ENSEIRB-Matmeca
2024-2025

1ère année - Filière Info
PG104

TD n°1 - Présentation de Ecmascript

Exercice 1: Mise en place de l'environnement Javascript

Le moteur d'exécution de Javascript que nous allons utiliser dans ces TDs se nomme Node.js (cf. `https://nodejs.org`). Il consiste en un exécutable `node`, qui peut :

— soit être utilisé comme boucle d'interaction, en tapant simplement la commande `node`,

— soit interpréter un fichier de code `file.js` donné en tapant la commande `node file.js`.

Il est possible de vérifier la version de `node` installée sur votre machine avec la commande `node --version`. Les versions les plus récentes ont le numéro de version majeur 24.

▷ Sur les machines de l'Enseirb-Matmeca, la commande de `node` accessible pas défaut correspond à la version `20.9.0`. Ce sera la version minimale considérée pour les TDs et les projets. Si vous n'avez pas une version suffisamment à jour sur votre machine personnelle, il faudra penser à en installer une [a].

a. Le site `https://www.ubuntuupdates.org/ppa/node_20` peut aider.

Il existe différentes manières plus ou moins raffinées d'éditer des fichiers Javascript, la plus simple étant certainement d'utiliser votre éditeur favori et d'exécuter le code dans un terminal [1].

1. Ouvrir le fichier `src/1td/factorial.js` (dans le dépôt) dans un éditeur sur votre machine. Il contient entre autres la fonction suivante :

```
function factorial(n) {
    if (n <= 1)
        return 1;
    else
        return n * factorial(n-1);
}
```

2. Exécuter le fichier précédent avec la commande `node`. Ajouter un appel à la fonction `console.log` afin d'afficher un calcul de la fonction `factorial`, et exécuter à nouveau.
3. Que se passe t'il lorsqu'on calcule `factorial("4")` ?
4. Ouvrir le fichier `test/1td/test_factorial.js`, et remarquer les lignes avec les directives `import`. Faites le nécessaire pour pouvoir exécuter ce fichier.
5. L'expression `` `factorial(5) === ${factorial(5)} : ${aResult}` `` est utilisée pour afficher le résultat du test (noter les caractères "backquote" `` ` `` (AltGr + 7 sur un clavier AZERTY)). Au vu de l'exécution du fichier, comment expliquez-vous son fonctionnement ?

1. Vous êtes d'ailleurs prié de ne pas procéder à des installations complexes *pendant* le TD.

▷ Pour cet enseignement, on pourra privilégier l'éditeur `emacs` en mode Javascript. Il se place dans le mode approprié en ouvrant un fichier `.js`. Au besoin, utiliser la commande `M-x javascript-mode` pour activer le mode Javascript. Pour éviter d'alterner en permanence entre l'éditeur et un terminal, il est possible d'utiliser dans `emacs` la commande `M-x compile` qui exécute par défaut un `make`, mais qu'il est très simple de remplacer par la commande que l'on souhaite, comme par exemple `node td1.js`.

Exercice 2: Quelques expérimentations avec `npm`

Avec le moteur Node.js est livré un gestionnaire de paquetages extrêmement populaire nommé `npm` (cf. `https://www.npmjs.com`). Ce gestionnaire permet d'ajouter avec facilité d'autres bibliothèques Javascript ainsi que des utilitaires plus ou moins pratiques.

1. Installer le paquetage calc avec la commande `npm install calc`. Cette commande va télécharger un ensemble de paquetages placés dans le sous-répertoire `node_modules`.

Lorsqu'un paquetage est installé, il ajoute parfois avec lui un ensemble d'exécutables. Noter que `npm` est connu pour comporter des paquetages avec des failles de sécurité, et donc qu'il faut faire attention en exécutant du code téléchargé automatiquement. Les exécutables installés se situent dans le sous-répertoire `./node_modules/.bin`.

2. Exécuter la commande `./node_modules/.bin/calc "3*5/7"`.

Remarque : la commande précédente est donnée à titre pédagogique, pour vérifier où les fichiers sont installés sur le disque.

3. Exécuter la commande `npx calc "3*5/7"`.

▷ On veillera à respecter aux mieux des standards de codage lorsque l'on écrit du code Javascript. Il n'existe pas de standard absolu en Javascript, néanmoins il est fortement recommandé de garder les mêmes conventions dans un code donné. La page de Mozilla `https://developer.mozilla.org/en-US/docs/MDN/Guidelines/Code_guidelines/JavaScript` fournit un ensemble de recommandations sensé.

Le compilateur Typescript peut être installé à l'aide de `npm`, et nous allons compiler un fichier d'exemple pour expérimenter avec :

4. Installer le paquetage typescript avec la commande `npm install typescript`.
5. Copier le code suivant dans un fichier `factorial_with_types.ts` :

```
function factorial(n: number) : number {
    if (n <= 1)
        return 1;
    else
        return n * factorial(n-1);
}

console.log(factorial("5"));
```

Pour compiler un fichier avec typescript, on peut utiliser le compilateur `tsc` de la manière suivante :

`npx tsc ⟨file.ts⟩ --outFile ⟨other.js⟩`

Cette compilation *ignore* les fichiers de configuration. La compilation de ce fichier va créer par défaut un fichier du même nom avec l'extension `.js`.

⚠ Par défaut, il n'y a *aucun* avertissement lors de la création de ce fichier si un fichier de ce nom existe déjà. *You have been warned.*

6. Compiler le fichier précédent
7. Corriger l'erreur de type, le recompiler, et exécuter le fichier `.js` obtenu avec `node`.

Exercice 3: Quelques fonctions simples

▷ Dans les cours et les tds, Javascript est utilisé en mode strict (`node --use_strict`) et toutes les variables doivent impérativement être définies avec `let` ou `const`.

Écrire, documenter et tester les fonctions suivantes :

- `square`, qui calcule le carré d'un nombre passé en paramètre,
- `discriminant`, qui calcule le discriminant $b^2 - 4ac$ d'un trinôme $ax^2 + bx + c$ donné par ses coefficients a, b et c,
- `evalQuadratic`, qui évalue un trinôme $ax^2 + bx + c$ donné par ses coefficients a, b et c en un point x donné,
- `caracQuadratic` qui, étant donné un trinôme $ax^2 + bx + c$, indique si le trinôme aura 2 racines réelles, 1 racine réelle ou 2 racines complexes,
- `roots` qui, étant donné un trinôme $ax^2 + bx + c$, calcule la liste des racines du trinôme.

Exercice 4: Les chaînes de caractères en Ecmascript

La spécification du langage Ecmascript spécifie très précisément les types de données manipulables dans le langage. Ici, nous allons en manipuler quelques un pour se faire la main, en commençant par les chaînes de caractères :

1. Compter sur la page `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String` le nombre de méthodes pour un objet de la classe `String`.
2. Utiliser la méthode `includes` pour tester si une chaîne en inclut une autre.

Exercice 5: Les nombres en Ecmascript

Les valeurs numériques en Ecmascript possèdent largement moins de méthodes que les chaînes de caractères. Néanmoins,quelques comportements propres au langage sont remarquables.

1. L'opérateur `**` permet de faire des exponentiations de nombre. Quelle est la plus grande puissance de 2 que l'on peut représenter de manière exacte en Javascript ? Quel est le plus grand entier que l'on peut représenter ?
2. Construire le grand entier `const twobig = 2n`, et vérifier que sur ces valeurs, il est possible de calculer bien plus loin qu'avec les nombres précédents.
3. Que se passe t'il lorsqu'on essaie d'additionner `twobig` avec un (petit) entier ? Avec une chaîne de caractères ?

Exercice 6: Bases numériques (style impératif)

▷ Rappel : toutes les variables doivent impérativement être définies avec `let` ou `const`.

1. Écrire une fonction `convert` utilisant un style *impératif* (i.e utilisant une boucle) et transformant un nombre entier en la chaîne de caractères de sa représentation binaire.

   Exemple :

```
convert(0) ;;   // → ""
convert(1) ;;   // → "1"
convert(666) ;; // → "1010011010"
```

2. Étendre cette fonction pour qu'elle fonctionne en n'importe quelle base ≤ 9. Pour les plus aventureux, gérer les bases ≤ 35 (l'utilisation de la méthode `charCodeAt()` pourrait servir).

   Exemple :

```
convert2Base(0, 2);    // → ""
convert2Base(1, 2);    // → "1"
convert2Base(666, 2);  // → "1010011010"
convert2Base(666, 3);  // → "220200"
convert2Base(666, 16); // → "29A"
```

3. Pour tester votre fonction, écrire la fonction `unconvert` qui prend une chaîne de caractères et une base et calcule la valeur entière correspondante.