Cours 3 : Langage  et intro à 

Applications web et mobiles

Christophe Gonzales

Javascript 

Exécution de javascript

- Browsers contiennent un moteur Javascript :

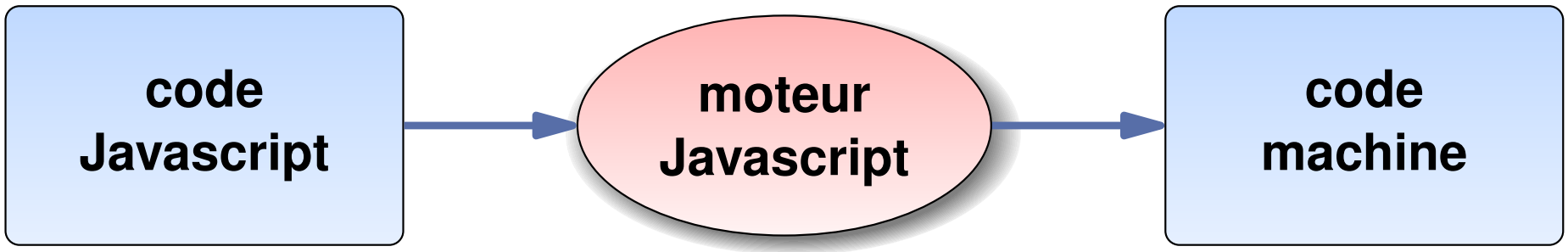

Chakra


Nitro


SpiderMonkey

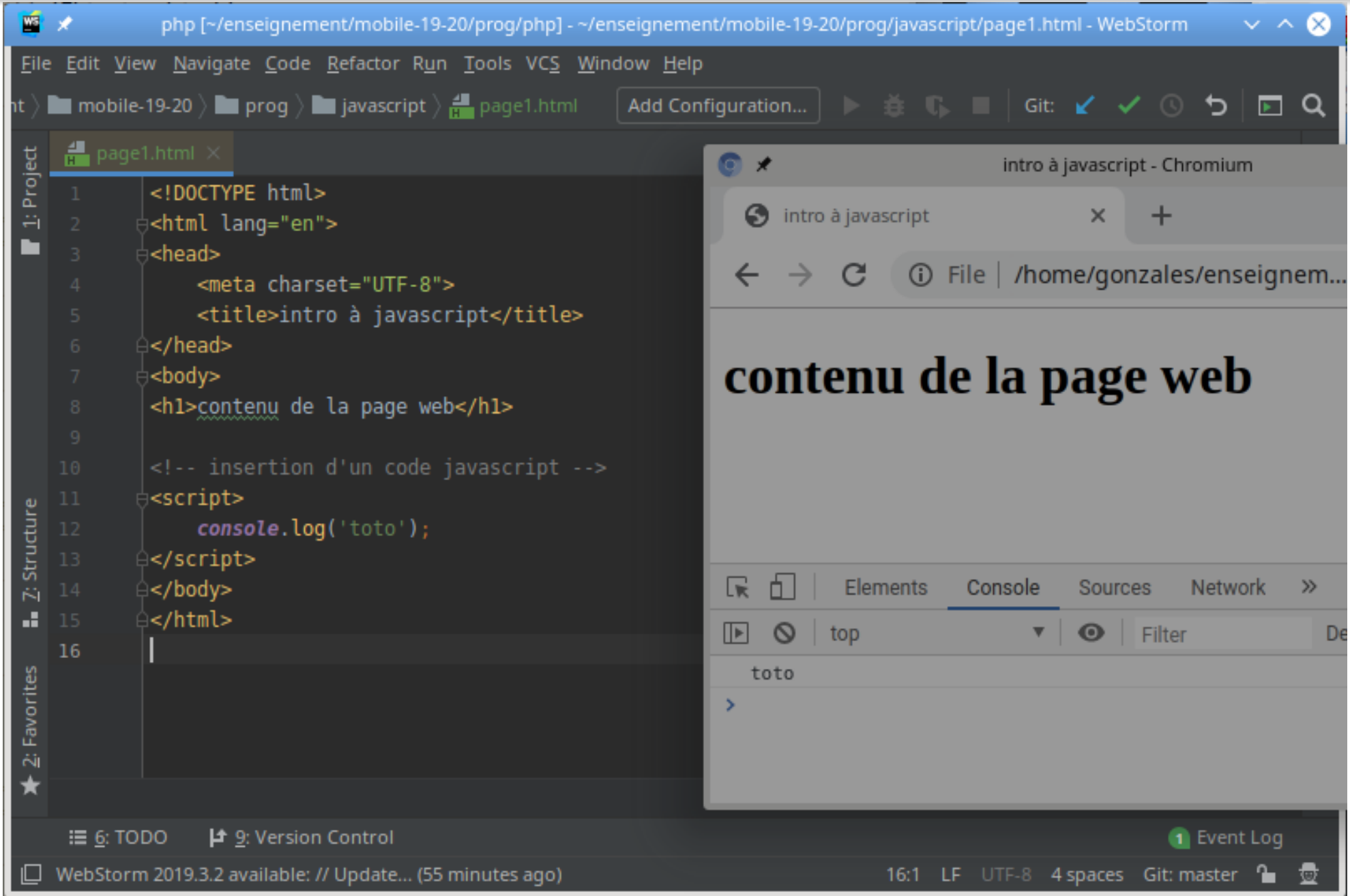

V8

- Vérifient les normes ECMAScript (ES7 = ES2016) et la plupart sont compatibles avec WebAssembly
- Principe de fonctionnement :

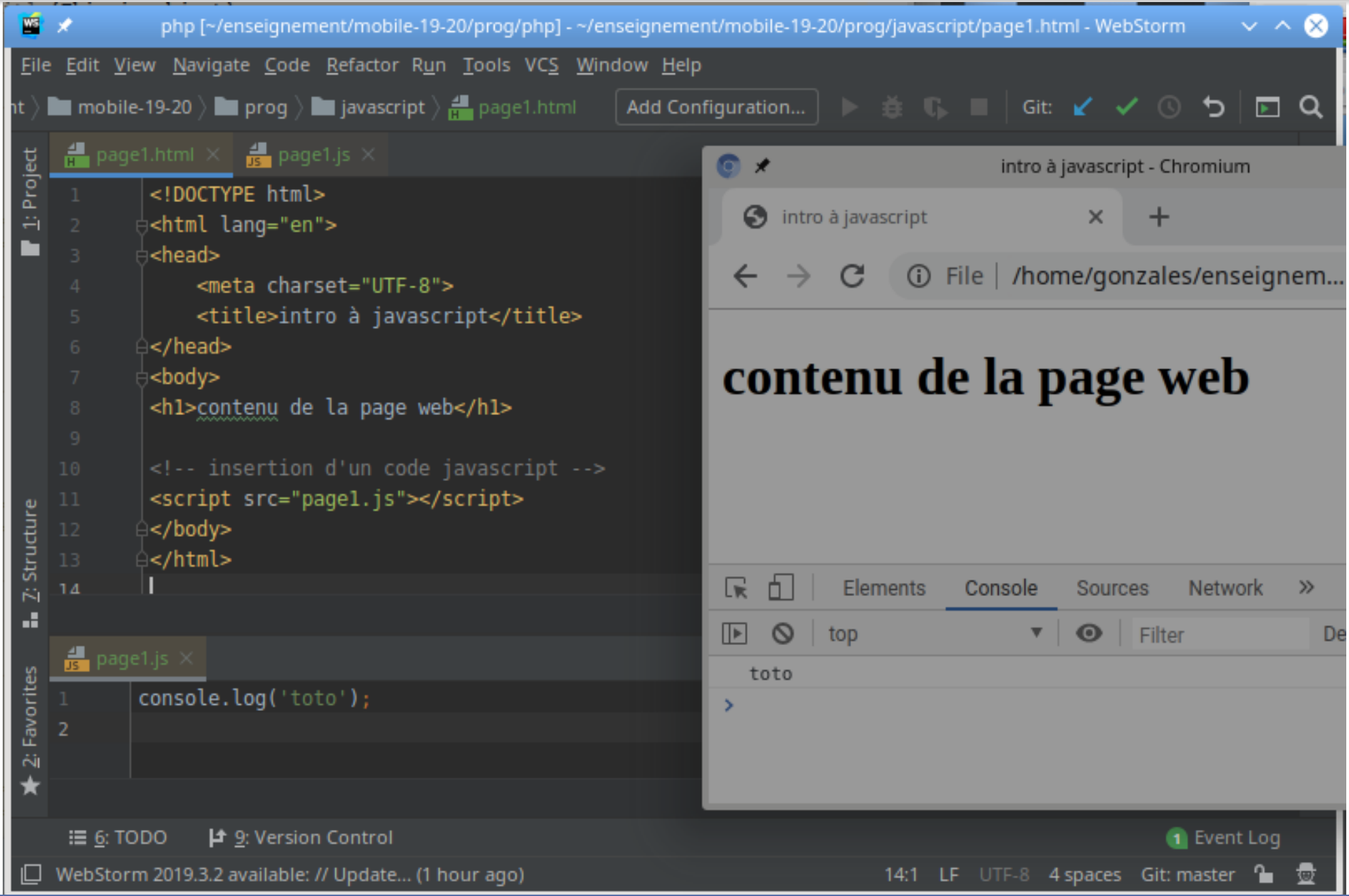


- Browser exécute ce code machine
- Depuis 2009 : compil/exec hors browser :

HTML et javascript



Inclusion d'un fichier javascript



Création de variables en javascript : 4 manières

✔ `let nom_variable = valeur;`
`const nom_variable = valeur;`
portée de bloc si déclaration dans un bloc
portée de fichier/module sinon

~ `global.nom_variable = valeur;`
~ `window.nom_variable = valeur;`
création/modification d'une variable globale

✖ `nom_variable = valeur;`
modification de la valeur d'une variable si elle existe déjà
création d'une variable globale sinon

✖ `var nom_variable = valeur;`
portée de fonction si déclaration dans une fonction
variable globale si déclaration hors fonction

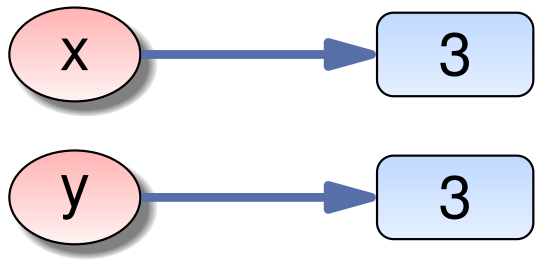
Types des variables (1/2)

Types primitifs :

- string : `let x = 'toto', y = 'to' + 'to';`
- number : `let x = 30, y = 4.5;`
- boolean : `let x = true;`
- undefined : `let x, y = undefined;`
- null : `let x = null;`

Types primitifs $\Longrightarrow$ copie par valeur :

```
let x = 3;
let y = x;
```



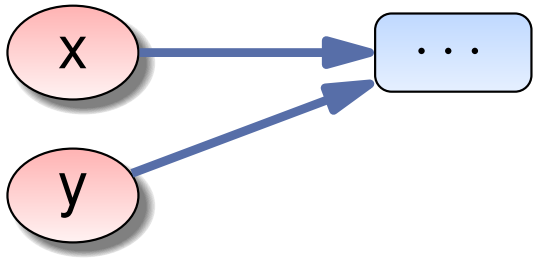
Types des variables (2/2)

Types référence :

- Object
- Array
- Function

Types référence $\Longrightarrow$ copie par référence :

```
let x = ...;
let y = x;
```



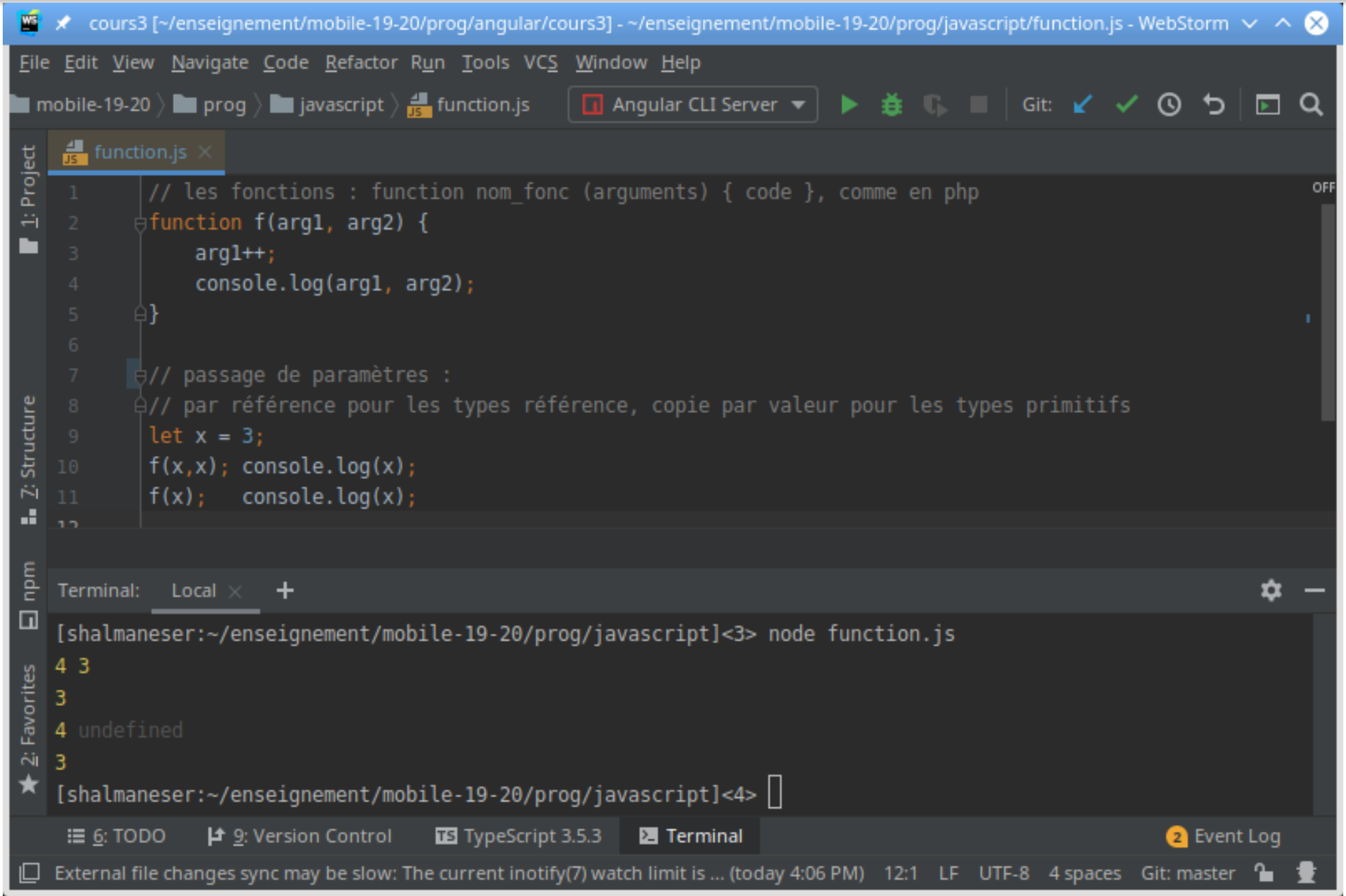
Noms des variables

- **Noms autorisés :** mêmes règles qu'en Java ou C
- **Convention :** notation Camel

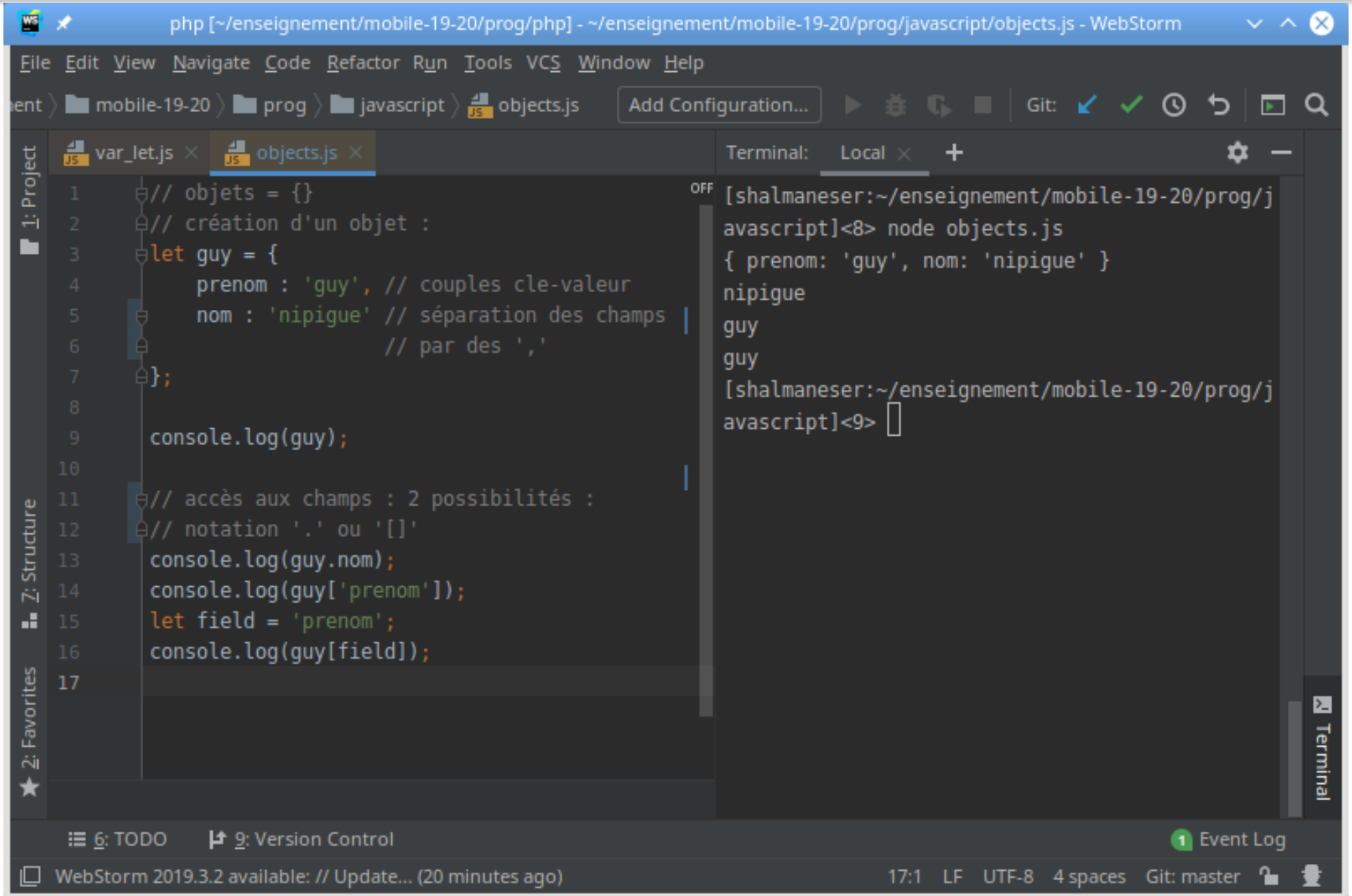
  Exemple : `firstName`
- **Attention :** Javascript sensible à la casse :

  Exemple : `firstName` $\neq$ `FirstName`

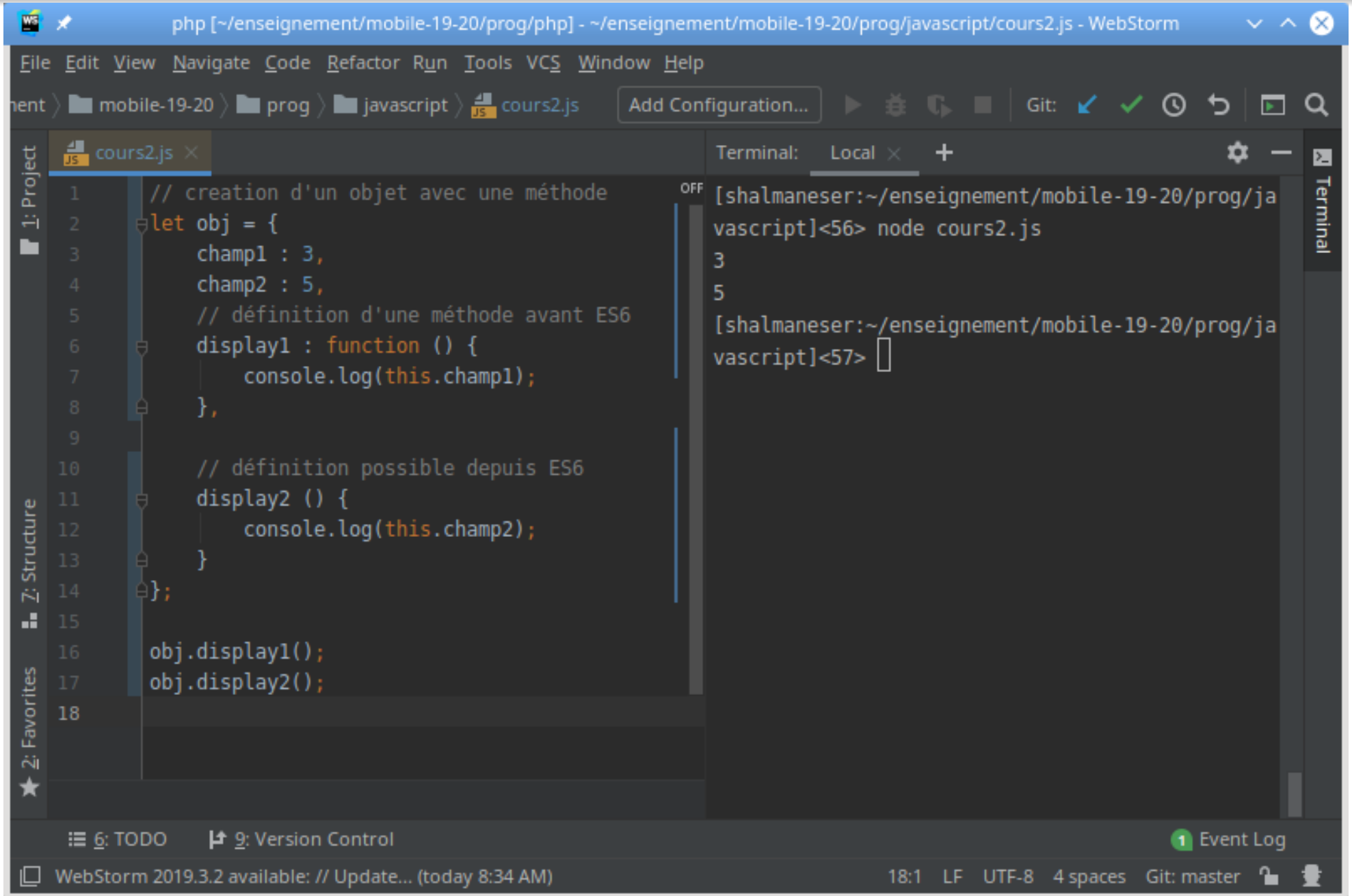
Les fonctions



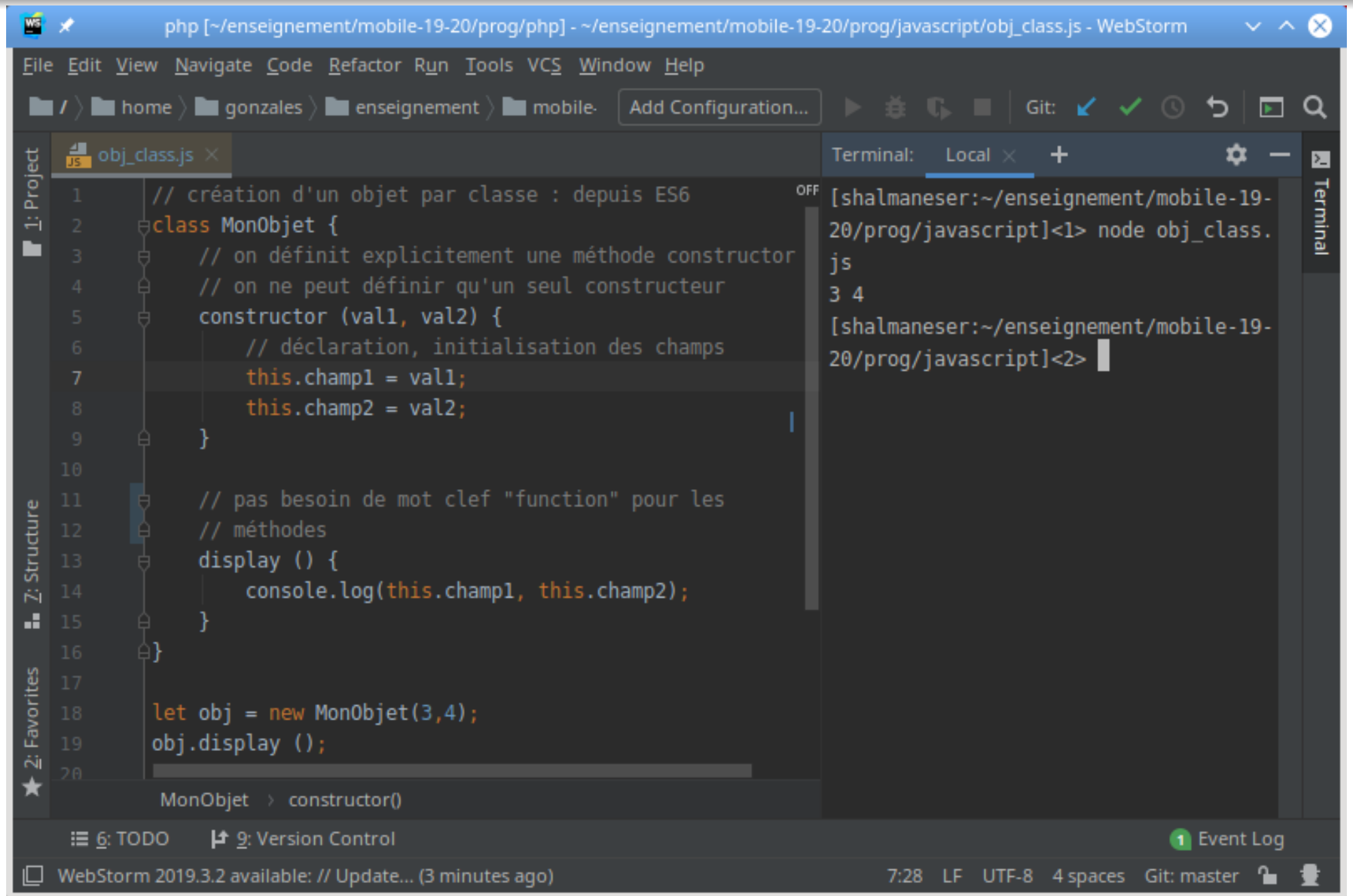
Les objets (1/3)



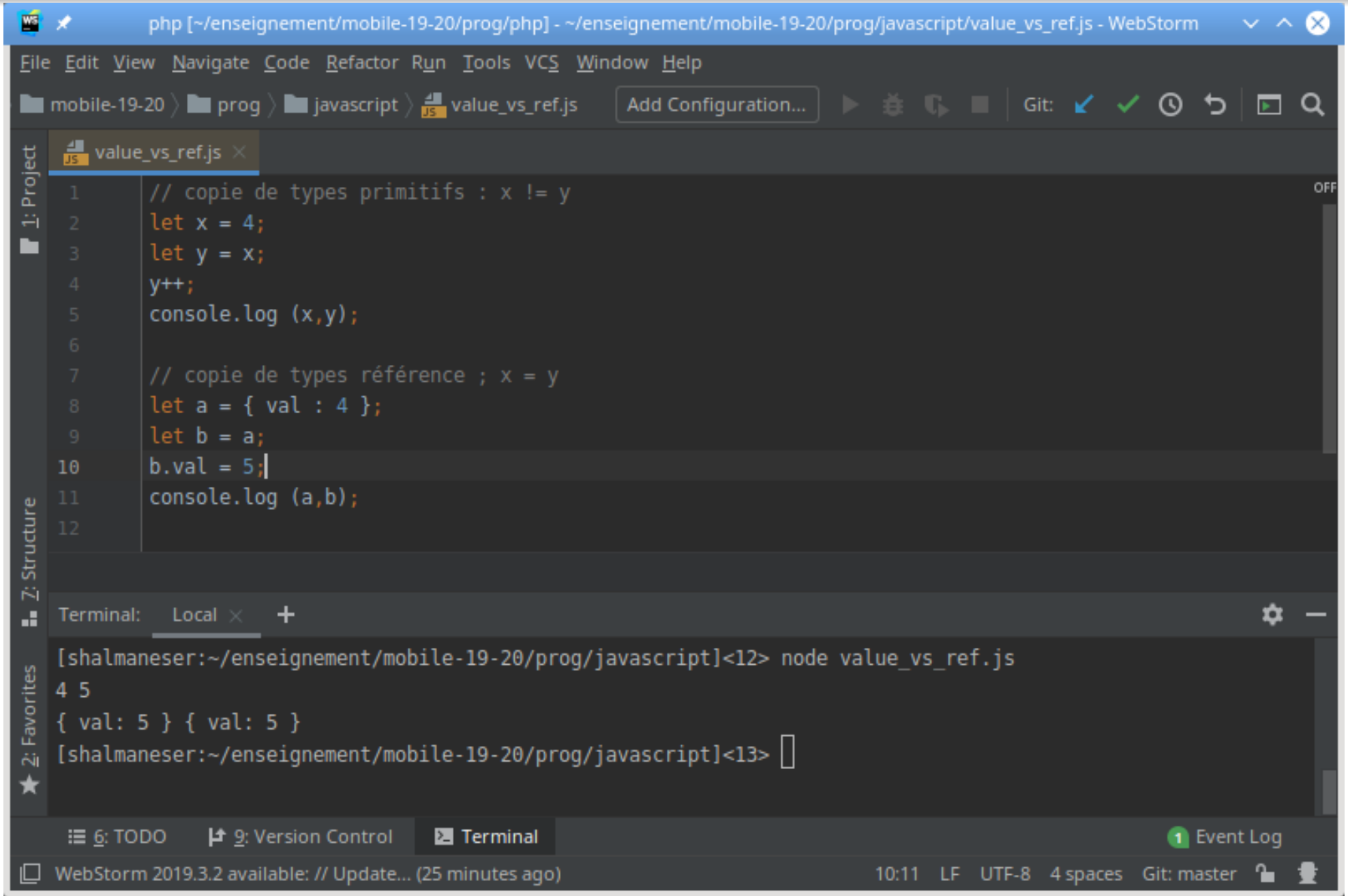
Les objets (2/3)



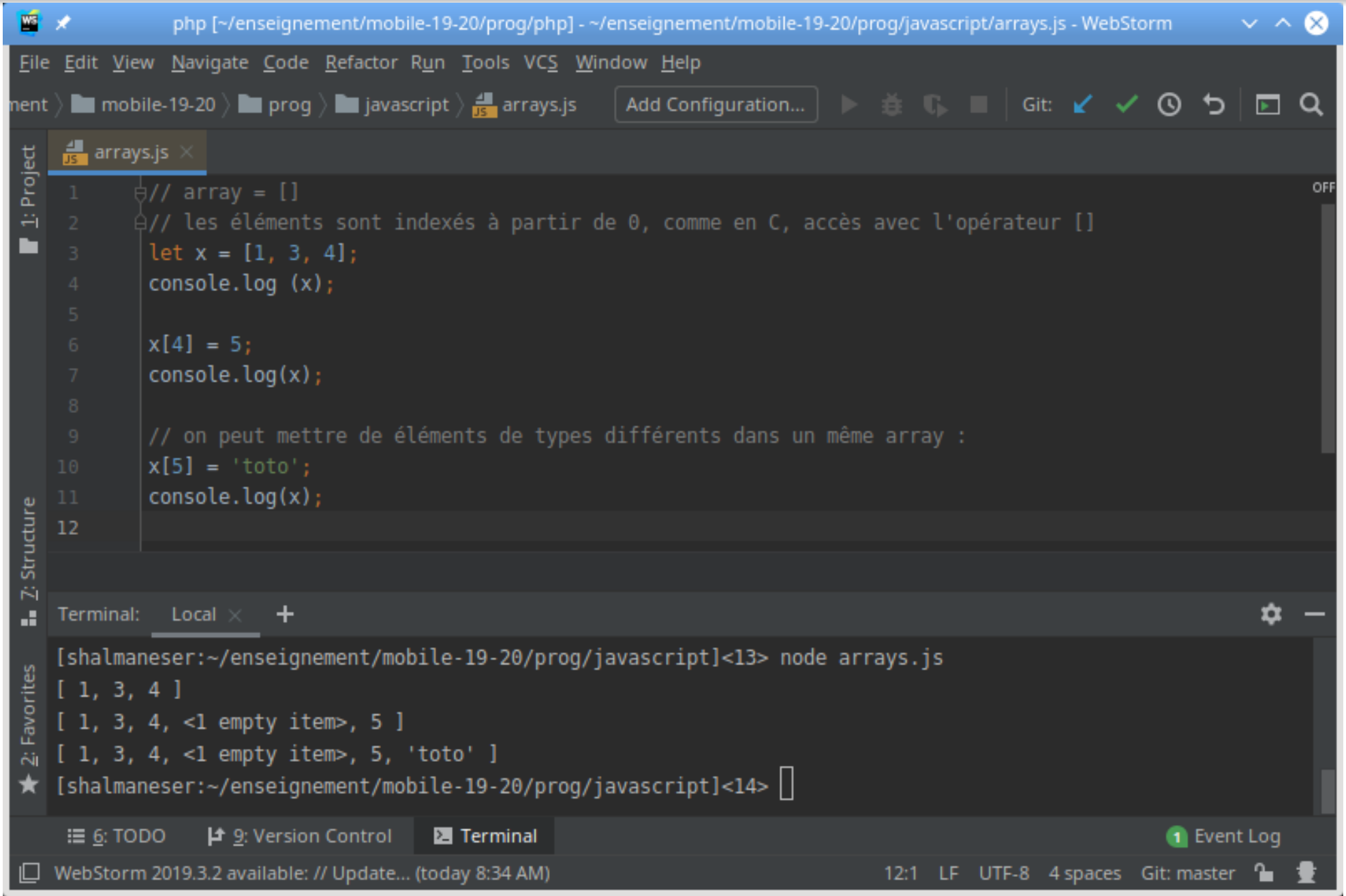
Les objets (3/3)



Comparaison types primitifs / référence

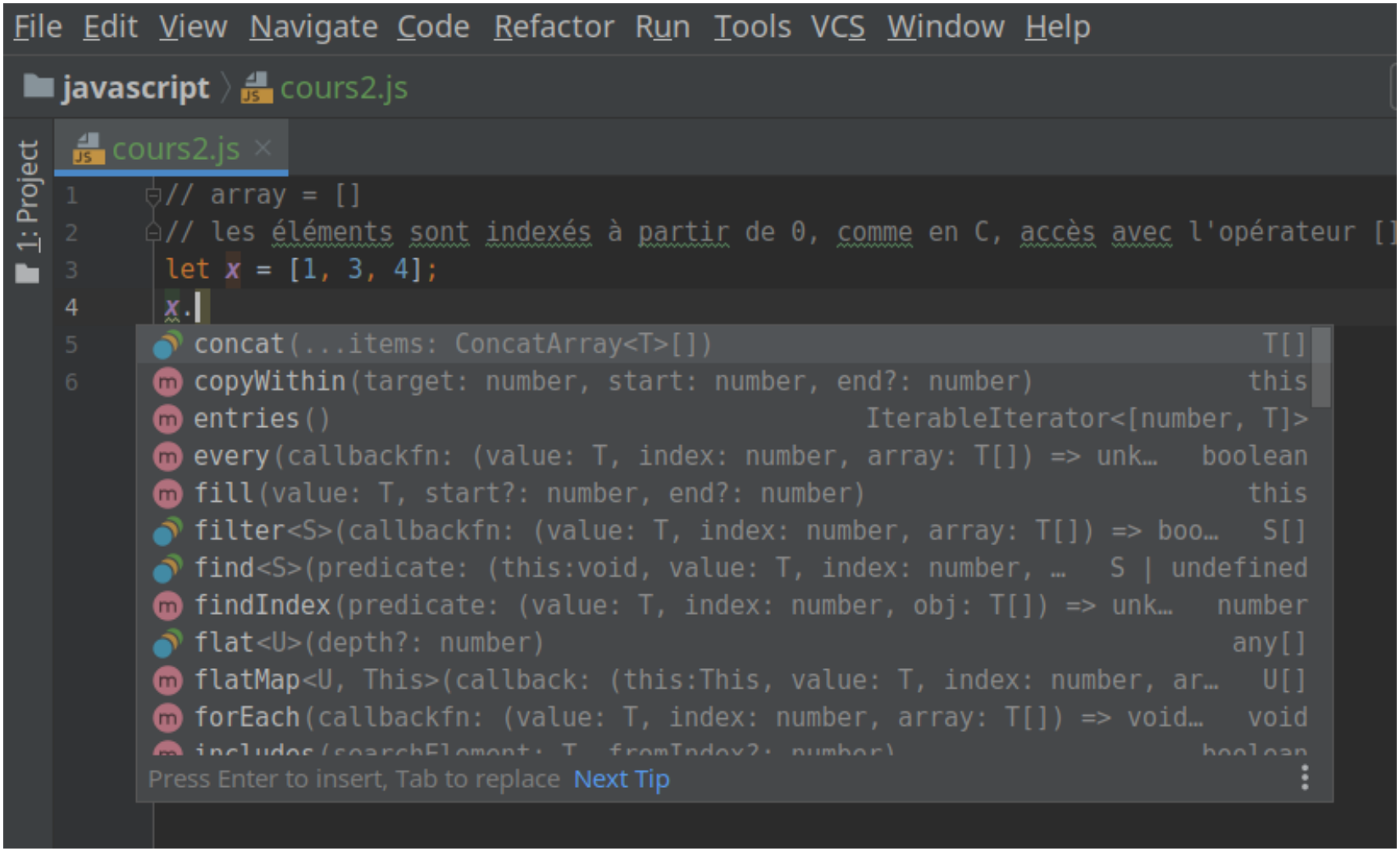


Les tableaux (1/2)



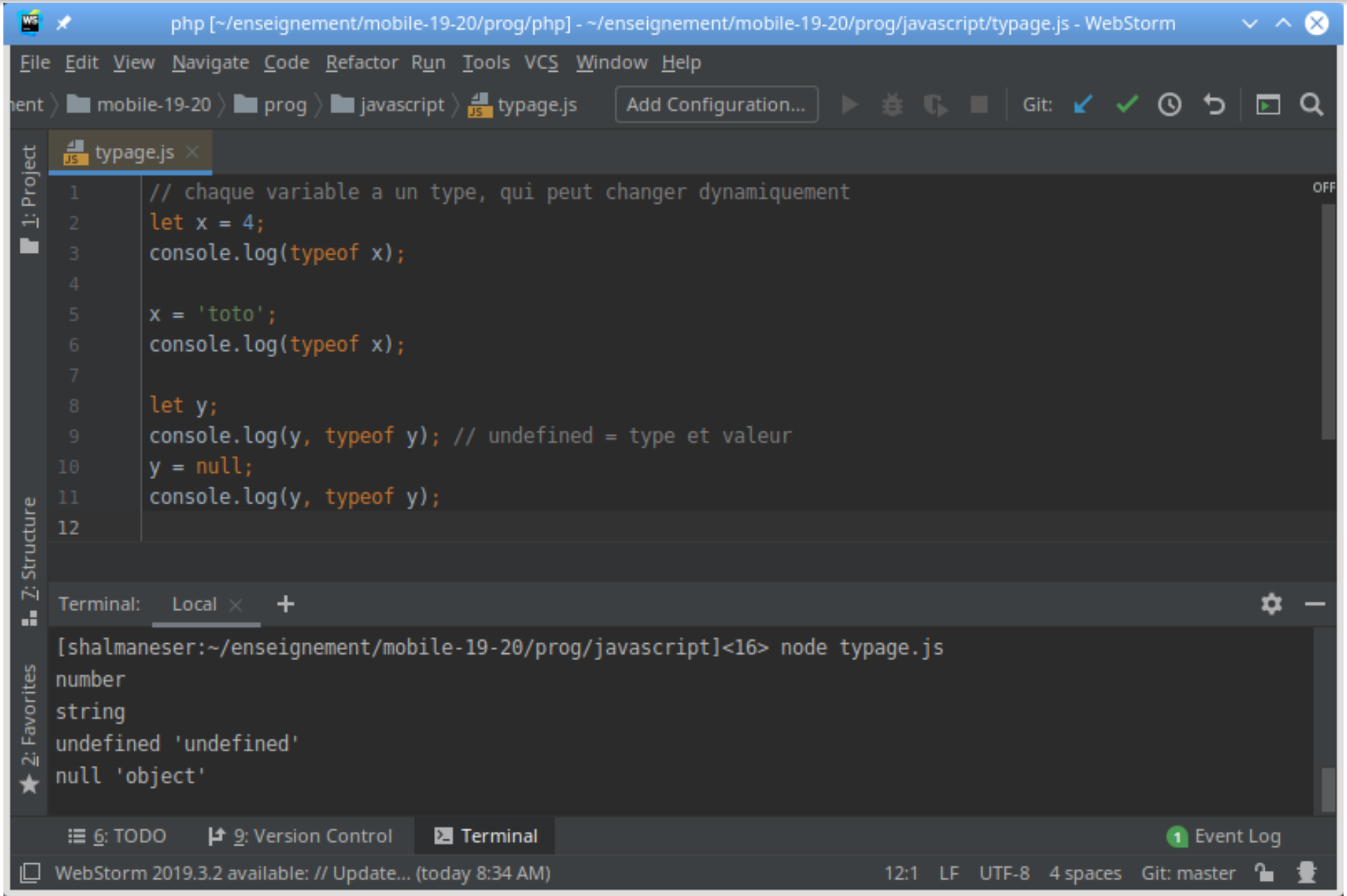
Les tableaux (2/2)

Les tableaux sont des objets :

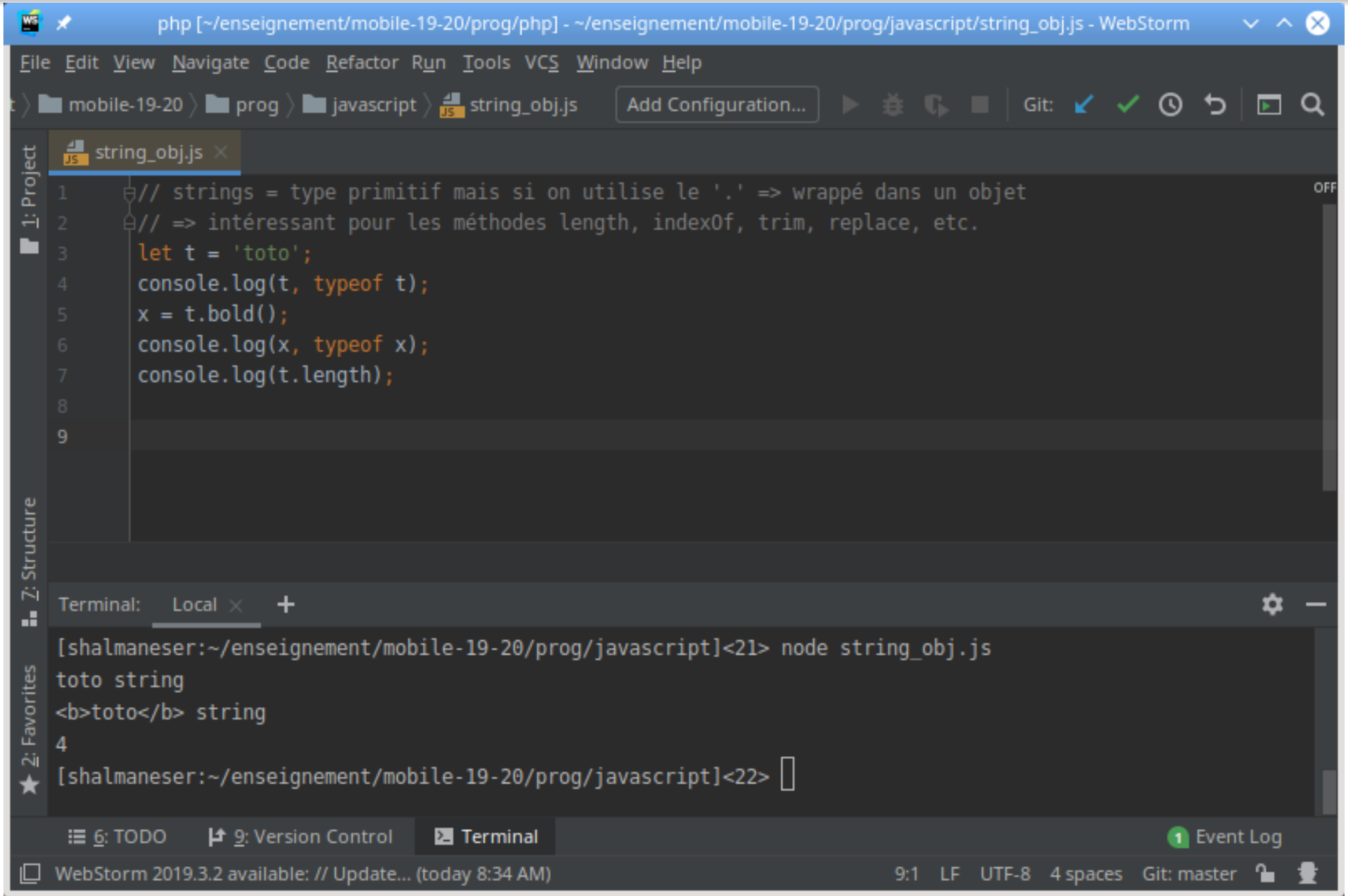


$\Longrightarrow$ contiennent des méthodes (length, filter, forEach, *etc.*)

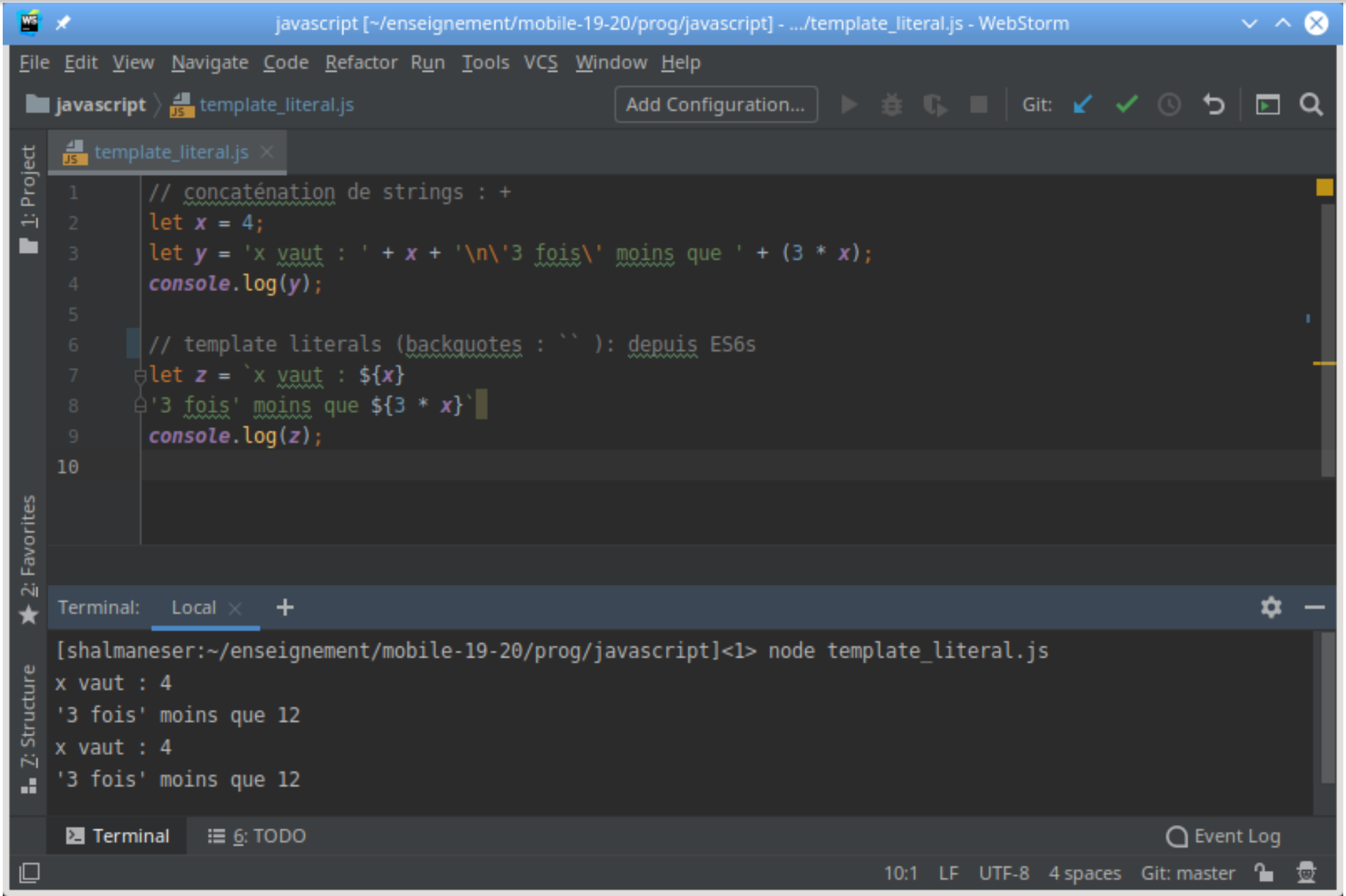
Typage dynamique



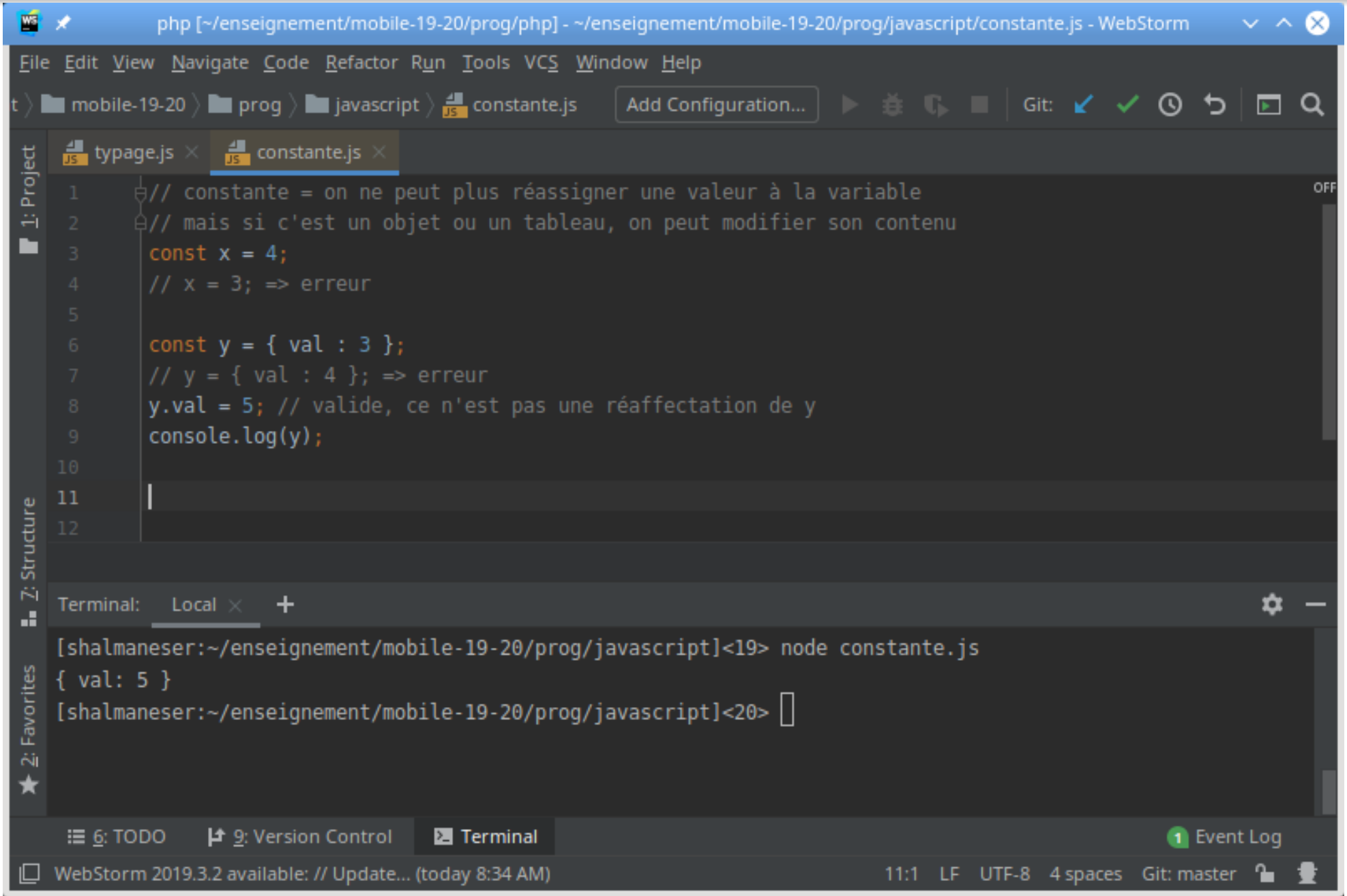
Retour sur les string et les objets



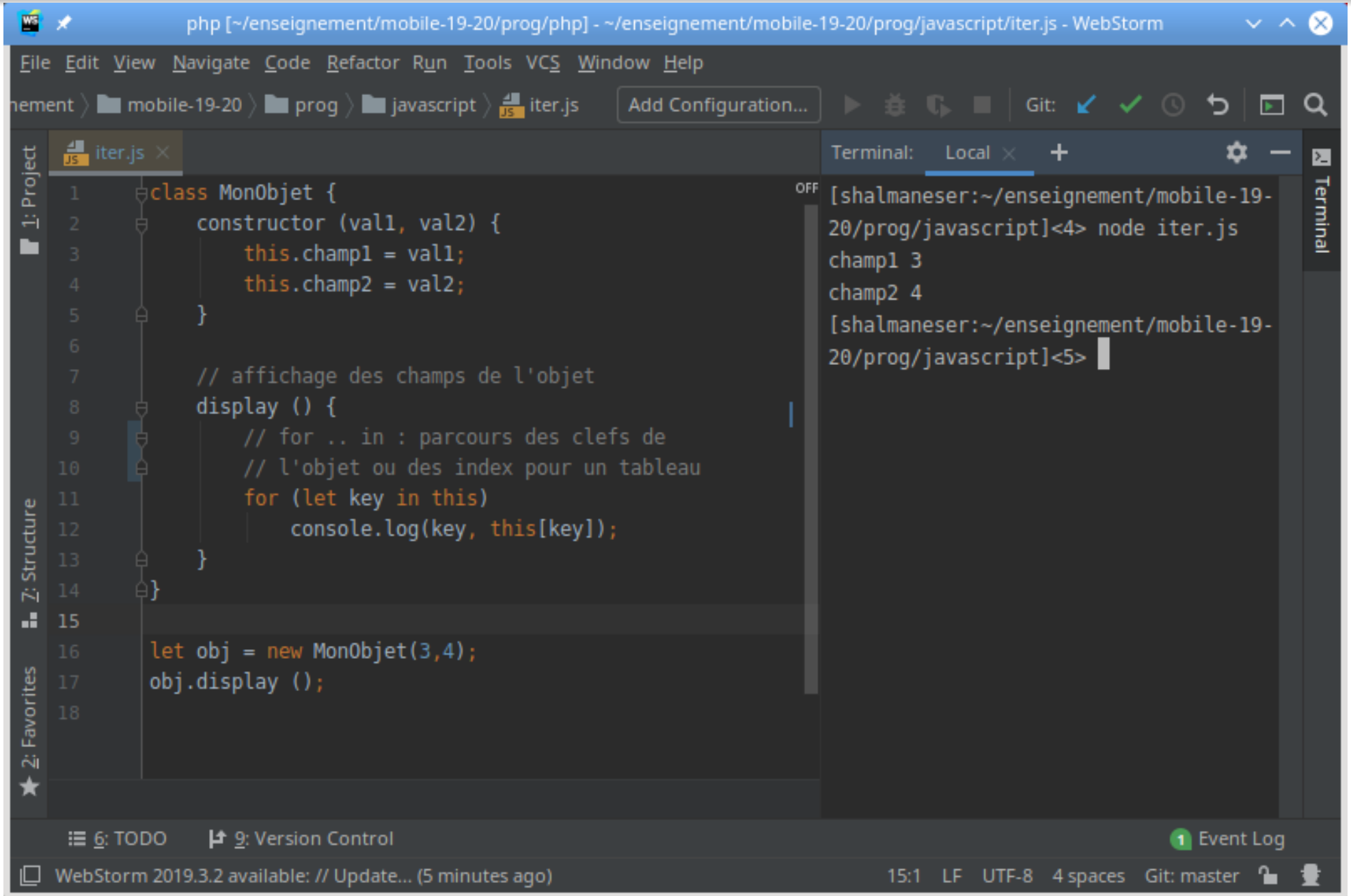
Template literals



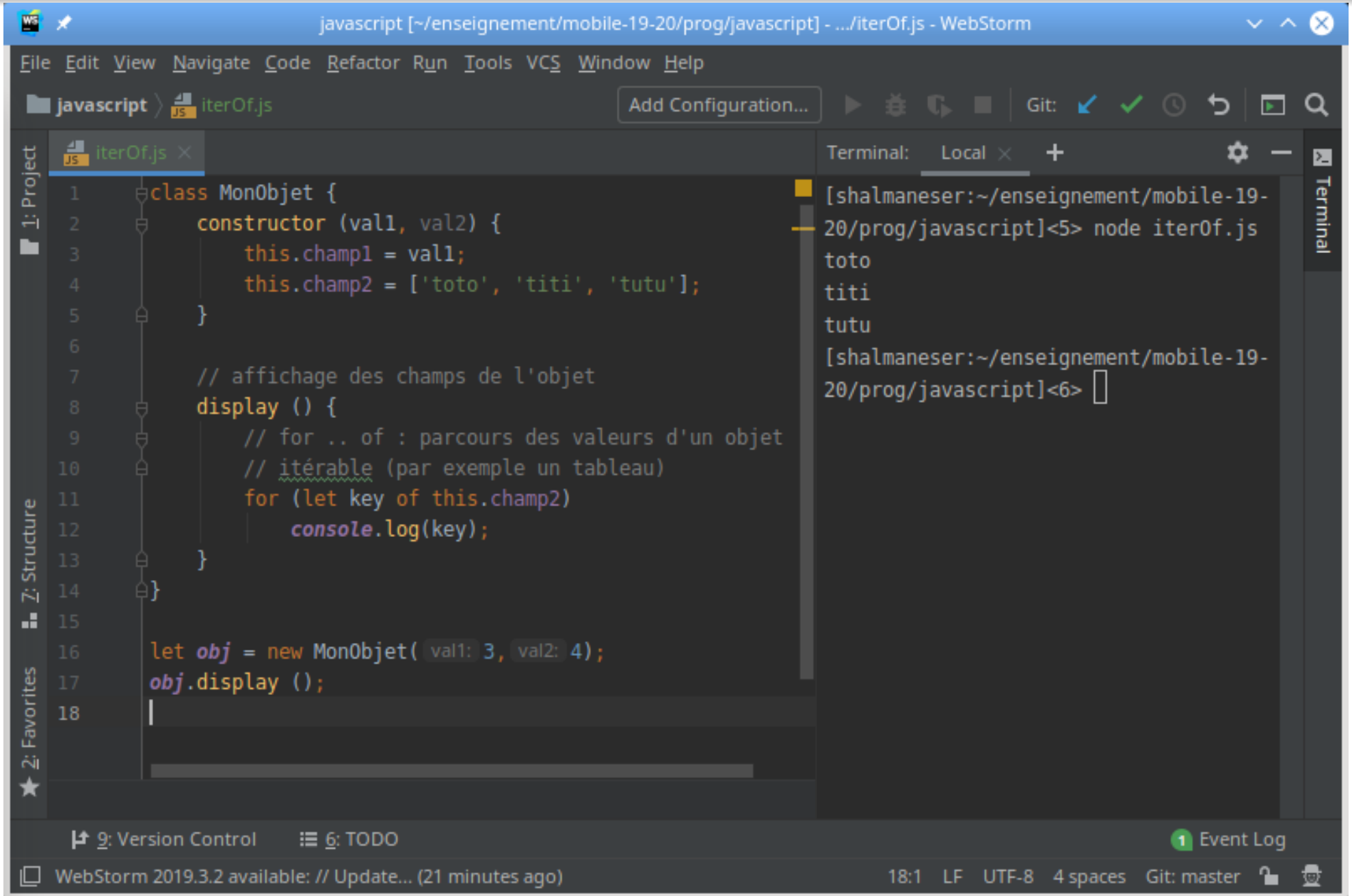
Les constantes



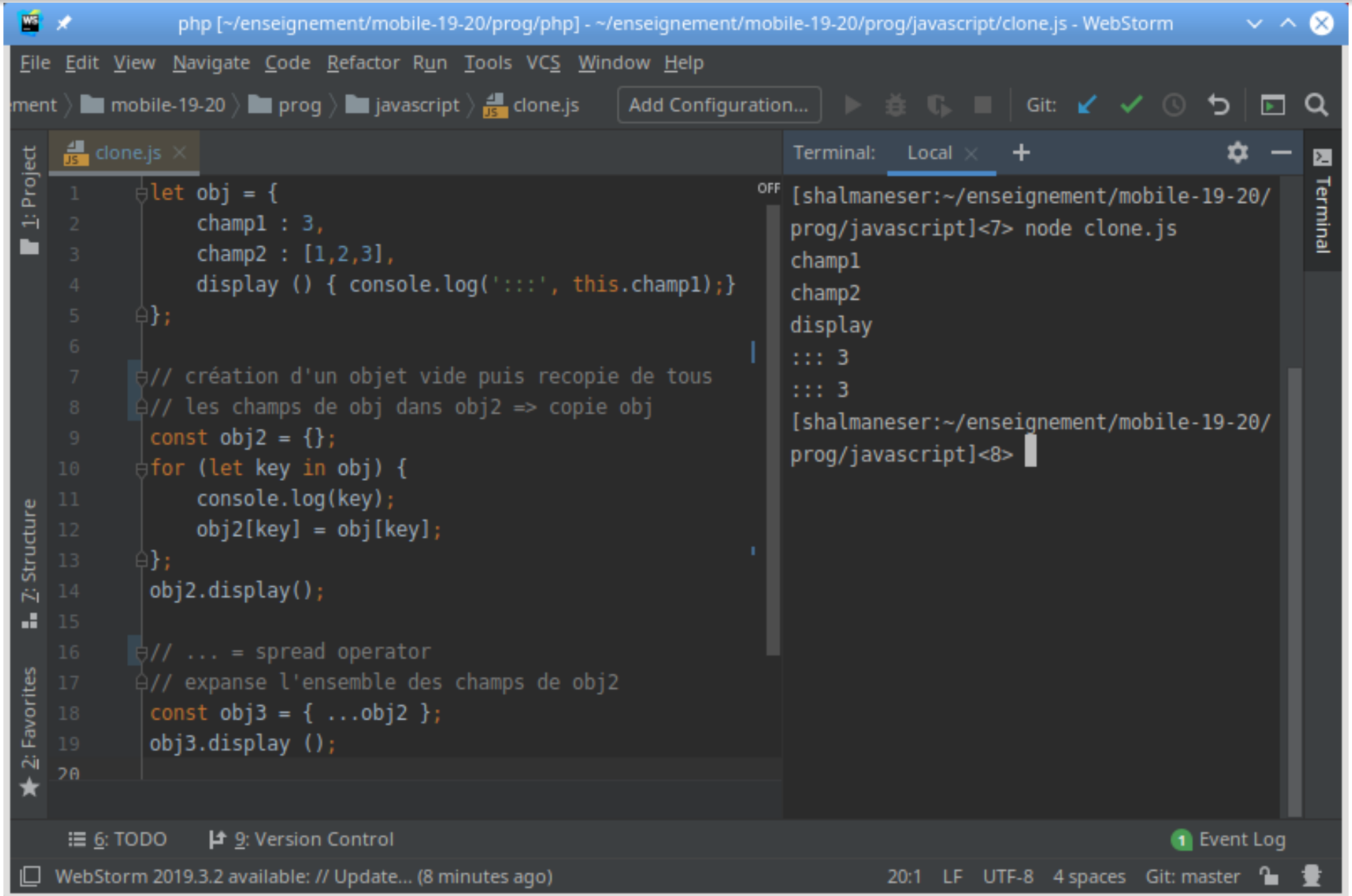
Itérer les champs d'un objet (1/2)



Itérer les champs d'un objet (2/2)



Copier (cloner) un objet



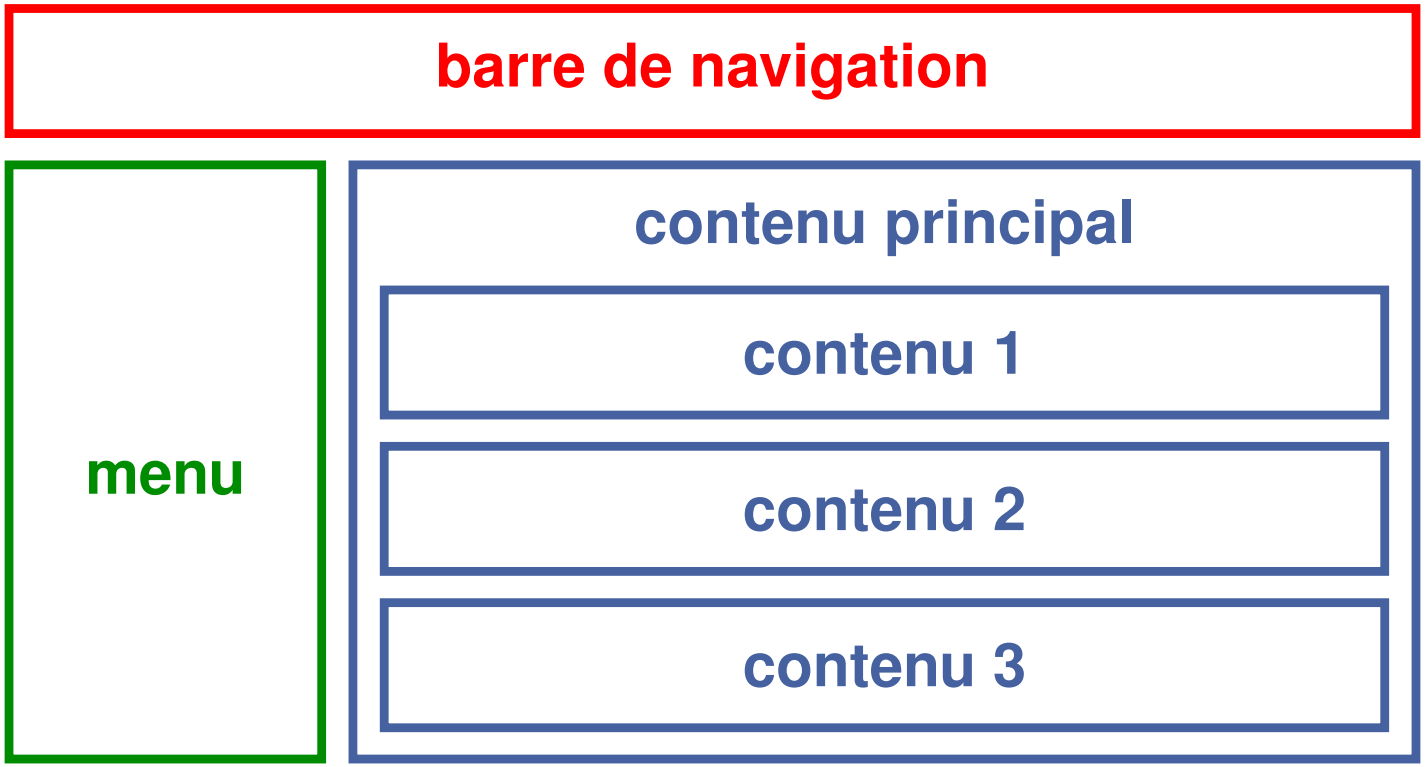
Angular 

Qu'est-ce qu'Angular

- Framework pour construire des applications clientes
  $\Longrightarrow$ front-end
- Structure l'application
  $\Longrightarrow$ simplifie programmation/maintenance/débuggage
- Mise en place de tests simple
- Utilise TypeScript/Javascript, HTML, CSS

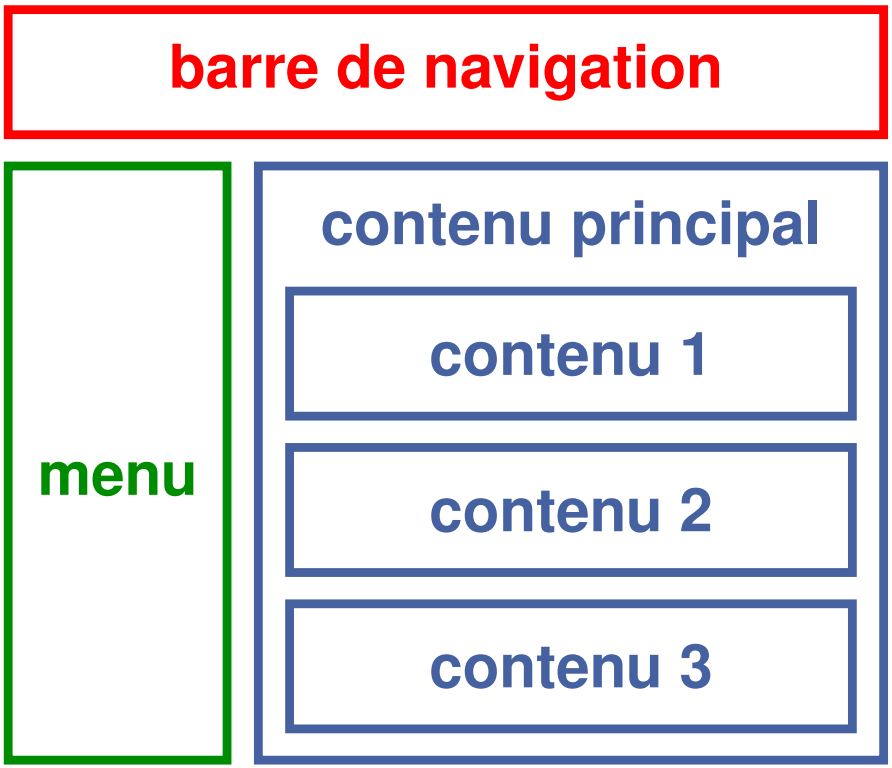
Structure d'une appplication Angular

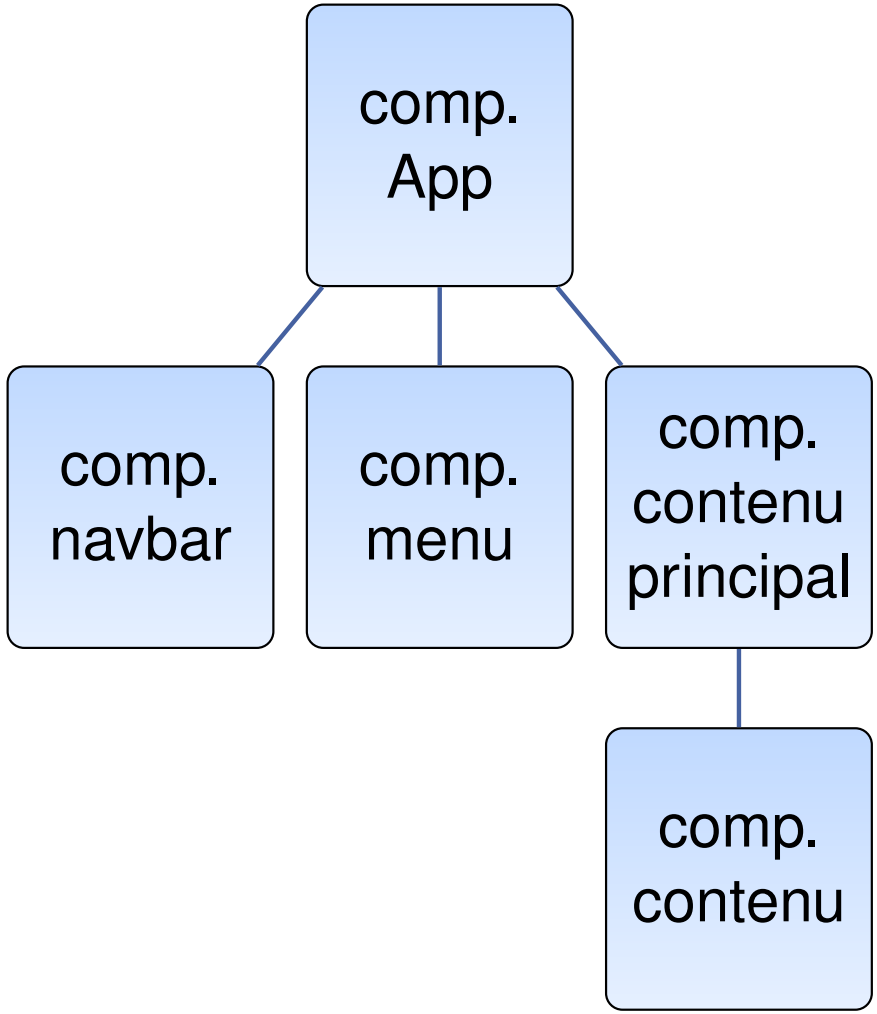
Affichage de la page web :



- Affichage $\Longrightarrow$ structure
- Chaque rectangle = composant Angular
- Intérêt des composants : réutilisables plusieurs fois
- Un composant peut en inclure d'autres

Logique de l'application : arbre de composants





$\Longrightarrow$ permet de structurer facilement le code !

Contenu d'un composant Angular

Un composant Angular contient essentiellement :

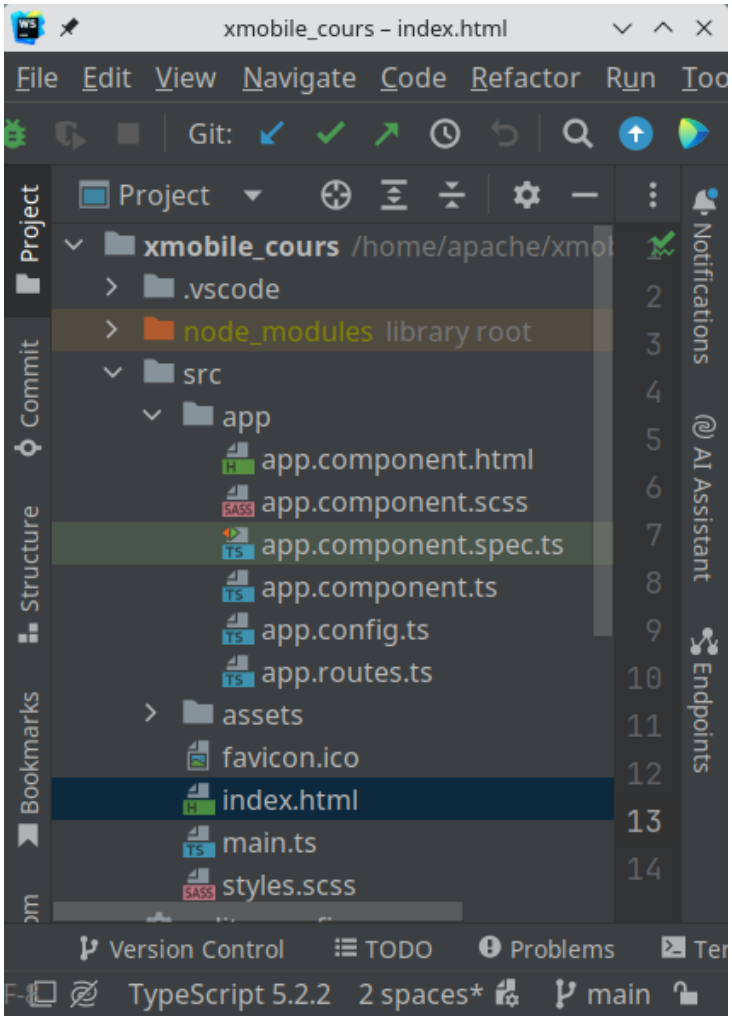
1. un fichier TypeScript contenant :
   - les données du composant
   - la logique/le comportement du composant
2. un fichier html
   - contenant le code HTML affiché par le browser
   - des instructions pour interagir avec le code TypeScript
3. un fichier css contenant le style propre au composant

- Répertoire `src/app` contient les composants
- 1 composant principal appelé app ou root

Génération d'un projet Angular

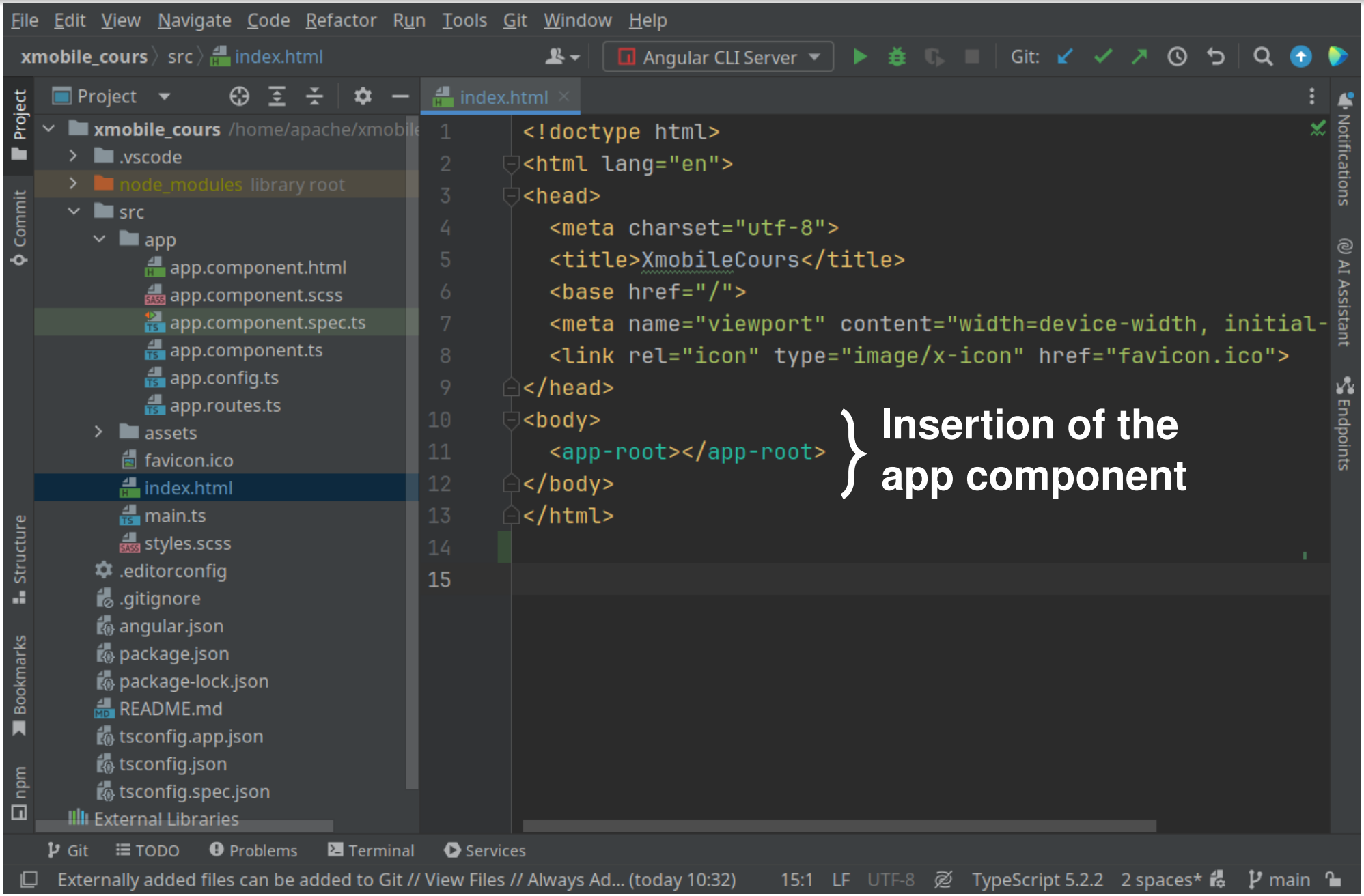
- `ng new mon-projet`

  $\Rightarrow$ crée le composant app :



- Dans src/app :
  - `app.component.ts` :
    code TypeScript
  - `app.component.spec.ts` :
    pour faire des tests
  - `app.component.html` :
    template HTML
- Dans src :
  - `index.html` :
    point d'entrée de l'appli

Index.html


File Edit View Navigate Code Refactor Run Tools Git Window Help
xmobile_cours src index.html
Angular CLI Server
Git:
Project
index.html
xmobile_cours /home/apache/xmobile
.vscode
node_modules library root
src
app
app.component.html
app.component.scss
app.component.spec.ts
app.component.ts
app.config.ts
app.routes.ts
assets
favicon.ico
index.html
main.ts
styles.scss
.editorconfig
.gitignore
angular.json
package.json
package-lock.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
External Libraries
1 <!doctype html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>XmobileCours</title>
6 <base href="/">
7 <meta name="viewport" content="width=device-width, initial-
8 <link rel="icon" type="image/x-icon" href="favicon.ico">
9 </head>
10 <body>
11 <app-root></app-root>
12 </body>
13 </html>
14
15
Insertion of the app component
Git TODO Problems Terminal Services
Externally added files can be added to Git // View Files // Always Ad... (today 10:32)
15:1 LF UTF-8 TypeScript 5.2.2 2 spaces* main
Project
Commit
Structure
Bookmarks
npm
Notifications
AI Assistant
Endpoints

Fonctionnement d'un projet Angular

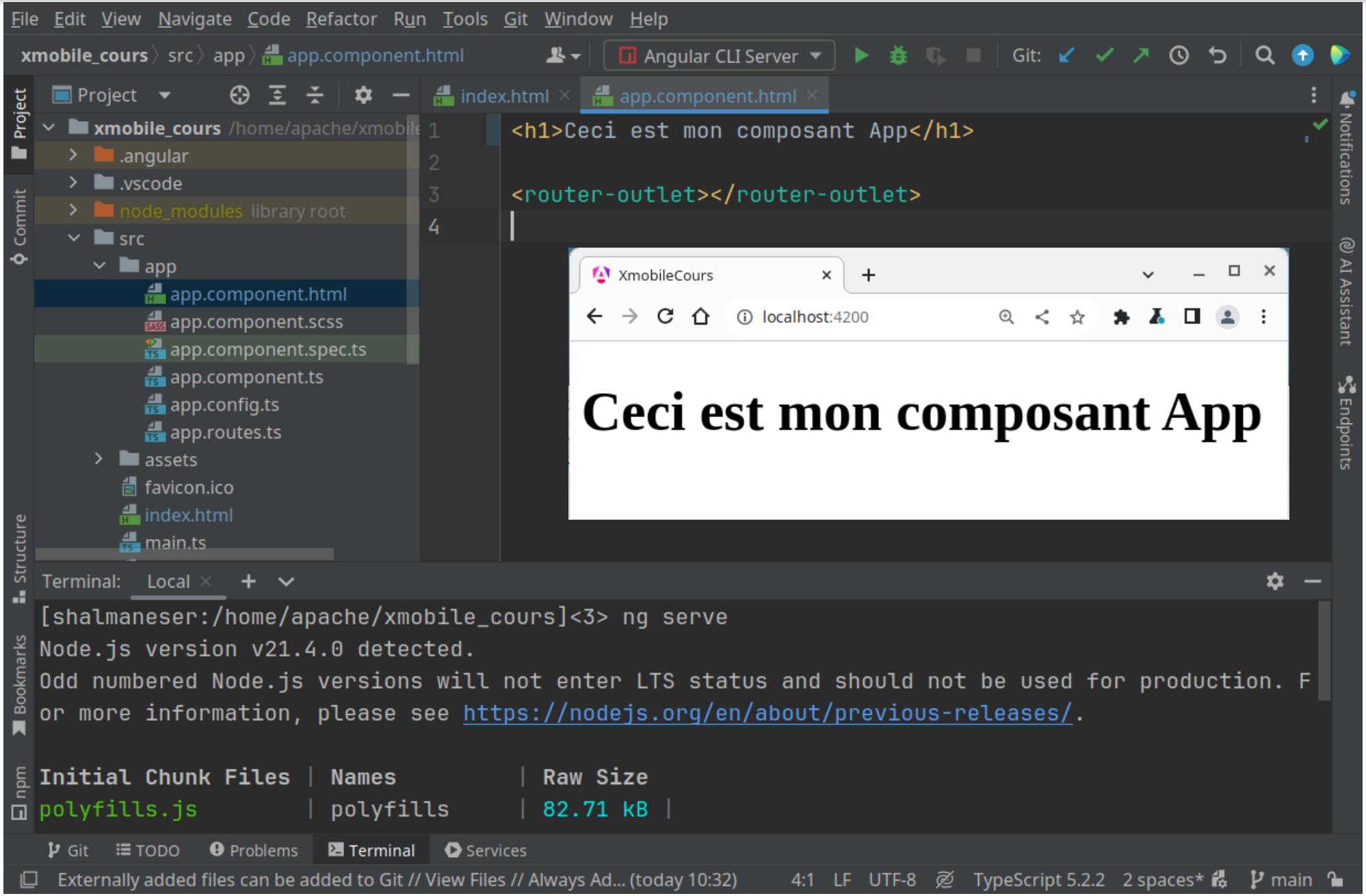
1. Créer les composants (et les modules)
2. Les insérer dans l'application via des balises dans les fichiers HTML

   Exemple : `<app-root></app-root>`

- Pour compiler et « servir » votre application (en mode dev) :

  `ng serve` ou `npm start`

Le composant App et l'appli servie



Paramétrage du ng serve / npm start

Données http $\Longrightarrow$ servir 127.0.0.1 plutôt que localhost

- npm start :
  - objet “`scripts`” du fichier `package.json` :

    ```
    "start" : "ng serve --host 127.0.0.1"
    ```

- ng serve :
  - dans l’objet “`serve` du fichier `angular.json`, rajouter :

    ```
    "options" : { "host" : "127.0.0.1" }
    ```

Le TypeScript du composant app