Université Grenoble Alpes

Technologies avancées du *e-learning* 2

mathieu.loiseau@univ-grenoble-alpes.fr
http://lzbk.univ-grenoble-alpes.fr

Université Grenoble Alpes

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

- Premières séances
  - Rappels, exercices, nouveaux concepts
  - Spécifications du projet
- Fin du semestre → réalisation du projet
- Projet
  - Une librairie de gestion d'une base de questions en JavaScript est mise à votre disposition (d'une manière ou d'une autre…)
  - Vous utilisez/étendez 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.

Qu'est-ce que vous n'avez PAS compris en JavaScript ?

Variables, types de données et structures conditionnelles

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 assez basique pour être exécutée 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 doit être assez basique pour être exécutée par un homme utilisant un papier et un crayon
- Un **programme** est une manière (dépendant d'un langage) de réaliser un **algorithme**.

Ici, nous allons apprendre à écrire des **programmes** implémentant des algorithmes en JavaScript.

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 dépend du type des opérandes
  - Nombres : +,−,/,∗,% (reste de la divis
  - Chaines de caractère : + → concaté
  - Booléens :
    - && → et logique
    - || → ou logique
    - ! → non
  - Entre une variable et une expressio
    - = → 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 ?

Parce que...

Transtypage et booléens

- Booléens → Utilisés pour les tests
  - Résultat d'une comparaison
  - Instructions conditionnelles
  - Boucles
- Les autres types peuvent être les opérandes de tests d'égalité
  → Conversion
  - Number :
    - `0` → `false`
    - `1`, `-1`, `500` → `true`
  - String :
    - `""` → `false`
    - `"false"`,`"0"` → `true`

- Comparaisons
  - `==` → Égalité avec transtypage
  - `!=` → Différence avec transtypage
  - `===` → Égalité sans transtypage
  - `!==` → Différence sans transtypage
  - `>` → supérieur
  - `>=` → supérieur ou égal
  - `<` → inférieur
  - `<=` → inférieur ou égal

Tests et conditions
Instruction conditionnelle `if`

- Permet d'exécuter une séquence d'instruction différente selon le résultat d'un test logique

```
if(/*test1*/){
   //Séquence d'instructions à exécuter si test1 est vrai
}
else if(/*test2*/){
   //Séquence d'instructions à exécuter si test1 est faux et test2 est vrai
}
//…
else{
   //Séquence d'instructions à exécuter si test1 est faux et test2 est faux
}
```

Voir aussi
`switch`

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 ↑
```

- En JS, « procédures » et « fonctions » utilisent le même mot clé (`function`).
  - Par convention on appellera « une procédure » toute fonction qui n'a pas d'instruction **`return`**.

Exercice 1 : une fonction

- Créer une fonction `estChiffre` qui renvoie `true` si un caractère est un chiffre.

Tests et conditions
Instruction conditionnelle `switch`

- Quand le test porte sur une variable qui peut donner lieu à de multiples choix, on peut utiliser le mot-clé `switch`.
- Quand on entre dans un "`case`" on effectue les instructions jusqu'au prochain `break`.
- Le cas par défaut (non-couvert par tous les autres) s'appelle `default`.

```
function estChiffre(caractere){
    var resultat;
    switch(caractere.toString()){
        case "0":
        case "1":
        case "2":
        case "3":
        case "4":
        case "5":
        case "6":
        case "7":
        case "8":
        case "9":
            resultat = true;
            break ;
       default:
            resultat = false;
    }
    return resultat;
}
```

Structures de contrôle : boucle « pour »

for

- Syntaxe

```
for(/*initialisation*/;
    /*condition de continuation*/;
    /*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]);
}
```

Université Grenoble Alpes

Exercices 2 : Le 1 en boucle...

1) Créer une fonction `estNombre` qui renvoie `true` si une chaine de caractères `mot` représente nombre.
   - Données :
     - `mot.charAt(i)` → (i+1)e caractère d'une chaine
     - `mot.length` → nombre de caractères d'une chaine
2) Créer une fonction `promptNombre` qui demande à l'utilisateur d'entrer un nombre et s'il saisit un nombre le renvoie sous forme de `Number` , renvoie `false` sinon.
   - Donnée : `Number("25")` → `25`

Structures de contrôle : boucle « tant que » `while`

- Syntaxe

```
/*initialisation*/;
while(/*condition de continuation*/){
    /*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…*/
    /*incrément ce qu'on fait à la fin de chaque tour*/
}
```

Exercice 3
“While not stop”

- Créer une fonction `saisieNombres` qui demande à l’utilisateur d’entrer des nombres, n’accepte que les nombres et les stocke dans un tableau tant qu’il n’a pas entré `"stop"`, puis renvoie le tableau ainsi créé.

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";`

Université Grenoble Alpes

Quelques attributs et méthodes pour manipuler le DOM

- `DOMelt.children` → accéder à un tableau avec la liste des enfants d'un élément
- `document.createElement("tr")` → renvoie un élément tr à insérer…
- `DOMelt.src` → renvoie l'attribut src d'un élément
- `DOMelt.style` → renvoie l'attribut style d'un élément
- `DOMelt.appendChild(DOMelt2)` → ajoute l'élément `DOMelt2` à la suite des enfants de l'élément `DOMelt`
- `DOMelt.innerHTML="<tr><td>case1</td><td>case2</td></tr>";` → remplace la valeur de l'attribut `innerHTML` de l'objet `elt` par une chaine de caractère.

Exercices

- Créer une procédure `arrayToTable` qui prend en entrée un tableau de nombres (`leTableau`) et l'identifiant d'un élément `<table>` (`idTable`) et ajoute une ligne à la fin du tableau html (`idTable`) qui contient tous les nombres du tableau (`leTableau`) ;
- Créer une procédure `fillTable` qui demande à l'utilisateur de saisir des nombres et les met par ligne dans un élément `<table>` (`idTable`)
  - L'utilisateur entre les valeurs une par une
  - S'il entre autre chose qu'un nombre la valeur est ignorée
  - Quand il entre `"stop"`, on passe à la ligne suivante du tableau
  - On arrête quand il a entré `"stop"` deux fois de suite
    c'est à dire quand la ligne est …

Université Grenoble Alpes

Exercices : Saisie de notes

- Pour ces exercices, on peut partir du code précédent mais des modifications doivent être apportées
- On veut créer un tableau de notes avec pour chaque ligne, un nom et un même nombre de notes
- L'utilisateur va saisir un nombre de notes dans le trimestre
- Le système va l'aider à saisir ses notes en lui demandant pour chaque élève :
  - Un nom
  - Le bon nombre de notes
  - Si l'élève était absent, l'enseignant va saisir `"-"`
- Quand l'enseignant a fini, il appelle l'élève `"stop"` et le tableau s'affiche
- Questions subsidiaires (classées par ordre de difficulté)
  - Ajouter automatiquement une colonne à la fin de chaque ligne avec la moyenne de l'élève
  - Ajouter automatiquement une ligne à la fin du tableau avec les moyennes de classe sur chaque contrôle
  - Trier les élèves par ordre alphabétique

Exercices (supplémentaires)

La correction est en ligne

- Créer une fonction `somme` qui additionne tous les éléments d'un tableau (JS) et renvoie le résultat ;
- Créer une fonction `trier` qui prend un tableau (JS) et renvoie un tableau trié ;
- Créer une fonction `tableToArray` qui prend en entrée l'identifiant d'un élément `<table>` (`idTable`), un numéro de ligne (`idLigne`) et renvoie un tableau (JS) contenant tous les éléments de la ligne ;
- Écrire un programme (procédure) réutilisant ces *fonctions* qui trie une par une les lignes d'un élément HTML `<table>` (`idTable`), qui met à la fin de chaque ligne la somme de ses éléments et met le tout dans un autre élément `<table>` (`idTableTriee`).
  Question subsidiaire : Ignorer les lignes qui contiennent autre chose que des nombre dans une case ou plus.

Exercices (supplémentaires)

La correction est en ligne

- Créer une procédure `randomizeTable`, qui prend un élément `<table>` (`idTable`) et met dans une autre `<table>` (`idTableMelangee`), la liste des éléments de `idTable` mais dans un ordre aléatoire.
  - Méthodes utiles :
    - Aléatoire

```
var nbEntre0et1 = Math.random();//1 exclu
var nbEntre0et10 = Math.random()*10;//10 exclu
var intEntre0et10 = Math.floor(Math.random()*11);
   //10 inclus
```

      - Voir MDN pour des informations détaillées sur l'aléatoire