JS Introduction

Histoire de JavaScript

- Histoire de JS avec les pages web
  - Javascript est un langage de programmation qui a été conçu dés le départ pour le Web et plus spécifiquement pour donner vie aux pages html
  - Un programme JavaScript, appelé un Script, peut s'écrire directement dans une page html
  - Le navigateur est alors responsable d'interpréter les commandes du Script afin d'interagir avec le contenu HTML sans passer par un serveur

Maintenant

- JavaScript Maintenant
  - Aujourd'hui l'usage de Javascript n'est plus réservé aux pages HTML
  - Il devient un langage de programmation a part entière qui peut aussi s'exécuter sur une machine virtuelle indépendamment du navigateur
  - Il existe plusieurs machines virtuelles pour interpréter le JS
    - La machine V8 de google
    - La machine SpiderMonkey de Firefox

Le nom JavaScript

- Relation avec Java lors de la création
  - C'est la société Netscape qui fut la première à sortir un langage de programmation pour les pages html nommé: LiveScript
  - Netscapet et Sun travaillaient également sur un langage de programmation pour Internet : Java
  - Un accord entre Sun et Netscape a permis d'utiliser le nom JavaScript au lieu de LiveScript
  - Attention: Techniquement il n'existe aucun lien entre Java et Javascript
- Les spécifications ECMAS
  - ECMAScript est le nom officiel des spécifications du langage JS
  - L'organisation ECMAS adopte les différents standards sous forme de documents de spécification
  - Les spécifications ECMAScript sont publiées sous forme d'Editions
  - Les éditions les plus utilisées aujourd'hui sont: ES5, ES6 et ES7
  - Attention un navigateur peut être compatible qu'avec certaines éditions de ECMAScript

Le rôle de la VM

- La VM est responsable de l'interprétation des instructions Javascript
- Elle offre aussi un environnement 'runtime' pour accéder aux ressource externes comme l'arbre DOM ou les ressources du système d'exploitation avec node.js

Langages sur du JS

- Nous voyons aujourd'hui apparaitre des langages de programmation qui sont *transcrits* vers le langage Javascript
- Cette démarche cherche à combler certaines lacunes de JS comme de le manque de typage statique.
- Il est cependant très recommandé d'avoir une expérience avec JS pour bénéficier pleinement des avantages de ces langages
- Voici des exemples de langages:
  - Typescript
  - Flow de Facebook
  - Dart de Google

IDEs

- Programmer en JS c'est avant tout écrire des scripts dans des fichiers texte
- Un IDE facilitera plus ou moins cette tâche avec des services comme la coloration des instructions et l'autocomplition
- Pour programmer vos scripts vous pouvez utiliser les IDEs suivants
  - Visual Studio Code
  - Sublime Text
  - Atom

Comment ouvrir la console

- il est souvent utile de pouvoir écrire afficher un contenu pour tracer l'exécution de son script
- En JS, nous allons écrire nos messages sur la *console*.
- La console est cachée par défault et il faut la rendre visible sur les navigateurs pour accéder a son contenu
- Vous pouvez accéder à la console avec la touche F12
  - Chrome, F12
  - Firefox, F12

Un premier programme

- Comme le veut la tradition, nous commençons l'apprentissage du langage JS en affichant un message 'Hello World'
- Pour cela il suffit d'utiliser l'instruction suivante
  - `console.log('Hello World!')`

La console

- Ouvrez ensuite la console de votre navigateur pour voir le message affiche
- *console* est un objet disponible dans l'environnement d'execution ou le runtime
- l'objet console offre une méthode log
- Cette méthode prend en paramètre un string pour l'afficher

Les instructions

- Un script JS s'organise comme une suite ou séquence d'instructions
- Chaque instruction peut être écrite sur une seule ligne
- Nous pouvons séparer les instructions en utilisant un point-virgule ';'
- L'utilisation des points-virgules n'est pas obligatoire en JS pour séparer les instructions écrites sur des lignes différentes

Les commentaires

- Les commentaires sont utiles pour documenter votre code et le rendre ainsi plus maintenable
- Il existe deux methodes pour créer des commentaires en JS
  - Commentaire sur une seule ligne avec //
  - Un bloc commentaire avec /*....*/

ECMAS

- Les différentes évolutions des spécification ECMAScript ne garantissent pas une rétro-compatibilité
- Pour pouvoir utiliser les fonctionnalités des anciennes version ECMAS vous devez explicitement le mentionner avec l'utilisation du mode 'strict'
- La commande 'use strict' doit obligatoirement être votre première instruction du script
- Il est conseillé de toujours se mettre dans le cadre du mode 'strict'
- Tous vos scripts doivent commencer par 'use 'strict';'

C'est quoi une variable ?

- Une variable JS c'est un nom ou label que nous donnons pour referencer une valeur
- C'est donc une clé qui permet d'accéder à une valeur

Let

- Pour déclarer une variable et la faire exister dans un environnement lexical, il faut utiliser le mot 'let'

var

- Vous pouvez aussi rencontrer le mot clé 'var' pour déclarer une variable
- Attention il existe une différence entre 'let' et 'var'
- let va créer la variable dans l'environnement local des variables
- var va creer la variable dans l'environnement global des variables

Nom de variable

- Un nom de variable JS
  - Peut contenir des caractères, chiffres et les symboles $ et _
  - Ne doit pas commencer avec un chiffre
- Le nom de variable doit être différents des mots clés du langage: let, var, class, function

Les constantes

- Une constante est une variable qui accepte une unique affectation

Les types de JS

- Dans JS nous avons les types suivants
- types simples
  - number
  - string
  - boolean
  - undefined
  - symbol
- types complexes
  - object
  - function

number

- Le type number est utilisé pour les valeurs numériques
- Les valeurs NaN et Infinity font partie également des valeurs de ce type

string

- Les valeurs du type string sont les chaines de caractères
- Pour créer une chaine vous pouvez:
  - "Test"
  - 'Test'
  - `Test`
- Dans le cas des backquote, nous pouvons utiliser ${ } pour évaluer une expression JS
  - `Trois = ${2+1}` va devenir "Trois = 3"

boolean

- Le type boolean accepte deux valeurs true et false

undefined

- Le type 'undefined' contient une seule valeur 'undefined'

symbol

- Les valeurs peuvent être utillisées comme identifiant uniques

object

- object est le type des objets en JS qui sont considérés comme des tableaux associatifs clé-valeur

function

- function est associé aux valeurs fonctionnelles

L'opérateur typeof

- typeof va donner le type associé à une valeur JS

Comment convertir les types ?

- Nous allons utiliser des fonctions pour convertir une valeur vers un autre type
- String ( ) : convertir en string
- Number ( ): convertir en number
- Boolean (): convertir en booléen

Opérateurs de comparaison

- Vous pouvez utiliser les opérateurs classiques
  - x < y , y > x
  - x >= y ; y <= x
  - x != y
- Attention à l'égalité
  - x == y : égalité de valeur sans vérifier le type
  - x === y : égalité stricte avec égalité des types
- Exemple
  - `0 == false // vrai`
    `0 === false //faux`

if

- if (condition) code
- if ( ) else
- if () else if ()

opérateur ?

- let v = ( ) ? valTrue : valFalse

Les opérateurs logiques sont

- || (ou)
- && (et)
- ! (not)

Boucle while

- ```
  while(  )
  {

  }

  do {

  }while(  )
  ```
  ```

Boucle for

- `for( debut ; condition ; etape ) {`

  `}`

Contrôle dans une boucle

- Pour 'casser' la boucle, nous utilisons 'break'
- Pour continuer l'étape suivante: 'continue'

Syntaxe pour switch

- ```
  switch( exp  )  {
  case val1 :


  break;
  case val2 :


  break;
  default:

  }
  ```
  ```

Déclarer une fonction nommée

- ```
  function hello(){

  }
  ```
  ```

Déclarer une fonction

- ```
  let hello = function () {

  }
  ```
  ```

Déclarer une fonction arrow

- ```
  let hello =  () => {
  }
  let sum = (x,y) => x + y
  ```
  ```