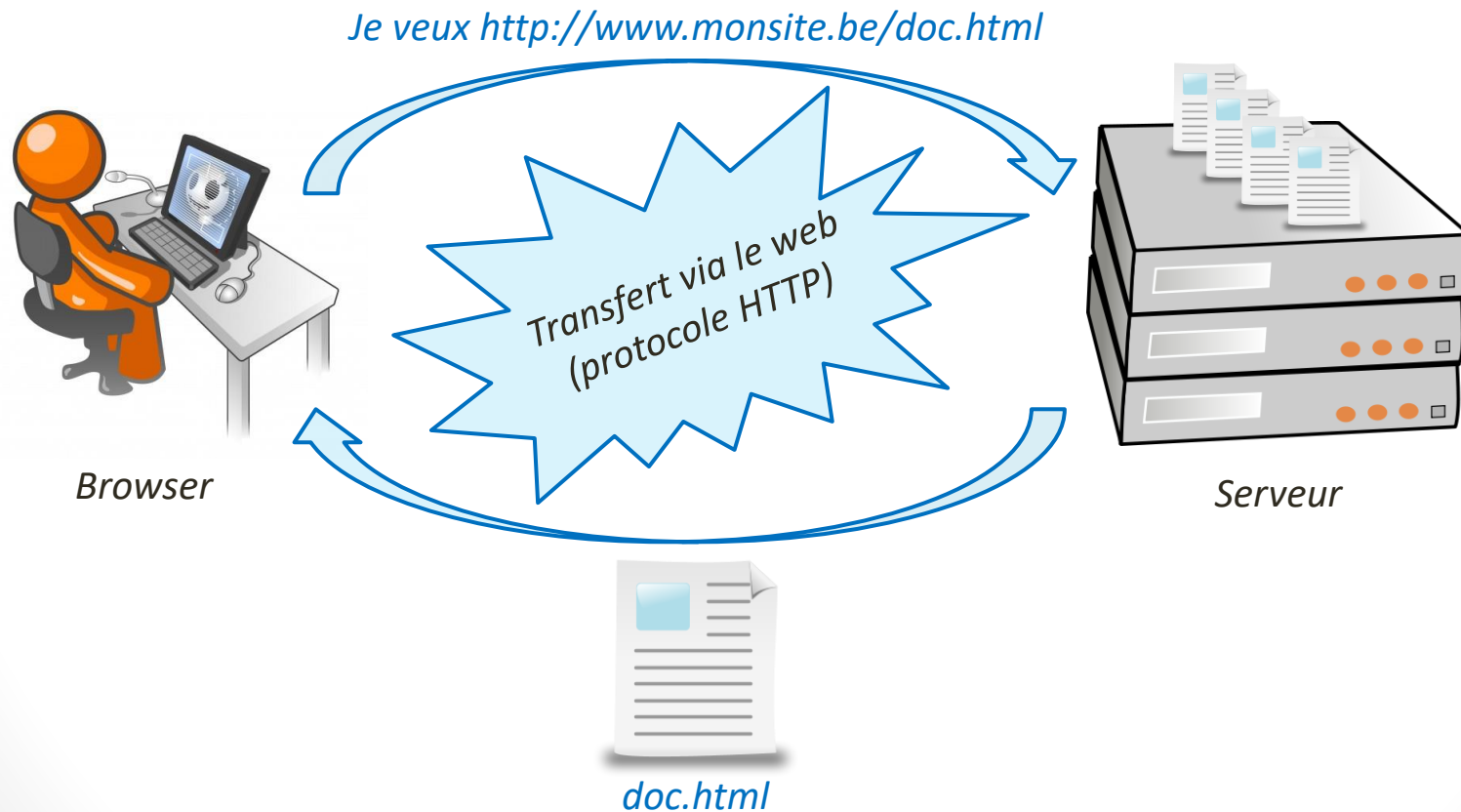
- Particularités du langage, premières lignes en Javascript

➤ **Utilisation de Javascript**

- Comment combiner HTML, CSS et Javascript

Un site web

- Comment fonctionne un site web ?



Un site web

- **Quelles sont les ressources utilisées ?**
 - un **navigateur** utilisé par l'internaute
 - qui envoie les demandes,
 - réceptionne les réponses et
 - gère l'affichage (comprend les standards HTML et CSS).
 - un **serveur**
 - serveur = à la fois machine physique et logiciel
 - machine physique accessible via internet
 - logiciel-serveur capable de répondre aux demandes de fichiers (par exemple Apache)

Un site web

- Pour pouvoir **communiquer** entre eux, le navigateur et le serveur utilisent...
- (aspect concret) le **protocole** http (hypertext transfer protocol)
 - pour envoyer une demande « je veux doc.html »
 - pour envoyer la réponse « voici le contenu de doc.html »
- (aspect logique) des **standards** du web
 - HTML : HyperText Markup Language
 - CSS : Cascading Style Sheets
 - organisés par le W3C
 - conventions plus ou moins bien respectées par les navigateurs (certains plus que d'autres...)

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>
```

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 (les blancs répétés sont ignorés)
 - Respect de la syntaxe (malgré les navigateurs permissifs)


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 utilisations possibles 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" />`

CSS

- Une **règle CSS** comporte... `h2 {color:blue; font-size:250%}`
 - un **sélecteur** (désigne les cibles de la règle),
 - Balise `h2` ou identificateur `#maphoto` ou classe `.nom`
 - Nombreuses combinaisons : `#contenu p.intro a:first-of-type`
 - une liste de **déclarations**, chacune d'elles étant composée...
 - d'une **propriété** (indique quel aspect modifier) et
 - d'une **valeur** (indique comment le modifier).
- « **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

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

Ensuite : *Présentation du langage Javascript*

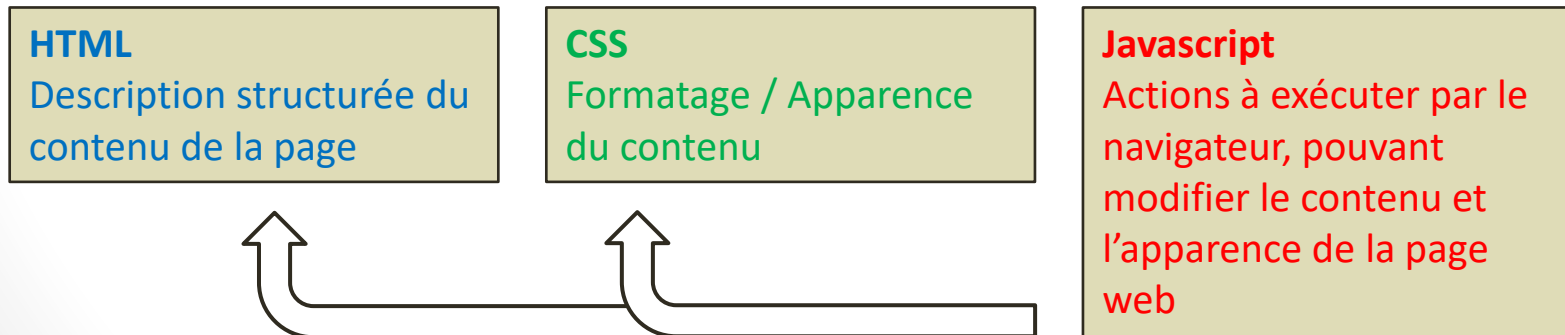
Statique et HTML-dynamique

- Une page web **statique**...
 - possède un contenu et une apparence déterminée par le code HTML/CSS envoyé par le serveur ;
 - et ce contenu et cette apparence ne changent pas.
- Une 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 modifiées !
- Sur une **page HTML-dynamique**,...
 - 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

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

$$\begin{array}{c} \text{DHTML} \\ = \\ \text{HTML} + \text{CSS} + \text{Javascript} \end{array}$$



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 quelque part dans le contenu de la page
 - 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

Que permet Javascript ?




- Un exemple :

Choisissez votre année :

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

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

Présentation de Javascript

- 
- **Caractéristiques du langage Javascript**
 - Quelques particularités générales de Javascript
 - **Règles syntaxiques générales de Javascript**
 - **Premières lignes en Javascript**

Ensuite : *Utilisation de Javascript*

Caractéristiques de Javascript

- **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**
 - même si la (nouvelle) syntaxe cache les différences...

Caractéristiques de Javascript

- (4) ... conçu pour la **programmation événementielle**.
 - = programmation où certains modules s'exécutent en réponse à des éléments-déclencheurs
 - Quelques exemples d'**événements** :
 - clic sur un bouton
 - passage du curseur sur une image ou un élément de menu
 - fin du chargement de la page
 - Programmation événementielle = préciser ce qui doit se passer 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).
 - **Danger** : certaines erreurs ne sont pas repérées automatiquement !

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 :

```
let somme = function (x,y) { return x+y; };  
z = somme(2,3);
```

```
let applique = function (g,a,b) { return g(a,b); };
```

```
z = applique(somme,2,3);
```

Caractéristiques de Javascript

- (7) Javascript est un **langage qui évolue**.
 - [1995] Créé par chez Netscape (par Brendan Eich)
 - course pour établir des standards pour le web
 - nommé Mocha, Livescript, puis Javascript
 - Quelques alternatives sans succès (Microsoft) : VBScript,
 - [1996] Finalement 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
 - Popularité de plus en plus grande
 - développement de nombreuses bibliothèques (jQuery, NodeJS...)
 - Voir par exemple <https://github.info/>



Caractéristiques de Javascript

- **Javascript ≠ Java !**

	Javascript	Java
Créateurs	Netscape	Sun / Oracle
Exécution	interprété	compilé
Objets	à base d'objets	à base de classes
Typage	non typé, dynamique	fortement typé, statique
Utilisation	propre au web	langage indépendant
Fichiers	aucun accès (a priori)	lecture, écriture...
Sécurité	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 minimisés**
 - 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)
return 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)retu
rn 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.replace(bb,"\\$&"):b.setAttribute("id",s),s="[id='"+s+"']"
",l=o.length;while(1--)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 Javascript se terminent normalement par ;.
- Le ; est facultatif : un retour à la ligne peut suffire (**à éviter [CC]**).
 - « ASI » : Automatic Semi-colon Insertion

- **Attention aux ambiguïtés !**

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

Utilisation de Javascript

- 
- **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 = prompt("Entrez un nombre.", 7);`
- **Valeurs possibles** (premier aperçu)
 - Nombres (entiers ou réels) : `resultat = -217 * 53.17;`
 - Caractère/chaîne : `msg = "hello" + ' world';`
 - Booléens : `reussite = (dossier == "ok" && cote >= 10);`
 - Opérations : syntaxe standard

Premières lignes en JS

- **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 [,valDefaut])` : demande une valeur
 - propose éventuellement une valeur par défaut

- **Sorties**

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

Premières lignes en JS

- **Déclaration de fonction**

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

Premières lignes en JS

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

- **Affectations**

```
annee = 1;
annee++;
```

- **Conditionnelle**

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

ainsi que `switch`

- **Boucles**

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

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

Où placer le code JS ?

- Solution 1 : **code "interne"** (dans une balise `script`)

- `<script> codeJS </script>`

(Obsolète) ajouter l'attribut `language="javascript"` ou `type="text/javascript"` pour distinguer de VBScript ou JScript

- Il s'agit d'**exécution synchrone** (immédiate).
 - On peut avoir plusieurs balises script.
 - Le navigateur exécute le contenu 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 ?

- Solution 2 : **code "inline"**
 - Dans un attribut correspondant à un événement :
`<button onclick="codeJS">Cliquez moi!</button>`
 - Dans une référence href :
`texte`
 - 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 ?

- Solution 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 solutions**
 - inline et interne
 - peut être utile lors des tests (ou exercices)
 - à éviter dans la version finale d'un site
 - externe
 - de préférence
 - placer le lien `<script>` dans l'en-tête `<head>`
 - principal avantage : le code Javascript est chargé séparément du fichier HTML (peut-être "caché" indépendamment)