INFO102

Langages du web 1

Langage du web 1 : JavaScript

- Rappel
- Bref historique
- Exemples
- Comparatif Python/JavaScript
- Programmation événementielle
- Rechercher, modifier et créer des éléments

Rappel HTML + CSS

- A quoi sert HTML ?
- HTML pour structurer le contenu (DOM) et le rendre compréhensible par les robots (sémantique)
- A quoi sert CSS ?
- CSS pour la mise la mise en forme
- démo scratch (glisser déposer par exemple) + afficher le code
- On comprend bien qu'il manque une brique (une grosse brique !) = JavaScript

Bref historique

- Inventé en 1995 par Brendan Eich informaticien chez **Netscape**. Brendan Eich participera aussi à la fondation **Mozilla**.
- A l'origine baptisé *LiveScript*, *Netscape* profite de la notoriété grandissante de **Java** pour le renommer *JavaScript*, artifice marketing qui 25 ans après porte encore préjudice au *JavaScript*...

Bref historique

- Standardisé en 1997 par l'organisation *ECMAScript* sous le numéro *ECMA-262*
- *JavaScript*, *JS*, *ECMAScript* ou *ES* désignent le même langage

Nom de l'édition	Année de publication	Événements notables
ES1	1997	Premier standard
ES2	1998	Corrections
ES3	1999	Améliorations (String, Error, Number…)
ES4	Abandonnée	Aucun accord entre les membres
ES5	2009	Support natif du format JSON
ES6 ou ES2015	2015	Nombreuses évolutions qui font du JavaScript un vrai langage moderne Voir chapitre « Réintroduction au JS », page 139
ES7 ou ES2016	2016	Itération annuelle Ajout de Array.includes()
ES8 ou ES2017	2017	Itération annuelle Ajout de `Object.values()` et `.entries()` Ajout de `String.padStart()` et `.padEnd()`
ES9 ou ES2018	2018	Itération annuelle Essentiellement des améliorations de performances du moteur JS
ES10 ou ES2019	2019	Itération annuelle Ajout de `Array.flat()` Ajout de `Function.toString()`
ES11 ou ES2020	2020	Itération annuelle Ajout du type BigInt
ESNext		Nom générique pour la future version à venir

Olivier Hondermarck
TOUT JavaScript
DUNOD 2020

Bref historique

- 10 ans entre *ES3* et *ES5*, une éternité dans l'histoire du web
- Aujourd'hui une nouvelle version tous les ans
- Probablement un des langages les plus importants et les plus prometteurs du moment

Sites de référence

- Mozilla !
- ToutJavaScript

Exemple 1

- Dans cet exemple, le code est intégré dans le fichier HTML grâce à la balise `<script>`
- Code interprété par le navigateur (client)
- *Output* (sortie) = page web affichée
- Notation objet .
- Le ; à la fin de chaque instruction

Exemple 2

- Variable et affectation
- Éviter les *popup*…
- La **console** pour tester
- Les commentaires

Exemple 3

- Du code js peut être intégré dans la valeur de l'attribut `onclick`
- Programmation évènementielle
- Améliorations : fonction, intégration du résultat dans la page et séparation complète HTML/JavaScript

Exemple 4

- Des fonctions comme en python
- La balise <script> peut être placée n’importe où mais attention...
- Améliorations : ~~fonction~~, intégration du résultat dans la page et séparation complète HTML/JavaScript

Exemple 5

- Améliorations : ~~fonction~~, ~~intégration du résultat dans la page~~ et séparation complète HTML/JavaScript

Exemple 6

- Attention aux conflits `id="fact"` et function `fact`
- Fichier *.js* (fichier texte) avec le code sans la balise `<script>`
- Lien entre le fichier HTML et le fichier JS
  `<script src="cours3-6.js"></script>`
- Voir ce qui se passe lorsque je déplace la balise `<script>` dans le head...

Hello World

Python

```
print("Hello world")
```

Écrit avec *IDLE*

Interprété par *IDLE*

Et affiché sur *IDLE*

Javascript

```
document.write("Hello world");
```

Écrit avec un éditeur de texte

Interprété par navigateur

Et affiché sur la page web

Commentaires

Python

```
""" commentaires
sur plusieurs
lignes """

# sur une ligne
```

JavaScript

```
/* commentaires
sur plusieurs
lignes */

// sur une ligne
```

Variables et affectation

Python

```
a = 3
x = 2.1
ch = "python"
test = False
```

JavaScript

```
var a = 3;
var x = 2.1;
var ch = "JavaScript";
var test = false;
const a = 3;
let a= 3;
```

Tests conditionnels

Python

```
note = 8

if note>10 :
   res = "admis"
elif note>=8
   res = "rattrapage"
else
   res = "non admis"
```

JavaScript

```
var note = 8;
var res;

if (note>10) {
   res = "admis";
} else if (note>=8) {
   res = "rattrapage";
} else {
   res = "non admis";
}
```

Opérateurs de comparaison et opérateurs logique

==	==
!=	!=
not	!
and	&&
or	||

Boucle for

Python

```
Somme = 0
for i in range(1,10) :
   somme = somme + i
```

JavaScript

```
var somme = 0;
for (var i=1; i<=10; i=i+1){
   somme = somme + i;
}
```

Boucle while

Python

```
j = 1
n =10
fact = 1

While j<=n :

  j = j+1

  fact = fact*j
```

JavaScript

```
var j=1;

var n=10;

var fact = 1;


while (j<=n) {

  j = j+1;

  fact = fact * j;

}
```

Fonction

Python

```
def fact(n) :
  res=1
  for i in range(2,n) :
    res = res*i
  return res
```

JavaScript

```
function fact(n){
  var res=1;
  for (var i=2; i<=n; i++){
    res = res*i;
  }
  return res;
}
```

Chaînes

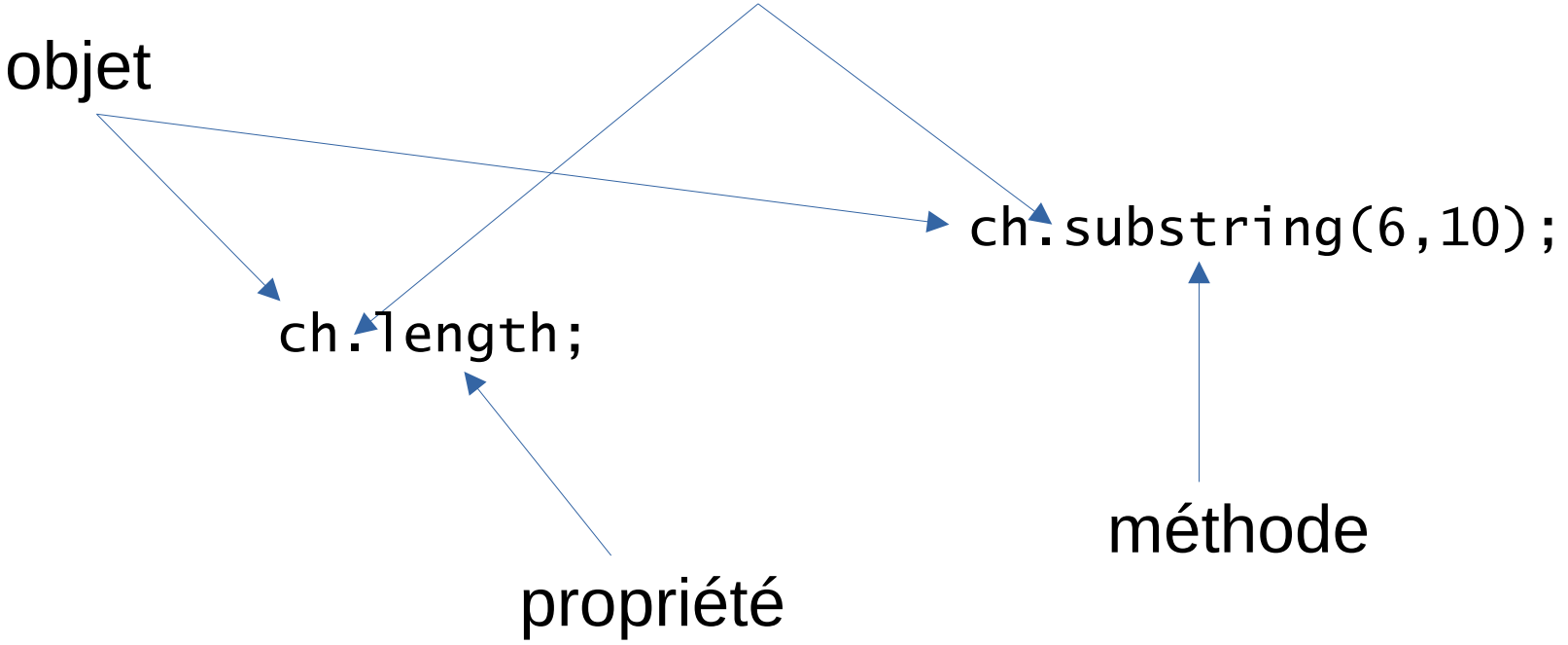
Python

```
ch = "Hello World"
ou 'Hello World'

caractère d'échappement \

chaîne sur plusieurs lignes ``

Concaténation +

len(ch)
ch[0]
ch[6:11]
```

JavaScript

```
var ch = "Hello World";
ou 'Hello World'

caractère d'échappement \

chaîne sur plusieurs lignes ``

Concaténation +

ch.length;
ch.charAt(0);
ch.substring(6,10);
```

Notation objet



Les mathématiques

Python

```
a = int('10')

1.25e2

+ - * / // %

10//3

import math as math
math.cos(math.pi)
```

JavaScript

```
var a = parseInt('10');

1.25e2

+ - * / %

Math.floor(10/3);

Math.cos(Math.PI);
```

Les tableaux

Python

```
vide = []

jours = ["lun","mar","mer","jeu","ven","sam","dim"]

len(jours)

jours[1]



Copie par référence
```

JavaScript

```
var vide = new Array();
var vide = []; //format JSON

var jours = new Array("lun","mar","mer","jeu","ven","sam","dim");

var jours = ["lun","mar","mer","jeu","ven","sam","dim"]; //format JSON

jours.length


Jours[1]


Copie par référence
```

Les dictionnaires

Python

JavaScript

Programmation évènementielle

Le JavaScript est un langage évènementiel (exemple 3) :

Détection des évènements

Exemple 3 :
clic sur le bouton

Déclenchement de nouveaux traitements

Exemple 3 :
récupération prénom + affichage

Évènements

- Clic sur un élément
- Appui d’une touche du clavier
- Chargement d’un élément
- Rotation du smartphone en mode paysage
- Etc.

Programmation événementielle

- En python, les instructions sont interprétées les unes après les autres, on parle de programmation séquentielle
- En JavaScript, lorsqu'on introduit un « onclick » dans le code, l'« écoute » ne se fait pas qu'au moment où la ligne est interprétée et le déclenchement des instructions liées à l'évènement peut se faire à n'importe quel moment, on parle de programmation évènementielle

Programmation événementielle

Programmation évènementielle

- Ordre des événements inconnu
- Survenue d’un évènement non garantie
- Évènements parasites possibles (exemples chargement d’élément critique ralenti, coupure de connexion...)

Programmation séquentielle

- Traitement parfaitement séquencés
- Traitement parfaitement prédictibles
- Traitements parasites impossibles

- Comme nous l’avons vu dans les exemples, les évènements sont ajoutés à des éléments HTML et les traitements déclenchés modifient ou créent des éléments dans la page
- Il faut donc être capable d’accéder à l’élément qui nous intéresse…

Rechercher des éléments

- `HTMLElement` **`document.getElementById`**`(String identifiant)`
- `HTMLCollection` **`document.getElementsByClassName`**`(String nomClasse)`
- `HTMLCollection` **`document.getElementsByTagName`**`(String balise)`
- `HTMLCollection` **`document.getElementsByName`**`(String nom)`
- `HTMLElement` **`document.querySelector`**`(String selecteurCSS)`
- `NodeList` **`document.querySelectorAll`**`(String selecteurCSS)`
- Accès direct ex. `document.forms`

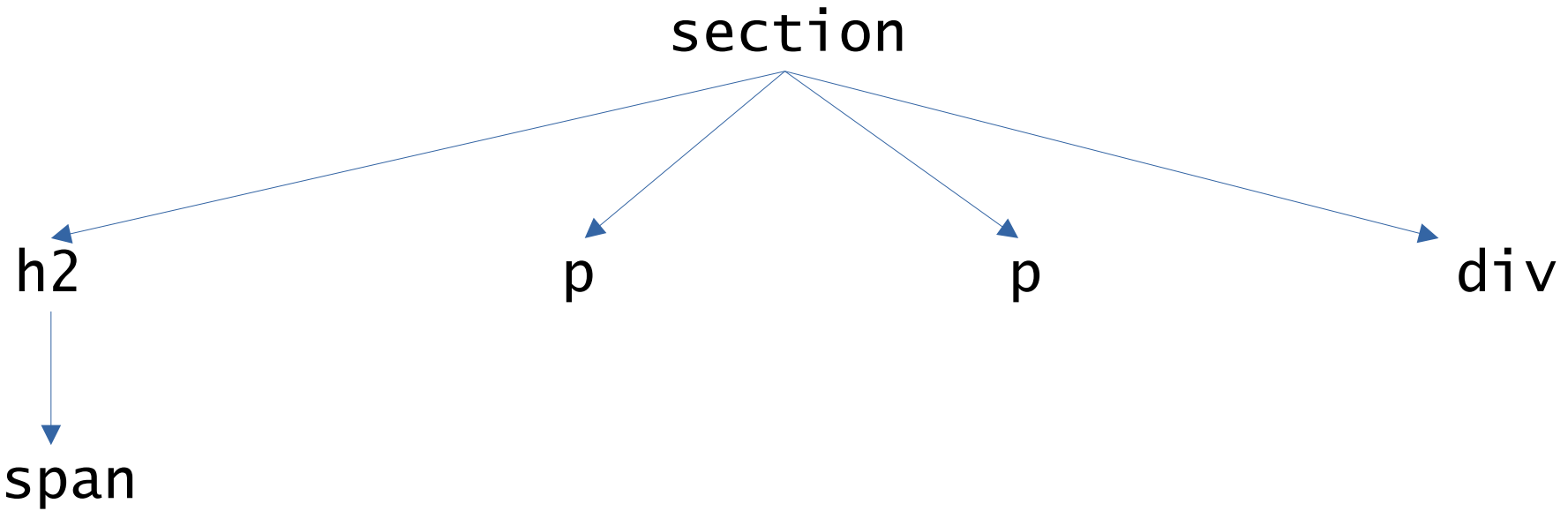
...accéder à leurs attributs et styles

- Accéder aux attributs :
  `String element.getAttribute(String nomAttribut)`
- Accéder aux styles :
  `getComputedStyle()` méthode de window qui retourne le style de l'**élément** passé en paramètre…
- Accéder à une propriété :
  Toutes les propriétés de style d'un élément sont accessibles via l'objet `style` (lecture) (attention pas de tiret : `background-color` → `backgroundColor`)

Enfants, frères et parents

- `childNodes`
- `firstElementChild` et `lastElementChild`
- `nextElementSibling` et `previousElementSibling`
- `parentElement`

Exemple 7



Le bon moment pour l’accès aux éléments

- Le JavaScript étant un langage évènementiel, l’état du document n’est pas connu à l’avance.
- Il faut donc s’assurer que le document est complètement construit pour chercher à accéder aux éléments.

Modifier les éléments

- Une fois que nous avons l’élément qui nous intéresse, nous pouvons lui ajouter un évènement ou le manipuler

Modifier le contenu

- Propriété `innerHTML` en lecture et en écriture
- `OuterHTML` et `innerText`

Modifier le style

- Toutes les propriétés de style d'un élément sont accessibles via l'objet `style` (écriture) (attention pas de tiret : `background-color` → `backgroundColor`)
- `className` à privilégier (séparation js/css)

Créer de nouveaux éléments

- `HTMLElement` **`document.createElement`**`(String typeElement)`
- Créer des attributs `String element.`**`setAttribute`**`(String nomAttribut, String valeur)`
- `parent.appendChild(HTMLElement enfant)`
- `parent.insertBefore(HTMLElement enfant, HTMLElement position)`
- `parent.removeChild(HTMLElement enfant)`

Exemple 10