UNIVERSITÉ Grenoble Alpes

Technologies avancées du *e-learning* 2

Mathieu.loiseau@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**

UNIVERSITÉ Grenoble Alpes

Déroulement du cours

- S1
  - Rappels, exercices, nouveaux concepts
  - Spécifications du projet
- S2 → 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.

UNIVERSITÉ Grenoble Alpes

Avant de commencer

- Qu’est-ce que vous N’avez PAS compris en JavaScript ?

UNIVERSITÉ Grenoble Alpes

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**.

UNIVERSITÉ Grenoble Alpes

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**.

Ici, nous allons apprendre à implémenter 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

UNIVERSITÉ Grenoble Alpes

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

UNIVERSITÉ Grenoble Alpes

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

UNIVERSITÉ Grenoble Alpes

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
  - Chaines de caractère : +
  - Booléens :
    - && → et logique
    - || → ou logique
    - ! → non
  - Entre une variable et u
    - = → 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`

UNIVERSITÉ Grenoble Alpes

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 ↑
```

Exercice

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

UNIVERSITÉ Grenoble Alpes

Structures de contrôle : boucle « pour »

- 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]);
}
```

Exercices

- Créer une fonction `estNombre` qui renvoie `true` si une chaine de caractères représente nombre.
  - Donnée : `mot.charAt(i)` → (i+1)ᵉ caractère d'une chaine
- 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 »

- 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

- 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"`.

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 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 fonction `arrayToTable` qui prend en entrée un tableau de nombres (`leTableau`) et l'identifiant d'un élément `<table>` (`idTable`) et remplace le contenu de la table par une unique ligne contenant tous les nombres du tableau ;
- Créer une fonction `somme` qui additionne tous les éléments d'un tableau ;
- Créer une fonction `trier` qui prend un tableau et renvoie un tableau trié ;
- Créer une fonction `tableToArray` qui prend en entrée l'identifiant d'un élément `<table>` (`idTable`) et renvoie un tableau ;
- Écrire un programme réutilisant ces fonctions qui trie un élément `<table>` (`idTable`) tout en indiquant la somme de son contenu et le met dans un autre élément `<table>` (`idTableTriee`).

Exercices (suite)

- Créer une fonction `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

UNIVERSITÉ Grenoble Alpes

Rappels des 1ers cours

Notions de base

- Les **boucles** permettent de répéter un traitement ; elles doivent toujours définir
  - **état initial** (ex : `var i=0`) ;
  - **condition d'arrêt** ou de continuation (ex : `i<10`) ;
  - Et **incrément** (ce que l'on fait pour passer au tour de boucle suivant, ex : `i++`)
- Les **fonctions** permettent de nommer une succession d'instructions
  - On peut ainsi les réutiliser à plusieurs endroits sans réécrire le code (pratique + maintenable)
  - Elles peuvent manipuler des données (paramètres)
  - Elles peuvent renvoyer une valeur avec `return` (fonctions) ou non (procédures)

UNIVERSITÉ Grenoble Alpes

Rappels des 1ers cours

Principes d'analyse descendante

- Grâce aux **fonctions**, on va pouvoir organiser le code de manière à décomposer un problème complexe en une série de problèmes plus simples
- Exemple : mélanger un tableau

  `melanger(tab)` peut s'écrire à partir de

  `tireAuSortEntre0Et(max)` → Tire au sort un nombre compris entre 0 et max

  `tab.splice(id,1)` → Supprime la cas d'id `id` d'un tableau

UNIVERSITÉ Grenoble Alpes

Rappels des 1ers cours

Principes d'analyse descendante

- Grâce aux **fonctions**, on va pouvoir organiser le code de manière à décomposer un problème complexe en une série de problèmes plus simples
- Exemple : mélanger un tableau
  - `melanger(tab)` peut s'écrire à partir de
    - `tireAuSortEntre0Et(max)` → Tire au sort un nombre compris entre 0 et max
    - `tab.splice(id,1)` → Supprime la cas d'id `id` d'un tableau

UNIVERSITÉ Grenoble Alpes

Rappels des 1ers cours

Objets

- Un **objet** permet d'encapsuler dans une même structure
  - des données (les **attributs**)
  - et des traitements (les **méthodes**).
  - On accède aux attributs et aux méthodes en utilisant un '.' ; Par exemple :
    - création d'un objet `elt` en utilisant la méthode `getElementById` de l'objet `document`

      ```
      var elt = document.getElementById("lId");
      ```

    - Remplacement de l'attribut `innerHTML` de l'objet `elt` par une chaine de caractère

      ```
      elt.innerHTML="<h1>Mon TITRE</h1>";
      ```

UNIVERSITÉ Grenoble Alpes

Les objets, outil de modélisation

- Manipulés notamment avec le DOM
- Manière de penser son système et d'organiser son code
- Complémentaire avec l'analyse descendante
- ...qui peut s'appliquer à des objets de la vie quotidienne
  - Une télécommande ?
  - Un jeu de carte ? (♠,♥,♦,♣)

Objets : Notation JSON

- 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"}
    ]
};
```

UNIVERSITÉ Grenoble Alpes

Exemples de méthodes JS utiles...

- ```
  var mot = "Exemple";
  var tabDeMots = ["Test", mot, "Démo", "Illustration"] ;
  ```
- Consultation d'un caractère d'une chaine de caractère donnée

  ```
  //Consultation des caractères d'une chaine de cara
  var initiale = mot.charAt(1);//"x"
  ```
- Recherche d'une sous-chaine dans une chaine

  ```
  var posMP = mot.indexOf("mp");//"3"
  var posZ = mot.indexOf("z");//"-1"
  ```
- Recherche d'une valeur dans un tableau

  ```
  var posEx = tabDeMots.indexOf("Démo");//"2"
  var posZZ = tabDeMots.indexOf("zz");//"-1"
  ```
  ```

Créer ses propres classes

- Exemple de déclaration de classe

```
class Carte{/*pour un memory*/
    constructor(image, id){
        this.img = image;
        this.id = id;
    }
    creerHTML(){
        return "<article id='carte"+this.id+"'><section class='recto'><img
src='"+this.img+"' /></section><section class='verso'></section></article>";
    }
}
```

- Exemples d'instanciation de la classe

```
var i = 0 , carte1 = new Carte("raton.png", i);
i++ ;
var carte2 = new Carte("chien.png", i);
var newElt = document.createElement("div");
newElt.className = "ligne";
newElt.innerHTML = carte1.creerHTML() + carte2.creerHTML();
```

Exercice

- Concevoir et créer une classe `Equipe`, qui permette de lui donner un nom, un logo et d'en recenser les membres…
- Transformer la classe pour que ses attributs manipulent directement le DOM …

DOM et événements (1)

- Selon les actions du navigateur, des **événements** peuvent se déclencher
- C'est toujours un objet du DOM qui déclenche l'événement
- Exemples d'événements :
  - Chargement (de la fenêtre...) ;
  - Clic de souris (sur un élément) ;
  - ...
- **Capturer un événement**, c'est demander au navigateur d'exécuter une instruction, quand l'événement est déclenché

DOM et événements (2)

- Pour exécuter un traitement au chargement de la page, on utilisera `onload`.
- Depuis le code HTML (solution à éviter) :

```
<body onLoad="instructions javascript à éxecuter quand le body est chargé">
    Contenu du body
</script>
```

- En manipulant le DOM (solution à privilégier) :

```
window.onload = uneFonction;
```

  - Attention en `onload` est une méthode (fonction associée à une classe d'objets), l'interpréteur attend donc à droite de l'opérateur d'affectation une fonction et non une instruction.

```
window.addEventListener("nom de l'événement", uneFonction);
```

  - Quand `uneFonction` sera appelée, l'événement qui l'a déclenchée lui sera fourni en paramètre.

Événement onload

- Pour exécuter un traitement au chargement de la page, on utilisera `onload`.
- Depuis le code HTML :

```
<body onLoad="instructions JS à éxecuter quand le body est chargé">
  Contenu du body
</script>
```

- En manipulant le DOM (solution à privilégier) :

```
window.onload = fonction;
```

  - Attention `onload` est une méthode (fonction associée à une classe d'objets), l'interpréteur attend donc à gauche de l'opérateur d'affectation une fonction et non une instruction.

Exercice

- Créer un bouton d'`id` `AjouterMembres` ;
- Créer une fonction `addMembers`, qui accède à la variable globale `monEquipe` et qui déclenche un prompt pour ajouter des membres tant que l'utilisateur n'a pas appuyé sur annuler ou saisi `"stop"`.
- Associer la fonction `addMembers` à l'événement `onclick` de l'élément `AjouterMembres`.

Portée d'une variable

- Le mot-clé `var` définit
  - des variables visibles dans tout le code si elle n'ont pas été définies dans une fonction : on parle de variable **globale** ;
  - Des variables accessibles uniquement dans la fonction qui les déclare.
- Un bloc est une portion de code délimitée par une accolade ouvrante et une accolade fermante
- Le mot-clé `let` définit
  - Une variable qui n'est ***visible***, que dans le bloc où elle a été déclarée.
  - Si un bloc A contient un bloc B, les variables définies dans A peuvent être manipulées dans B. L'inverse n'est pas vrai.

Polysémie de `this`

- Utilisé dans la déclaration d'une méthode d'une classe, `this` désigne l'instance de l'objet manipulée.
- Utilisé dans une fonction déclenchée par la capture d'un événement, `this` désigne l'élément du DOM qui a déclenché l'événement.

Exercice

- À l'aide de la méthode `remove()` qui permet de supprimer un objet du DOM, modifier `ajoutMembre` pour qu'un clic sur un membre de l'équipe permette de le supprimer (attention au rôle de `this` dans la fonction à associer à l'événement `onclick`).