Technologies avancées du *e-learning* 2

Mathieu.loiseau@univ-grenoble-alpes.fr

Objectifs du cours

- Ancrer des concepts génériques de programmation accumulés depuis 2 ans
  - Variable / Type
  - Structure de contrôle
  - Fonction
  - Objet / Classe
  - Événement
- Autres concepts
  - DOM
  - Client
  - Serveur
- **Modélisation**

Déroulement du cours

- S1
  - Rappels, exercices, nouveaux concepts
  - Spécifications du projet
- S2 ⟶ réalisation du projet
- Projet
  - Un groupe crée une librairie de gestion d'une base de questions en javascript
  - Les autres groupes utilisent cette librairie pour créer des jeux de quizz à vidéo-projeter en classe de langue (+ exemples de questions)
    - Exemples de jeux
      - Tirs aux buts
      - Désignation aléatoire de l'équipe et calcul des scores
      - Etc.

Avant de commencer

- Disponibilités la semaine prochaine pour un cours sur **Git** ?
- Qu'est-ce que vous N'avez PAS compris en JavaScript ?

Rappels du cours n°1

type de données, DOM

Algorithme et programme

- **Algorithme** (D. Knuth) : Ensemble fini de règles qui donne une suite d'opérations pour résoudre un certain type de problèmes
  - Fini: un algorithme doit toujours se terminer en un nombre fini d'étapes
  - Défini: chaque pas doit être défini de manière non ambiguë
  - Entrée: un algorithme à 0 ou plusieurs entrées issues d'ensembles spécifiés d'objets
  - Sortie: un algorithme à 0 ou plusieurs sorties (quantités en relation de façon spécifiée avec les entrées)
  - Effectivité: chaque opération doit être suffisamment basique pour être exécuté par un homme utilisant un papier et un crayon
- Un **programme** est une manière (dépendant d'un langage) de réaliser un **algorithme**.

Algorithme et programme

- **Algorithme** (D. Knuth) : Ensemble fini de règles qui donne une suite d'opérations pour résoudre un certain type de problèmes
  - Fini: un algorithme doit toujours se terminer en un nombre fini d'étapes
  - Défini: chaque pas doit être défini de manière non ambiguë
  - Entrée: un algorithme à 0 ou plusieurs entrées issues d'ensembles spécifiés d'objets
  - Sortie: un algorithme à 0 ou plusieurs sorties (quantités en relation de façon spécifiée avec les entrées)
  - Effectivité: chaque opération basique pour être exécuté par un homme avec un papier et un crayon

Ici, nous allons apprendre à *implémenter* des algorithmes en javascript.

- Un **programme** est une manière (dépendant d'un langage) de réaliser un **algorithme**.

Variable, type

- Variable : pour conserver et réutiliser une valeur
  - Le mot clé correspondant est `var`
  - Exemple :

```
var texte = "bonjour le monde !";
window.alert(texte);
window.alert("re-"+texte);
```

- Chaque variable a un type
  - Définit « ce que l'on peut faire avec »
  - Permet de détecter des erreurs
  - Allocation de l'espace mémoire
  - Parfois **explicite** et **inaltérable**, en JS il est **implicite** et **modifiable** pendant l'exécution

Les différents types en javascript

- Pour obtenir le type d'une variable : `typeof`
- `typeof nomDeLaVariable`
  - ⟶ renvoie une `string`, qui peut valoir :
    - `"undefined"` ⟶ pas de valeur
    - `"number"` ⟶ nombre
    - `"string"` ⟶ chaîne de caractères
    - `"boolean"` ⟶ booléen (vrai ou faux)
    - `"symbol"` ⟶ mot clé du langage/appel de fonction
    - `"function"` ⟶ une fonction (cf. plus loin)
    - `"object"` ⟶ n'importe quel autre objet
- Selon le type les opérateurs n'ont pas la même sémantique

Type et opérateurs

- Pour réaliser des opérations sur des variables et/ou des constantes
- En javascript, l'opération effectuée par un opérateur dépend du type des opérandes
  - Nombres : +,-,/, *,% (reste de la division entière)
  - Chaines de caractère : + ⟶ concaténation
  - Booléens :
    - && ⟶ et logique
    - || ⟶ ou logique
    - ! ⟶ non
  - Entre une variable et une expression constante :
    - = ⟶ affectation

Type et opérateurs

- Pour réaliser des opérations sur des variables et/ou des constantes
- En javascript, l'opération effectuée par un opérateur dépend du type des opérandes
  - Nombres : +,-,/, *,% (reste de la division entière)
  - Chaines de caractère : + → concaténation
  - Booléens :
    - && → et logique
    - || → ou logique
    - ! → non
  - Entre une variable et une expression constante :
    - = → affectation

```
<script type="text/javascript">
    var uneChaine, unNombre;
    uneChaine = "bonjour le";
    console.log("uneChaine:", uneChaine +
    " monde !");
    unNombre = 0;
    unNombre = unNombre + 1;
    console.log("unNombre:", "1" +
    unNombre);
</script>
```

Que va afficher la console du navigateur ? Pourquoi ?

Booléens et conditions

- Booléens ⟶ 2 valeurs
- Utilisables pour les tests
  - Instructions conditionnelles
  - Boucles
- Les autres types peuvent être l'objet de tests

  ⟶ Conversion
  - Number : `0` ⟶ `false` / `1`, `-1`, `500` ⟶ `true`
  - String : `""` ⟶ `false` / `"false"`,`"0"` ⟶ `true`

Fonctions

Et principes d'analyse descendante (en utilisant le DOM)

Fonctions

- ≈ séquence d'instructions pouvant avoir recours à des paramètres, et renvoyant une valeur calculée en fonction de ces paramètres.
- En JavaScript :
  - Définition

```
function addition(par1, par2, etc){
   //Séquence d'instructions
   /*Par exemple:*/
   var resultat = par1 + par2 + etc;
   return resultat;
}
```

  - Appel

```
console.log(addition(3,2,1)); //affiche 6
var sommeNb1a4 = 4 + addition(1,2,3);
    //renvoie une valeur ↑
```

Objets et DOM

- JS est un langage « objet »
- Définition
  - « un objet est un conteneur symbolique, qui possède sa propre existence et incorpore des **informations** et des **mécanismes**, éventuellement en rapport avec une chose tangible du monde réel, et manipulés dans un programme »
  - Informations ⟶ **Attributs**
  - Mécanismes ⟶ **Méthodes**
- Dans un navigateur, le document est représenté sous forme d'objet : Document Object Model (DOM)
  - Exemples d'appels :
    - Méthode :
      ```
      var elt = document.getElementById("idElement");
      ```
    - Attribut :
      - `console.log(elt.className);`
      - `elt.className="nomDeLaClasse à appliquer";`

Structures de contrôle : boucle « pour »

- Syntaxe

```
for(/*initialisation*/;
    /*condition d'arrêt*/;
    /*incrément ce qu'on fait à la fin de
chaque tour*/){
   /*instructions répétées tant que la condition
   d'arrêt n'est pas atteinte. Attention : si les
   instructions ne permettent pas d'obtenir la
   condition d'arrêt, le programme ne s'arrête jamais…
   */
}
```

- Exemple

```
console.log("Afficher les éléments d'un
tableau");
var tab = ["A","E","I","O","U"];
var long = tab.length ;
for(var i=0;i<long;i++/*↔ i=i+1*/){
   console.log(tab[i]);
}
```

Exercice

```html
<h2>Facile</h2>
<ul id="fastoche">
  <li>Maegan : 4, 8, 10, 12</li>
  <li>Ajay : 15, 16, 12, 16</li>
  <li>Sheri : 9, 10, 11, 15</li>
  <li>Marianne : 10, 10, 10, 10</li>
</ul>
<h2>Moins facile</h2>
<ul id = "moyen">
  <li>Gus : 4, 5.5, 9, 9.5, 10</li>
  <li>Sammy : 3, 17</li>
  <li>Belinda : 8.5, 11, 20</li>
</ul>
<h2>Très difficile</h2>
<ul id="chaud">
  <li>Brayden: 12, 14  , 8.5 ; 12 ;</li>
  <li>Kassidy : 13, 3.5 ;</li>
  <li>Jonas : 18, 19, 16.5, 10.</li>
</ul>
```

- On a un document HTML, qui contient une liste d'étudiant avec les notes obtenues lors du semestre, on veut :
  - Mettre en italiques le nom de l'élève ;
  - Mettre en gras les notes supérieures à la moyenne
  - Calculer et souligner la moyenne
- Informations utiles
  - `DOMelt.children` ⟶ renvoie un tableau avec la liste des enfants d'un élément
  - `mot.charAt(i)` ⟶ (i+1)ᵉ caractère d'une chaine
  - `chaineOuTableau.length` ⟶ nombre d'éléments
  - `Number("25")` → `25`

Objets

- Pour créer un objet à la volée, on peut utiliser la syntaxe JSON
  - { } ⟶ objet
  - [ ] ⟶ tableau
  - "attribut" : valeur ⟶ attribut

```
var monObjetComposite = {
    "attribut1":"valeur",
    "attribut2":"valeur",
    "etc":"et cætera"
};

var personne = {
    "prenom" : "John",
    "nom" : "Doe",
    "age" : 66,
    "adressesMail" : [
        {"type":"perso",
         "adresse":"jdoe@mail.com"},
        {"type":"pro",
         "adresse":"jd@anonymous.net"}
    ]
};
```

Calcul/affichage moyennes

- Corrections :
  - Étape par étape : https://codepen.io/lzbk/pen/EbLGKQ