

Scripts clients - JavaScript

Alain Mbayo
Septembre – Octobre 2021
alain.mbayo@ifosup.wavre.be

Scripts clients - JavaScript

Objectifs du cours

→ Objectifs généraux :

- Concourir à l'épanouissement individuel en promouvant une meilleure insertion professionnelle, sociale et culturelle ;
- Répondre aux besoins et demandes en formation émanant des entreprises, des administrations, de l'enseignement et d'une manière générale des milieux socio-économiques et culturels.

Scripts clients - JavaScript

Objectifs du cours

➔ Objectifs spécifiques :

Permettre à l'étudiant :

- d'insérer des scripts clients dans des pages web auxquelles ils ajouteront des possibilités d'interaction ou d'animation ;
- de mettre en œuvre des notions de programmation orientée objet dans des scripts clients ;
- d'accroître la richesse de ses réflexions techniques et ses compétences en communication, en organisation, en observation.

Scripts clients - JavaScript

➔ Horaire :

- jour : lundi, mercredi, vendredi
- heure : 13h35 à 17h20
- durée : du 13 septembre au 29 octobre 2021

➔ Examen 1^{ère} session

➔ Examen 2^{ième} session

Scripts clients - JavaScript

➔ Consignes :

- respect du port du masque et se désinfecter les mains
- correspondre toujours avec l'adresse email de l'école
- si un.e étudiant.e n'est pas sur la liste, aller au secrétariat pour se faire inscrire
- adresse email des étudiants :
prenom.nom@ifosup.wavre.be
- mot de passe : *IFOSUP21-22*

Scripts clients - JavaScript

Plan

- 1. Introduction
- 2. Programme JavaScript
 - 2.1 Environnement de développement
 - 2.2 Code JavaScript
 - 2.3 Exécuter JavaScript
 - 2.4 Console de développement JavaScript
 - 2.5 Commenter le code

Scripts clients - JavaScript

Plan

- 3. Les variables
 - 3.1 Créer et utiliser des variables
 - 3.2 Comprendre la portée des variables
 - 3.3 Nommer les variables
 - 3.4 Connaître les types de données de JavaScript
 - 3.5 Utiliser des fonctions intégrées pour travailler avec les variables

Scripts clients - JavaScript

- 4. Les tableaux
 - 4.1 Identifier et définir des tableaux
 - 4.2 Construire des tableaux
 - 4.3 Tableaux multidimensionnels
 - 4.4 Travailler avec les éléments des tableaux
 - 4.5 Utiliser les fonctions et les propriétés des tableaux

Scripts clients - JavaScript

- 5. Les opérateurs, les expressions et les instructions
 - 5.1 Lire et coder des expressions JavaScript
 - 5.2 Les opérateurs d'affectation
 - 5.3 Les opérateurs de comparaison
 - 5.4 Les opérateurs arithmétiques
 - 5.5 Les opérateurs au niveau du bit
 - 5.6 Les opérateurs des chaînes
 - 5.7 Les opérateurs logiques

Scripts clients - JavaScript

- 6. Les boucles et les branchements
- 7. Les fonctions
- 8. Les objets
- 9. Contrôler le navigateur avec l'objet Window
- 10. Manipuler les documents avec le DOM

Scripts clients - JavaScript

- 11. Travailler avec CSS et les graphismes
- 12. Exécuter des recherches avec des expressions régulières
- 13. Comprendre les fonctions de rappel et les fermetures
- 14. Ajax et JSON
- 15. HTML et ses API
- 16. jQuery
- 17. Dix bibliothèques et frameworks
- 18. Utilisation des événements
- 19. Entrées et sorties

Scripts clients - JavaScript

1. Introduction

- Début du Web(*) :
 - les navigateurs n'avaient pas de capacités particulières (n'étaient que des simples lecteurs)
 - Guerre des navigateurs : certains proposent l'affichage des images, d'autres l'utilisation des polices de caractères différents, le clignotement des textes, le déplacement des textes
 - Une idée s'impose : doter les navigateurs de la capacité de gérer eux-mêmes les choses

(*) Le chercheur britannique Tim Berners-Lee a inventé le World Wide **Web** en 1989, lorsqu'il travaillait au CERN. À l'origine, le projet a été conçu et développé pour que des scientifiques travaillant dans des universités et instituts du monde entier puissent s'échanger des informations instantanément

Source : <https://home.cern/fr/science/computing/birth-web/short-history-web#:~:text=Le%20chercheur%20britannique%20Tim%20Berners,s'%C3%A9changer%20des%20informations%20instantan%C3%A9ment.>

Scripts clients - JavaScript

1. Introduction (suite)

- Naissance de JavaScript : 1995, par Brandon Eich, qui travaillait pour Netscape (légende : il aurait écrit JavaScript en 10 jours !) : il s'est beaucoup inspiré des meilleures fonctionnalités de divers autres langages
- Nom original de JavaScript : **Mocha**
- 1ère version bêta, rebaptisée **LiveScript**
- Nom définitif, dans la version Netscape 2 (1995) : **JavaScript** (la référence à **Java** était marketing, du fait de la popularité de ce dernier à l'époque. Aujourd'hui, c'est l'inverse ! **JavaScript** et **Java** n'ont d'autre lien que celui-là)
- Le concurrent, Microsoft, lança un clone exact dans Internet Explorer : **Jscript**

Scripts clients - JavaScript

1. Introduction (suite)

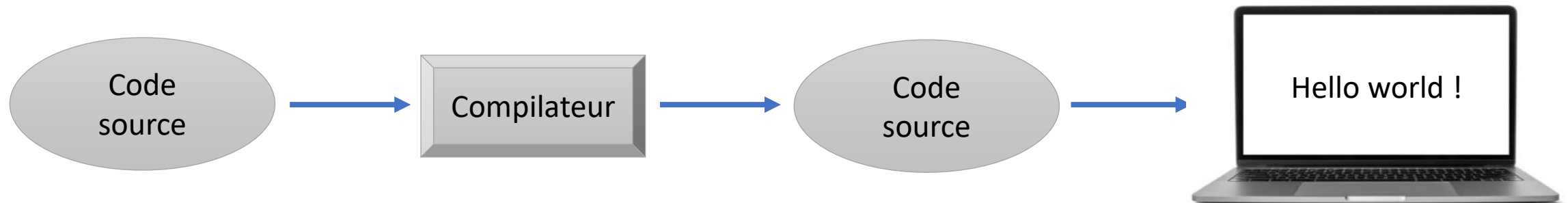
- Popularité grandissante de JavaScript → rendre les pages web plus dynamiques
- Conséquences de l'incorporation de JavaScript dans les navigateurs web → DHTML (Dynamic HTML) : introduction des menus déroulants, validation des formulaires, affichage des fenêtres pop-up, etc.
- JavaScript aujourd'hui → Langage de programmation le plus utilisé au monde (*top 10 des langages les plus utilisés, respectivement : JavaScript, Python, Java, PHP, C#, C++, CSS, TypeScript, Ruby, C*)

(Source : <https://www.blogdumoderateur.com/langages-programmation-evolution-tendances-communautes-emploi/>)

Scripts clients - JavaScript

1. Introduction (suite)

- JavaScript est un langage de script dynamique, car interprété et non compilé :
 - Langages compilés : le développeur écrit le code, l'envoie au compilateur, un programme spécial qui va convertir ce code en langage machine. C'est ce dernier qui est exécuté par l'ordinateur.



- Exemples : C, C++, Fortan, Java, Objective-C, COBOL

Scripts clients - JavaScript

1. Introduction (suite)

- Langages interprétés : le code source est converti en langage machine au moment de la lecture du code par le navigateur web (ie, au moment de l'exécution du programme).
 - intérêts : facilité de modifier le programme à tout moment
 - inconvénient : baisse des performances car la phase de la conversion en langage machine rajoute une étape supplémentaire au processus → mais cela a changé, grâce à la plus grande puissance et rapidité des ordinateurs actuels



- Exemples : PHP, Perl, Ruby, JavaScript, etc.

Scripts clients - JavaScript

1. Introduction (suite)

- JavaScript, en plus d'être dans les navigateurs web, se retrouve aujourd'hui dans les smartphones et tablettes, sur des serveurs web, dans des applications de bureau, dans toute sorte des dispositifs connectés portables.
- Dans les navigateurs web, JavaScript accomplit plusieurs tâches :
 - Contrôler le navigateur lui-même et se servir de ses fonctionnalités
 - Manipuler la structure et le contenu des pages web → (HTML5)
 - Manipuler les styles des pages web (mise en page, polices de caractères utilisés, ..) → (CSS : Cascading Style Sheet)
 - Accéder à des données provenant d'autres sources

Scripts clients - JavaScript

1. Introduction (suite)

- Quelques tâches réalisées par JavaScript :
 - Créer des effets chic et choc
 - Valider des saisies
 - Créer des effets d'enroulement
 - Créer des menus déroulants ou contextuels
 - Gérer le glisser et déposer
 - Faire défiler des pages web à l'infini
 - Compléter automatiquement des champs
 - Créer des barres de progression
 - Tabuler à l'intérieur de pages web
 - Créer des listes pouvant être triées
 - Proposer des zooms « magiques », etc.

Scripts clients - JavaScript

2. Programme JavaScript

2.1 Environnement de développement :

- un navigateur web : pratiquement, tous les navigateurs exécutent le code JavaScript correctement
→ préférence : *Google Chrome*
- un éditeur de code : possède des fonctionnalités dédiées à la programmation
→ exemples : Sublime Text, Notepad++, Coda, Visual Studio Code, etc.
→ préférence : *Visual Studio Code*
- **Code JavaScript :**
 - Sensible à la casse : fait la différence entre les majuscules et les minuscules (« **Adresse** » et « **adresse** » sont 2 mots différents pour JavaScript)
→ cause de plusieurs erreurs dans les codes !

Scripts clients - JavaScript

2. Programme JavaScript (suite)

- Non sensible aux espaces blancs : espaces proprement dits, tabulations et sauts de ligne
 - profiter de cette particularité pour insérer assez d'espaces blancs dans le code pour la lisibilité
 - Exception : les espaces blancs dans les textes que l'on veut afficher à l'écran
- Faire attention aux mots réservés : comme tout langage, JavaScript comprend des mots réservés, qu'il ne faut pas utiliser dans les variables. Exemples : *function, while, break, with, etc.*
- Terminer les instructions par un point-virgule
 - Si une instruction n'est pas terminée par un point-virgule, JavaScript le rajoutera; mais, cela peut causer des résultats imprévisibles !

Scripts clients - JavaScript

2. Programme JavaScript (suite)

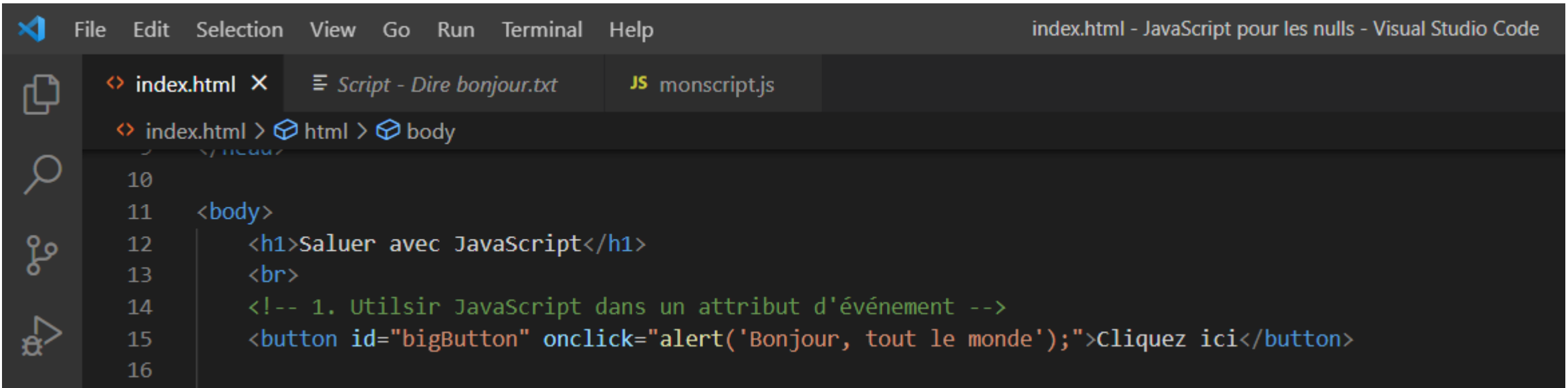
- Exécuter JavaScript dans la fenêtre du navigateur :

➔ 3 manières de le faire :

- *Placer le code dans un attribut d'événement HTML*

Exemples :

- événement **onclick** : lorsqu'on clique sur un bouton

A screenshot of the Visual Studio Code editor interface. The top menu bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. The title bar shows 'index.html - JavaScript pour les nulls - Visual Studio Code'. The editor has three tabs: 'index.html' (active), 'Script - Dire bonjour.txt', and 'JS monscript.js'. The breadcrumb navigation shows 'index.html > html > body'. The code in the editor is as follows:

```
10
11 <body>
12   <h1>Saluer avec JavaScript</h1>
13   <br>
14   <!-- 1. Utiliser JavaScript dans un attribut d'événement -->
15   <button id="bigButton" onclick="alert('Bonjour, tout le monde');">Cliquez ici</button>
16
```



Saluer avec JavaScript

Cliquez ici



Saluer avec JavaScript

Cliquez ici

127.0.0.1:5500 indique

Bonjour, tout le monde

OK

Scripts clients - JavaScript

2. Programme JavaScript (suite)

- *Placer le code entre des balises de début et de fin de script*

```
<script type="text/javascript"> // Mon code Javascript ... </script>
```

➔ Cfr code « *Saluer avec JavaScript* »

- *Inclure dans le document HTML, le code qui se trouvait dans un document « .js » séparé*

```
<script type="text/javascript" src="monscript.js"></script>
```

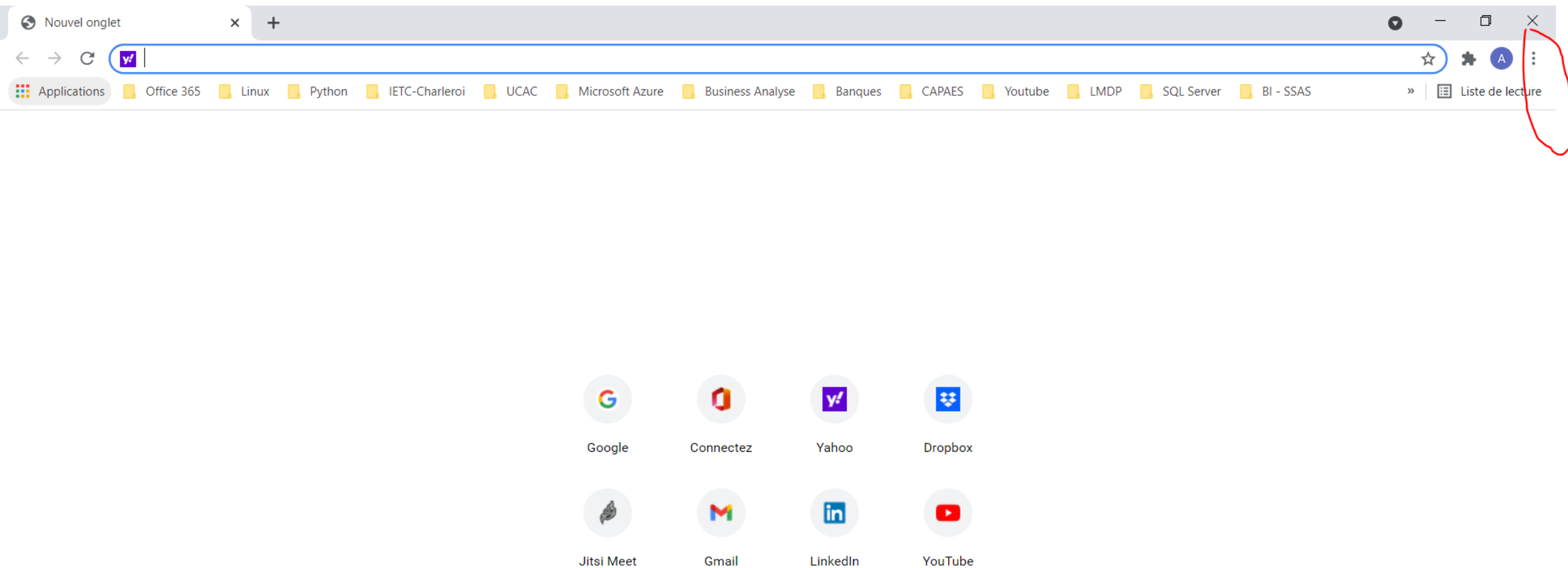
➔ Copier le code « *Saluer avec JavaScript* » dans le fichier « **monscript.js** »

Scripts clients - JavaScript

2. Programme JavaScript (suite)

- **Console de développement JavaScript :**
 - ➔ But : pouvoir exécuter des commandes JavaScript sans créer de page HTML contenant le script ou incluant un fichier de script distinct.
 - ➔ Comment accéder à cette console :
 - Ouvrir Chrome, et cliquer sur le menu en haut et à droite de la fenêtre

Scripts clients - JavaScript

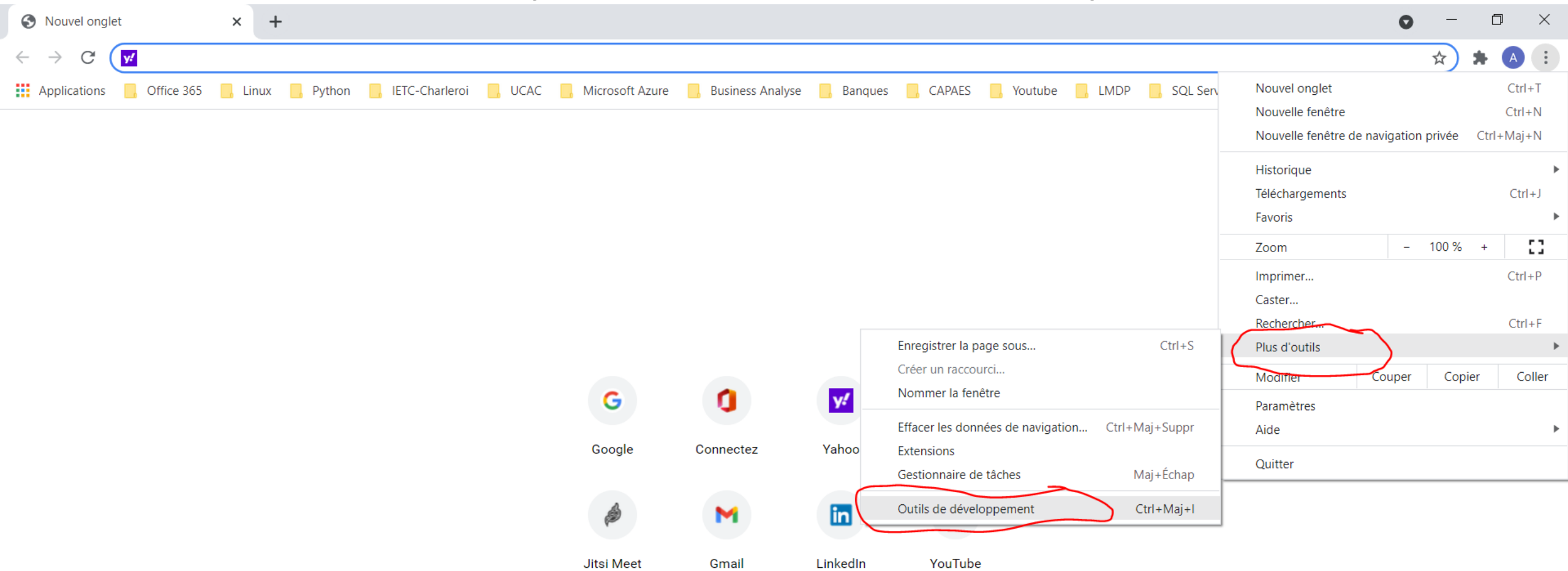


Scripts clients - JavaScript

2. Programme JavaScript (suite)

- **Console de développement JavaScript :**
 - ➔ But : pouvoir exécuter des commandes JavaScript sans créer de page HTML contenant le script ou incluant un fichier de script distinct.
 - ➔ Comment accéder à cette console :
 - Ouvrir Chrome, et cliquer sur le menu en haut et à droite de la fenêtre
 - Cliquer sur **Plus d'outils**, puis sur **Outils de développement**

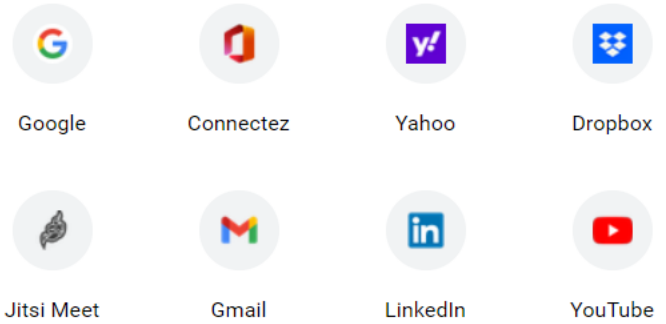
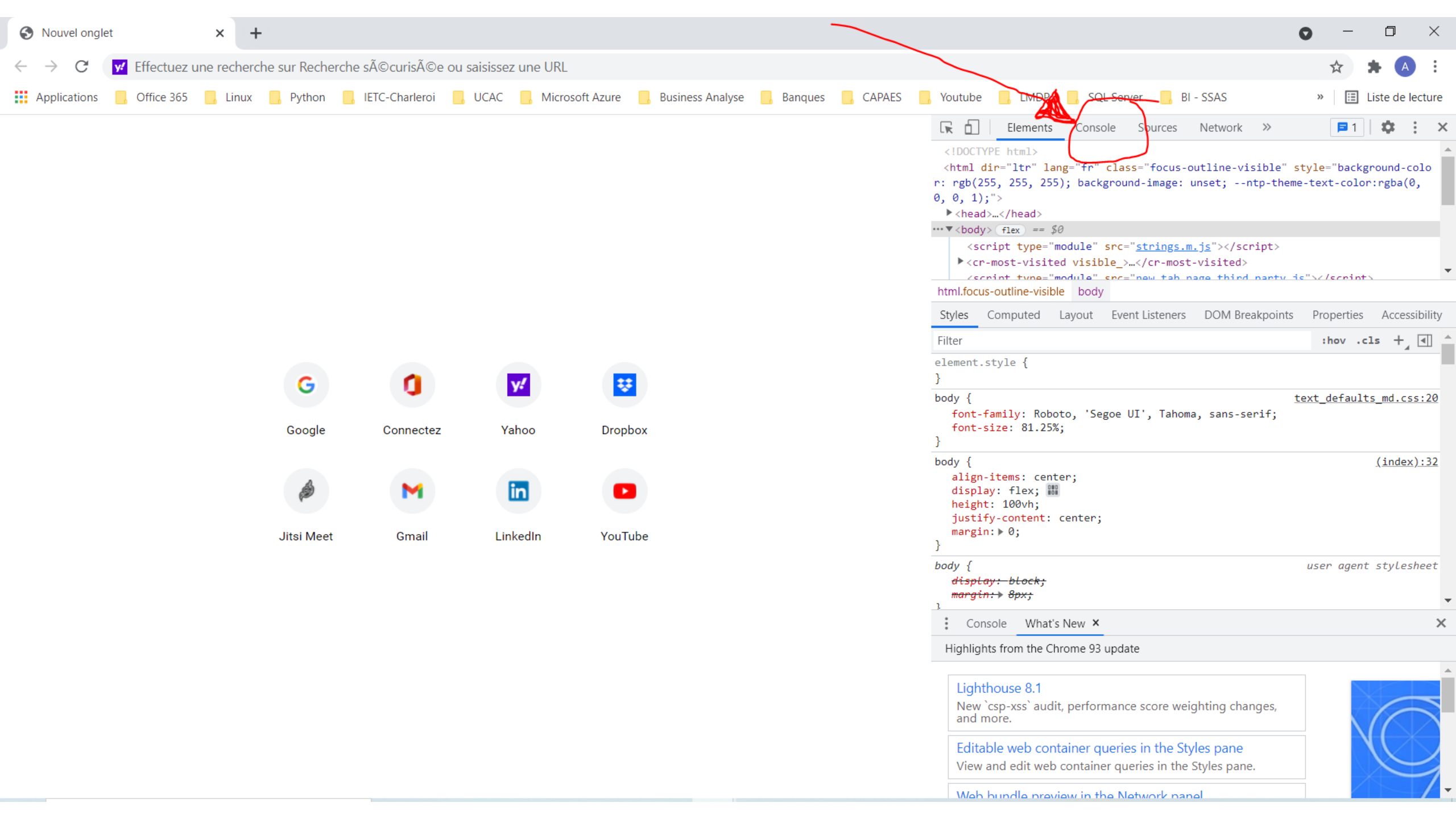
Scripts clients - JavaScript



Scripts clients - JavaScript

2. Programme JavaScript (suite)

- **Console de développement JavaScript :**
 - ➔ But : pouvoir exécuter des commandes JavaScript sans créer de page HTML contenant le script ou incluant un fichier de script distinct.
 - ➔ Comment accéder à cette console :
 - Ouvrir Chrome, et cliquer sur le menu en haut et à droite de la fenêtre
 - Cliquer sur **Plus d'outils**, puis sur **Outils de développement**
 - Cliquer sur **Console**
 - ➔ Une autre manière d'y arriver plus rapidement ➔ utiliser les raccourcis claviers :
 - Windows : **Ctrl + Maj + J**
 - Mac : **Alt + Command + J**



Elements Console Sources Network

```
<!DOCTYPE html>
<html dir="ltr" lang="fr" class="focus-outline-visible" style="background-color: rgb(255, 255, 255); background-image: unset; --ntp-theme-text-color: rgba(0, 0, 0, 1);">
  <head>...</head>
  <body>
    <script type="module" src="strings.m.js"></script>
    <cr-most-visited visible_>...</cr-most-visited>
    <script type="module" src="new_tab_page_third_party.js"></script>
```

html.focus-outline-visible body

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls +

```
element.style {
}

body {
  font-family: Roboto, 'Segoe UI', Tahoma, sans-serif;
  font-size: 81.25%;
}

body {
  align-items: center;
  display: flex;
  height: 100vh;
  justify-content: center;
  margin: 0;
}

body {
  display: block;
  margin: 0px;
}
```

Console What's New

Highlights from the Chrome 93 update

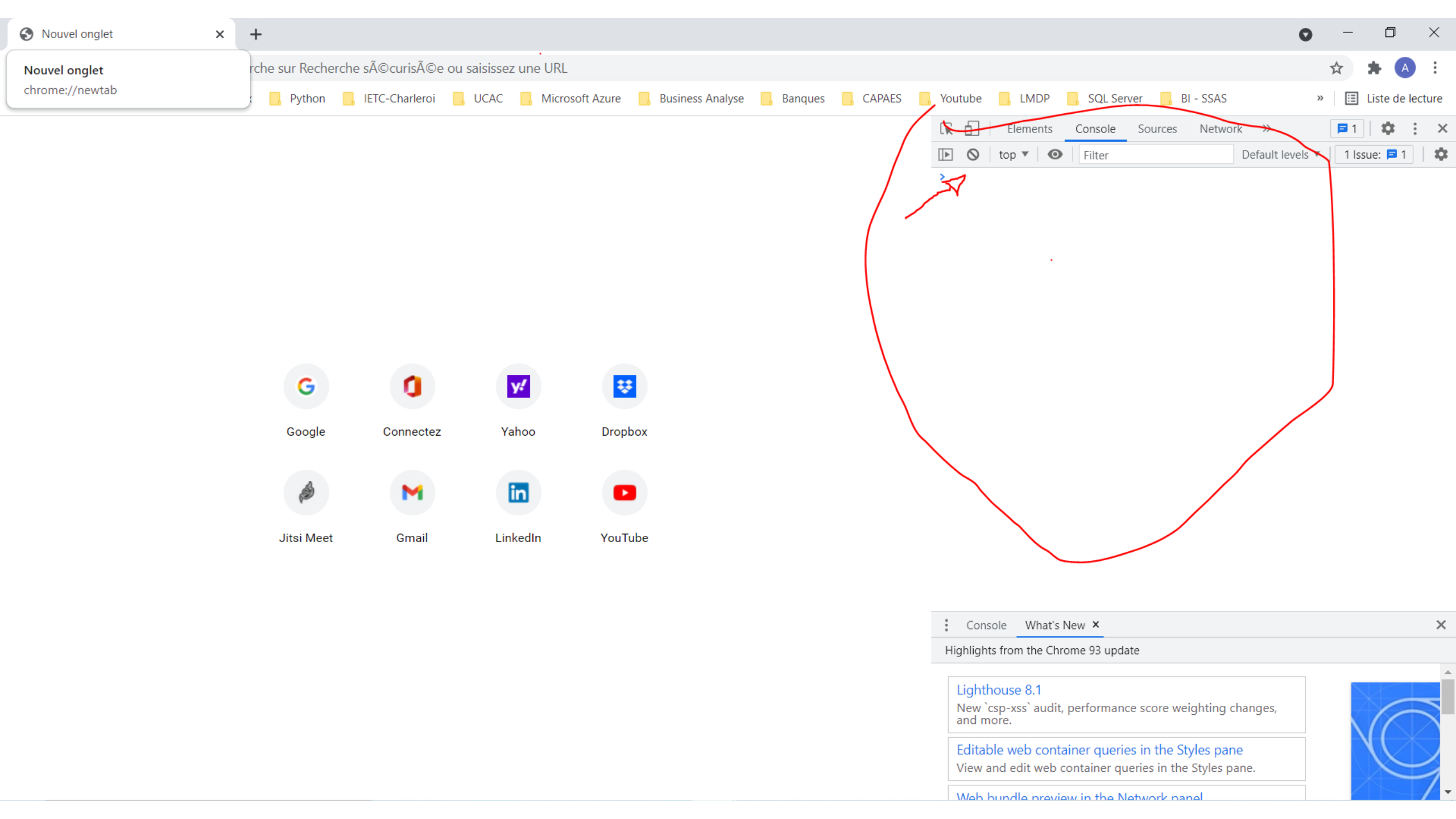
Lighthouse 8.1

New `csp-xss` audit, performance score weighting changes, and more.

Editable web container queries in the Styles pane

View and edit web container queries in the Styles pane.

Web bundle preview in the Network panel



Scripts clients - JavaScript

2. Programme JavaScript (suite)

- **Console de développement JavaScript :**

- ➔ But : pouvoir exécuter des commandes JavaScript sans créer de page HTML contenant le script ou incluant un fichier de script distinct.

- ➔ Comment accéder à cette console :

- Ouvrir Chrome, et cliquer sur le menu en haut et à droite de la fenêtre
 - Cliquer sur **Plus d'outils**, puis sur **Outils de développement**
 - Cliquer sur **Console**

- ➔ Une autre manière d'y arriver plus rapidement ➔ utiliser les raccourcis claviers :

- Windows : **Ctrl + Maj + J**
 - Mac : **Alt + Command + J**

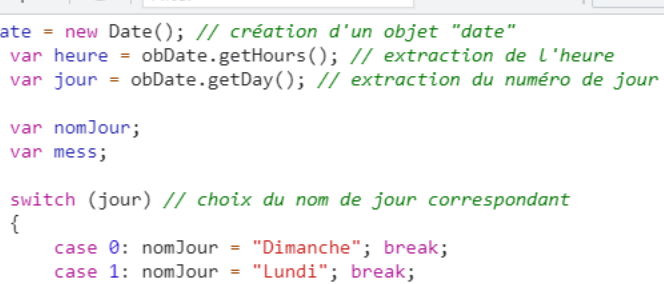
- ➔ Copier-coller le code JavaScript (se trouvant entre les balises de script) et faire **Enter**



Dropbox



YouTube



```
> var obDate = new Date(); // création d'un objet "date"
    var heure = obDate.getHours(); // extraction de l'heure
    var jour = obDate.getDay(); // extraction du numéro de jour de la
    semaine

    var nomJour;
    var mess;

    switch (jour) // choix du nom de jour correspondant
    {
        case 0: nomJour = "Dimanche"; break;
        case 1: nomJour = "Lundi"; break;
        case 2: nomJour = "Mardi"; break;
        case 3: nomJour = "Mercredi"; break;
        case 4: nomJour = "Jeudi"; break;
        case 5: nomJour = "Vendredi"; break;
        case 6: nomJour = "Samedi"; break;
        default: nomJour = "Un drôle de jour";
    }

    if (heure >= 22 || heure < 4) // choix du message d'alerte approprié
        mess = "Bonne nuit";
    else if (heure < 16)
        mess = "Bonjour";
    else
        mess = "Bonsoir";

    document.write(mess + ". ");
    document.write("<br />");
    document.write("Nous \n sommes " + nomJour + ".");
```


Bonsoir.
Nous sommes Dimanche.

```
var heure = obDate.getHours(); // extraction de l'heure
var jour = obDate.getDay(); // extraction du numéro de jour de la
semaine
var nomJour;
var mess;

switch (jour) // choix du nom de jour correspondant
{
    case 0: nomJour = "Dimanche"; break;
    case 1: nomJour = "Lundi"; break;
    case 2: nomJour = "Mardi"; break;
    case 3: nomJour = "Mercredi"; break;
    case 4: nomJour = "Jeudi"; break;
    case 5: nomJour = "Vendredi"; break;
    case 6: nomJour = "Samedi"; break;
    default: nomJour = "Un drôle de jour";
}

if (heure >= 22 || heure < 4) // choix du message d'alerte
approprié
    mess = "Bonne nuit";
else if (heure < 16)
    mess = "Bonjour";
else
    mess = "Bonsoir";

document.write(mess + ". ");
document.write("<br />");
document.write("Nous \n sommes " + nomJour + ".");
```

< undefined
> |

Console What's New

Highlights from the Chrome 93 update

[Lighthouse 8.1](#)

New `csp-xss` audit, performance score weighting changes, and more.

[Editable web container queries in the Styles pane](#)

View and edit web container queries in the Styles pane.

[Web bundle preview in the Network panel](#)

Scripts clients - JavaScript

2. Programme JavaScript (suite)

- **Console de développement JavaScript :**

- ➔ Intérêts d'utiliser la console JavaScript :

- possibilité de tester et exécuter rapidement et facilement du code JavaScript
 - signalisation des erreurs (la console possède des fonctionnalités qui aident à localiser et à résoudre les problèmes posés par le code.

Scripts clients - JavaScript

2. Programme JavaScript (suite)

- **Commenter le code**
 - ➔ But des commentaires :
 - expliquer ce que fait le code
 - clarifier, expliciter votre démarche
 - noter des nouvelles idées à développer
 - pour le testing : isoler des parties de code lors de l'exécution
 - ➔ Le moteur de JavaScript ignore totalement les commentaires. Ces derniers ne servent que pour les humains.
 - ➔ Commentaire sur 1 seule ligne : `//`
 - ➔ Commentaires multilignes :
 - début du commentaire : `/*`
 - fin du commentaire : `*/`

Scripts clients - JavaScript

3. Les variables

3.1 Créer et utiliser des variables

3.2 Comprendre la portée des variables

3.3 Nommer les variables

3.4 Connaître les types de données de JavaScript

3.5 Utiliser des fonctions intégrées pour travailler avec les variables

Scripts clients - JavaScript

3. Les variables

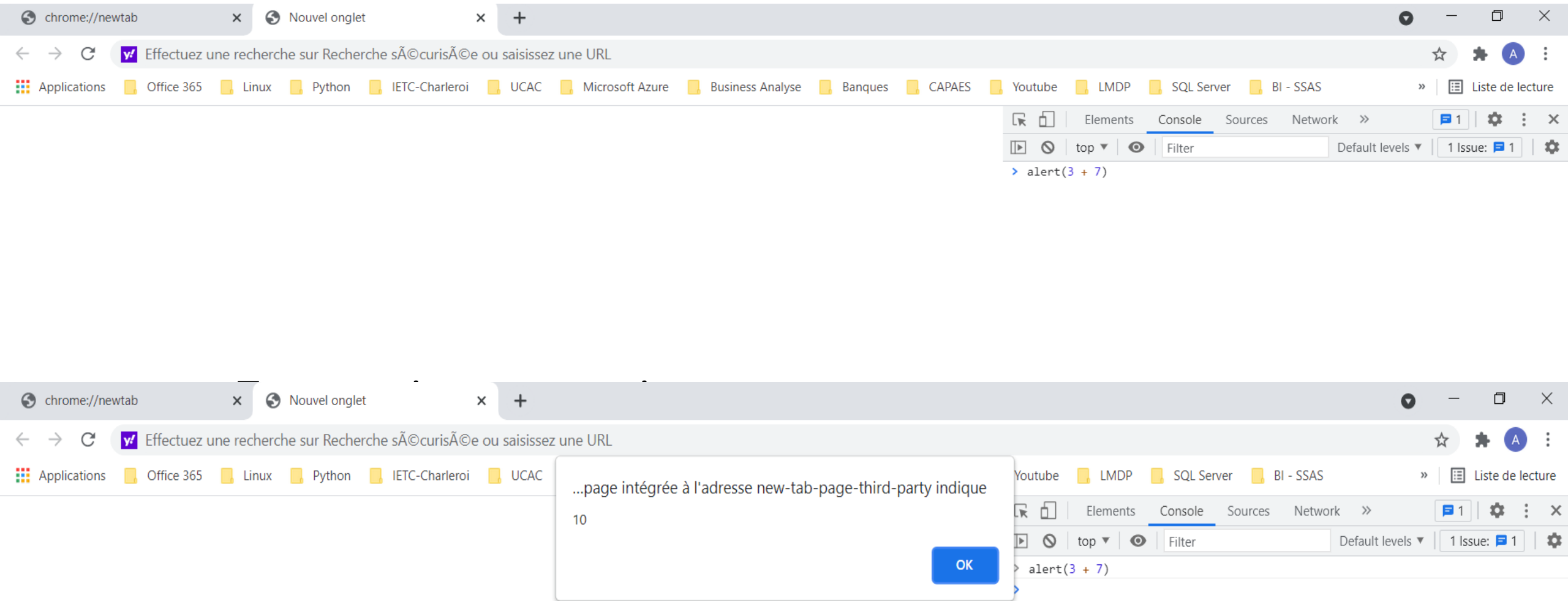
3.1 Créer et utiliser des variables

- ➔ Une variable est un conteneur dans lequel se trouve une (des) donnée(s)
- ➔ Un nom doit être attribué à ce conteneur, et l'on peut ainsi en récupérer le contenu, ou y placer autre chose
- ➔ Sans les variables, un programme informatique ne servirait pas à grand-chose, sinon, à effectuer toujours le même calcul ou à afficher toujours le même texte

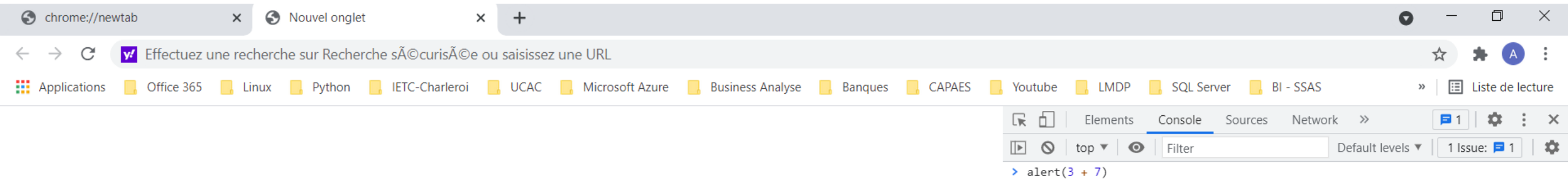
Exemple (taper cela dans la console JavaScript):

alert(3 + 7);

Scripts clients - JavaScript



Scripts clients - JavaScript



Tapez maintenant ceci :

```
var firstNumber = 3;
```

```
var secondNumber = 7;
```

```
var total = Number(firstNumber) + Number(secondNumber);
```

```
alert(total);
```

Scripts clients - JavaScript

The image shows a Chrome browser window with two tabs: 'chrome://newtab' and 'Nouvel onglet'. The address bar contains the text 'Effectuez une recherche sur Recherche sÃ©curisÃ©e ou saisissez une URL'. Below the address bar, there are several bookmarks including 'Applications', 'Office 365', 'Linux', 'Python', 'IETC-Charleroi', 'UCAC', 'Microsoft Azure', 'Business Analyse', 'Banques', 'CAPAES', 'Youtube', 'LMDP', 'SQL Server', and 'BI - SSAS'. The right side of the browser window shows the 'Elements', 'Console', 'Sources', and 'Network' panels. The 'Console' panel is active, displaying the following JavaScript code:

```
> alert(3 + 7)
< undefined
> var firstNumber = 3;
  var secondNumber = 7;
  var total = Number(firstNumber) + Number(secondNumber);
  alert(total);
```

An alert dialog box is displayed in the foreground with the text: "...page intÃ©grÃ©e Ã l'adresse new-tab-page-third-party indique 10". The dialog has an 'OK' button.

Scripts clients - JavaScript

3. Les variables

3.1 Créer et utiliser des variables

Pour utiliser bien utiliser les variables, demandons à l'utilisateur de saisir les 2 valeurs :

```
var firstNumber = prompt(« Entrez le premier nombre »);
```

```
var secondNumber = prompt(« Entrez le second nombre »);
```

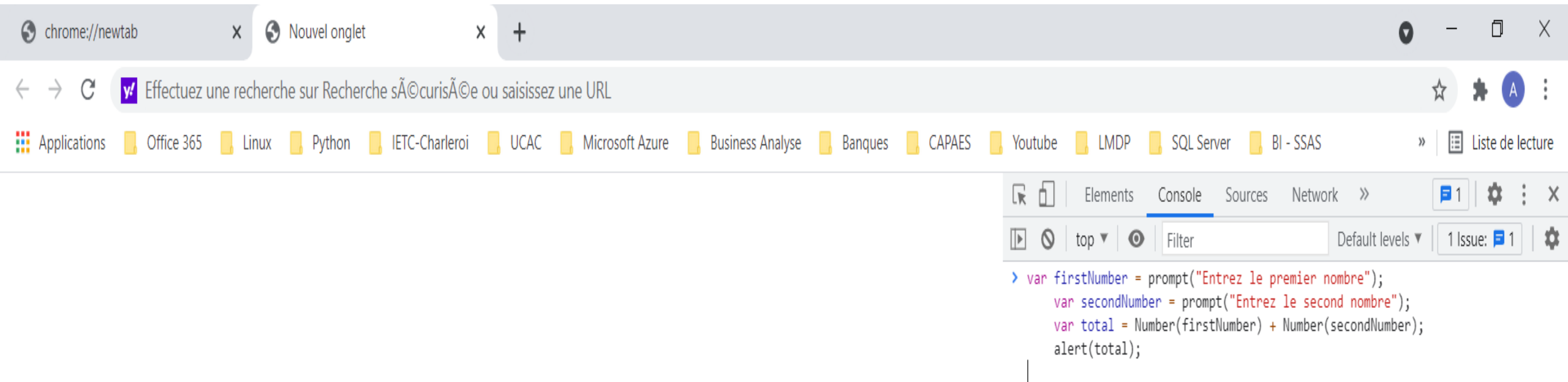
```
var total = Number(firstNumber) + Number(secondNumber);
```

```
alert(total);
```

Scripts clients - JavaScript

3. Les variables

3.1 Créer et utiliser des variables



chrome://newtab x Nouvel onglet x +

Effectuez une recherche sur Recherche sÃ©curisÃ©e ou saisissez une URL

Applications Office 365 Linux Python IETC-Charleroi UCAC

Youtube LMDP SQL Server BI - SSAS » Liste de lecture

Elements Console Sources Network » 1 Issue: 1

...page intégrée à l'adresse new-tab-page-third-party indique

Entrez le premier nombre

6

OK Annuler

```
var firstNumber = prompt("Entrez le premier nombre");
var secondNumber = prompt("Entrez le second nombre");
var total = Number(firstNumber) + Number(secondNumber);
alert(total);
```

chrome://newtab x Nouvel onglet x +

Effectuez une recherche sur Recherche sÃ©curisÃ©e ou saisissez une URL

Applications Office 365 Linux Python IETC-Charleroi UCAC

Youtube LMDP SQL Server BI - SSAS » Liste de lecture

Elements Console Sources Network » 1 Issue: 1

...page intégrée à l'adresse new-tab-page-third-party indique

Entrez le second nombre

8

OK Annuler

```
var firstNumber = prompt("Entrez le premier nombre");
var secondNumber = prompt("Entrez le second nombre");
var total = Number(firstNumber) + Number(secondNumber);
alert(total);
```

chrome://newtab x Nouvel onglet x +

Effectuez une recherche sur Recherche sÃ©curisÃ©e ou saisissez une URL

Applications Office 365 Linux Python IETC-Charleroi UCAC

Youtube LMDP SQL Server BI - SSAS » Liste de lecture

Elements Console Sources Network » 1 Issue: 1

...page intégrée à l'adresse new-tab-page-third-party indique

14

OK

```
var firstNumber = prompt("Entrez le premier nombre");
var secondNumber = prompt("Entrez le second nombre");
var total = Number(firstNumber) + Number(secondNumber);
alert(total);
```

Scripts clients - JavaScript

3. Les variables

3.1 Créer et utiliser des variables

- Ouvrez votre éditeur et créez un modèle HTML de base
- Entre les balises **<body>** et **</body>**, insérer une balise **<script>** et une fermante **</script>**
- Saisissez ce code entre les balises de script :

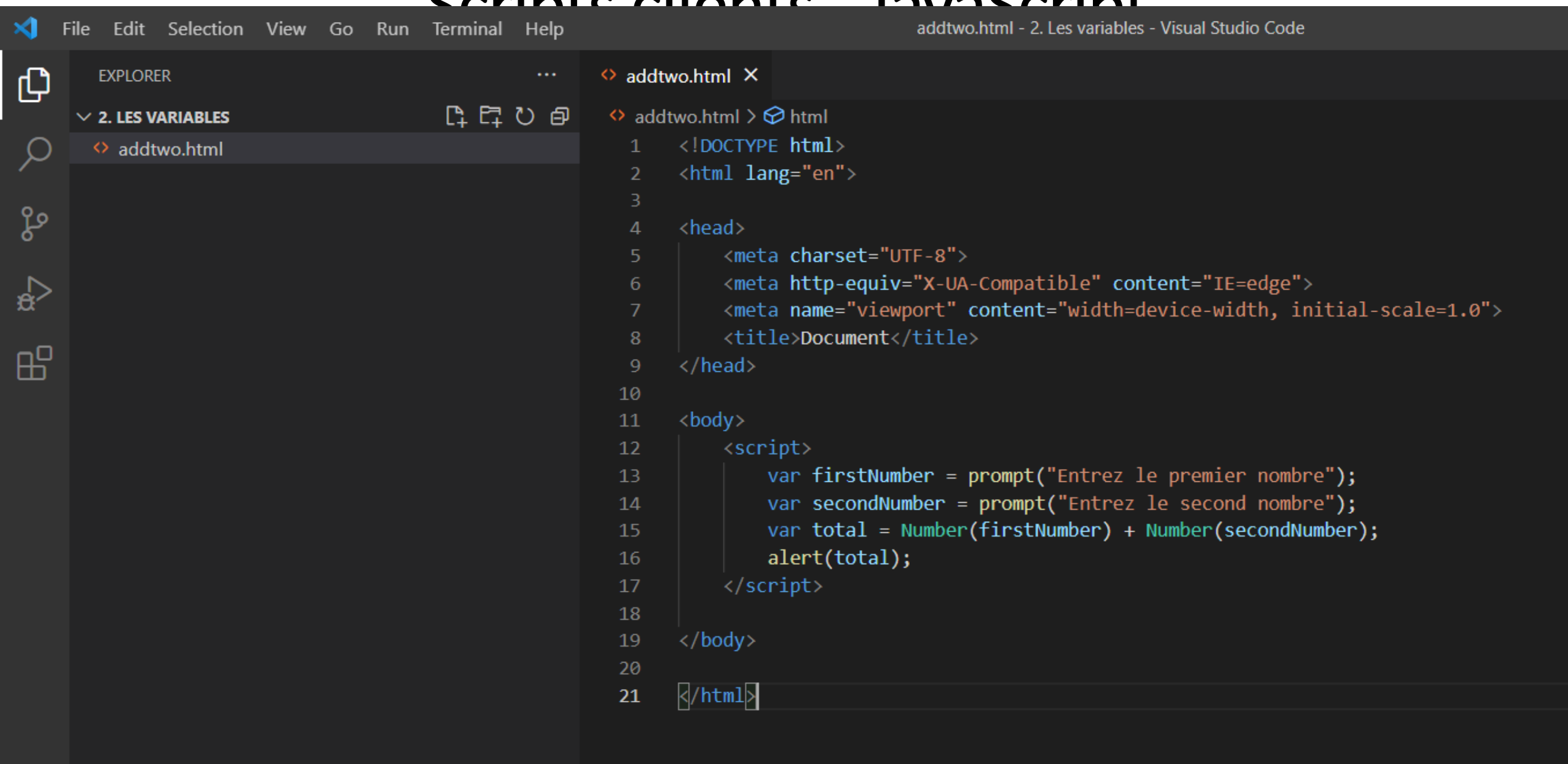
```
var firstNumber = prompt("Entrez le premier nombre");
```

```
var secondNumber = prompt("Entrez le second nombre");
```

```
var total = Number(firstNumber) + Number(secondNumber);
```

```
alert(total);
```

Scripts clients - JavaScript



```
File Edit Selection View Go Run Terminal Help
addtwo.html - 2. Les variables - Visual Studio Code

EXPLORER
2. LES VARIABLES
  addtwo.html

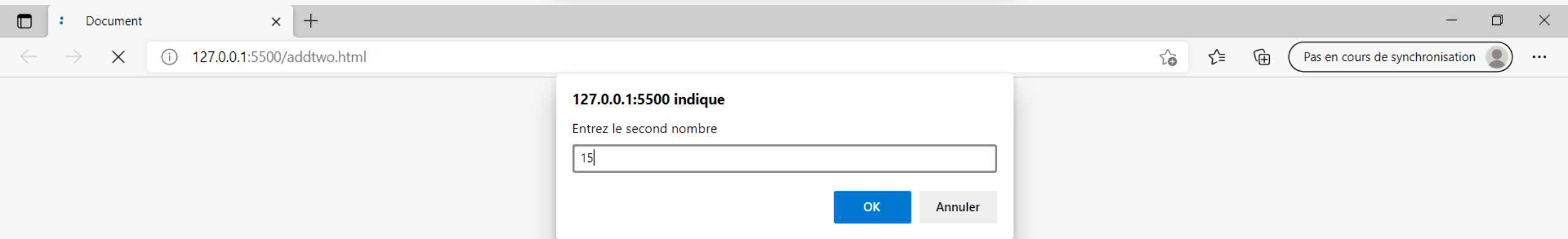
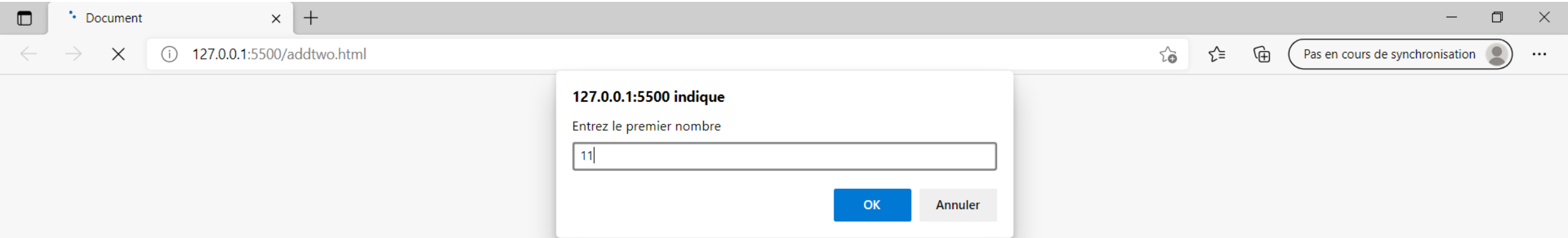
addtwo.html
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>Document</title>
9  </head>
10
11 <body>
12     <script>
13         var firstNumber = prompt("Entrez le premier nombre");
14         var secondNumber = prompt("Entrez le second nombre");
15         var total = Number(firstNumber) + Number(secondNumber);
16         alert(total);
17     </script>
18
19 </body>
20
21 </html>
```

Scripts clients - JavaScript

3. Les variables

3.1 Créer et utiliser des variables

- Tapez **AdditionDeuxChiffres** entre les balises **title**
- Sauvegardez le document (sous le nom **addtwo.html**)
- Ouvrez le document dans le navigateur
 - le navigateur demande de saisir le premier chiffre;
Validez
 - le navigateur demande de saisir le deuxième chiffre;
Validez
 - Cliquez **OK** pour avoir la réponse



Scripts clients - JavaScript

3. Les variables

3.1 Créer et utiliser des variables

➔ Déclaration des variables ➔ c'est le processus par lequel on crée une nouvelle variable dans un programme.

- 2 manières de déclarer une variable :

a) utiliser le mot-clé **var** :

var maVariable;

➔ une variable déclarée de la sorte aura une valeur initiale non définie

➔ Si on veut l'initialiser : **var maVariable = "Steph";**

Scripts clients - JavaScript

3. Les variables

3.1 Créer et utiliser des variables

b) sans le mot-clé **var** :

maVariable = "Steph";

→ une variable déclarée de la sorte sera une **variable globale**

➔ L'utilisation des apostrophes indique que la variable doit être traitée comme du texte, et non pas comme un nombre, un mot-clé JavaScript ou une tout autre variable.

Remarque : variables dont la valeur ne change pas tout au long de leur vie ➔ **constantes**

Elles sont définies par le mot-clé **const**

Exemple : *const nombreJoueurs = 20*

Scripts clients - JavaScript

3. Les variables

3.2 Comprendre la portée des variables

- ➔ La portée d'une variable est déterminée par la manière et le moment quand on déclare cette variable
- ➔ JavaScript comprend 2 types de portée :
 - les **variables globales** : peuvent être utilisées partout dans un programme
 - les **variables locales** : sont créées et ne sont utilisées qu'à l'intérieur d'une **fonction** (ie, une section protégée contenue dans le programme)

Scripts clients - JavaScript

3. Les variables

3.2 Comprendre la portée des variables

➔ Avantage des *variables locales* :

- leur portée plus limitée réduit les risques de remplacement accidentel du contenu d'une variable par celui d'une autre variable portant le même nom.
- l'emploi des *variables globales* peut rendre les problèmes difficiles à localiser et à réparer.

Scripts clients - JavaScript

3. Les variables

3.3 Nommer les variables

➔ Quelques règles :

- commencer le nom d'une variable par :
 - une lettre (majuscule ou minuscule) : recommandé !
 - un trait de soulignement « _ »
 - un signe dollar « \$ »
- ne pas mettre :
 - des espaces blancs
 - des opérateurs mathématiques
 - des signes de ponctuation

Scripts clients - JavaScript

3. Les variables

3.3 Nommer les variables

➔ Quelques règles :

- JavaScript est case sensitive (prise en compte de la différence entre majuscules et minuscules)
- Les noms des variables doivent être les plus explicites possible, mais éviter de trop longs noms (recommandation : pas plus de 20 caractères).
- Définissez vos règles de nommage des variables, et gardez-les:
 - caractère de début ?
 - underscore ou Camel Case, entre 2 mots du nom de la variable ?
 - où placer un adjectif décrivant le nom d'une variable (***bleuePersonne*** ou ***personneBleue***) ?

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- ➔ Type de données = le genre d'information que contient la variable
- ➔ Cela détermine ainsi les opérations qui peuvent être faites sur cette variable. Exemple :

alert("Ils sont 7 + 3") ; ➔ 7 et 3 pris comme textes

alert(7 + 3); ➔ 7 et 3 pris comme nombres

Connectez-vous à votre compte

javascript number() - Recherche

mbayo_a@yahoo.fr - Yahoo Mail

Nouvel onglet

Effectuez une recherche sur Recherche s@curis@e ou saisissez une URL

ApplicationsOffice 365LinuxPythonIETC-CharleroiUCACMicrosoft AzureBusiness AnalyseBanquesCAPAESYoutubeLMDPSQL ServerBI - SSAS

Liste de lecture

ElementsConsoleSourcesNetwork

topFilter

Default levels1 Issue

alert("ils sont 7+3")

...page intégrée à l'adresse new-tab-page-third-party indique
ils sont 7+3

OK

Connectez-vous à votre compte

javascript number() - Recherche

mbayo_a@yahoo.fr - Yahoo Mail

Nouvel onglet

Effectuez une recherche sur Recherche s@curis@e ou saisissez une URL

ApplicationsOffice 365LinuxPythonIETC-CharleroiUCACMicrosoft AzureBusiness AnalyseBanquesCAPAESYoutubeLMDPSQL ServerBI - SSAS

Liste de lecture

ElementsConsoleSourcesNetwork

topFilter

Default levels1 Issue

alert("ils sont 7+3")

undefined

alert(3 + 7)

...page intégrée à l'adresse new-tab-page-third-party indique
10

OK

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

➔ Type de données (définition 1) = *le genre d'information que contient la variable*

➔ Cela détermine ainsi les opérations qui peuvent être faites sur cette variable. Exemple :

`alert("Ils sont 7 + 3");` ➔ 7 et 3 pris comme textes

`alert(7 + 3);` ➔ 7 et 3 pris comme nombres

➔ Type de données (définition 2) = *la manière dont JavaScript interprète ce qui doit être pris pour un mot, de ce qui doit être considéré comme une expression mathématique*

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- ➔ JavaScript = langage *pauvrement typé*, ie, pas besoin de connaître à l'avance quel type de donnée contiendra la variable.
- ➔ JavaScript comprend seulement 5 types de données (types primitifs):
 - Données numériques
 - Données de type chaîne
 - Données de type booléen
 - Données de type NaN
 - Données de type indéfini

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données numériques

- pas de différence entre entiers et nombres
- tous les nombres sont enregistrés en virgule flottante sur 64 bits
- la déclaration d'une variable numérique comprendra toujours :
 - le mot-clé **var**
 - le nom de la variable
 - l'opérateur d'affectation (« = »)
 - un nombre (ou une équation) d'initialisation
 - un point-virgule (« ; »)

Scripts clients - JavaScript

3. Les variables

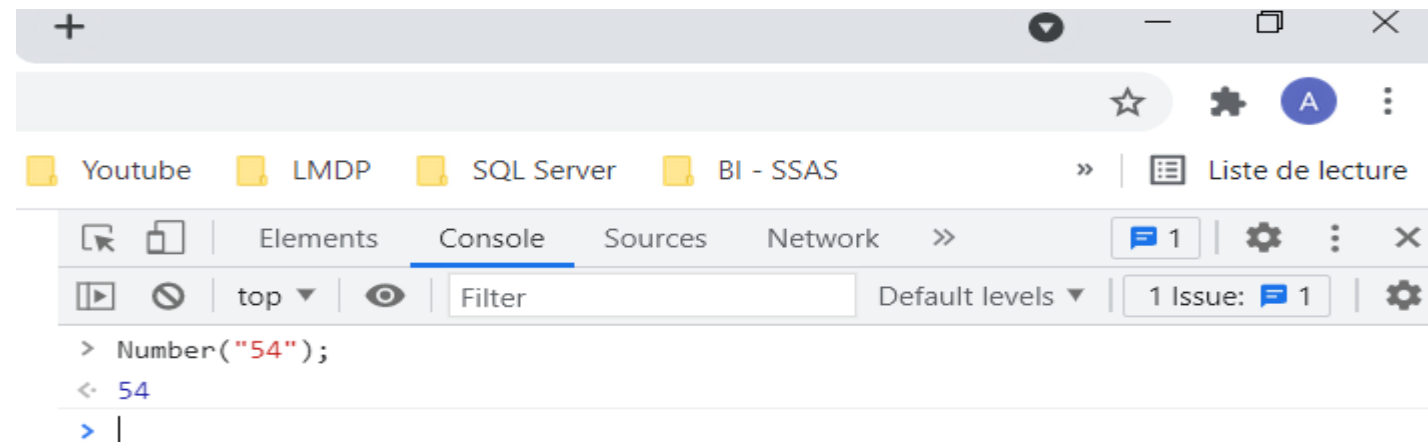
3. 4 Connaître les types de données de JavaScript

- Données numériques

- quelques fonctions intégrées utiles :

- ***Number()*** → convertit :

- les nombres formatés en chaînes de caractères en nombre



Scripts clients - JavaScript

3. Les variables

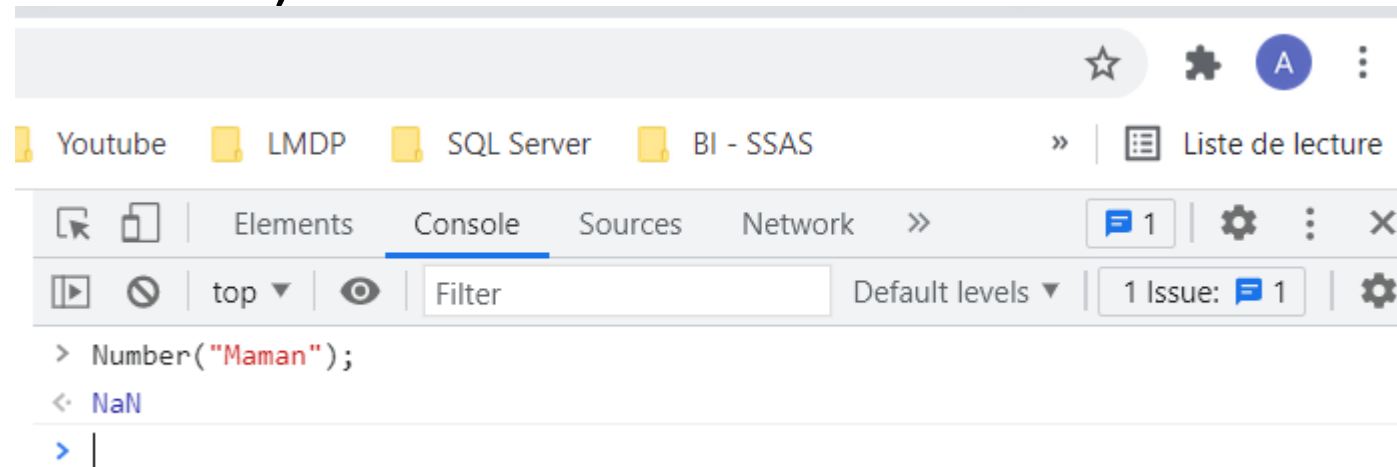
3. 4 Connaître les types de données de JavaScript

- Données numériques

- quelques fonctions intégrées utiles :

- *Number()* → convertit :

- les chaînes de caractères en la valeur **NaN** (**Not A Number**)



Scripts clients - JavaScript

3. Les variables

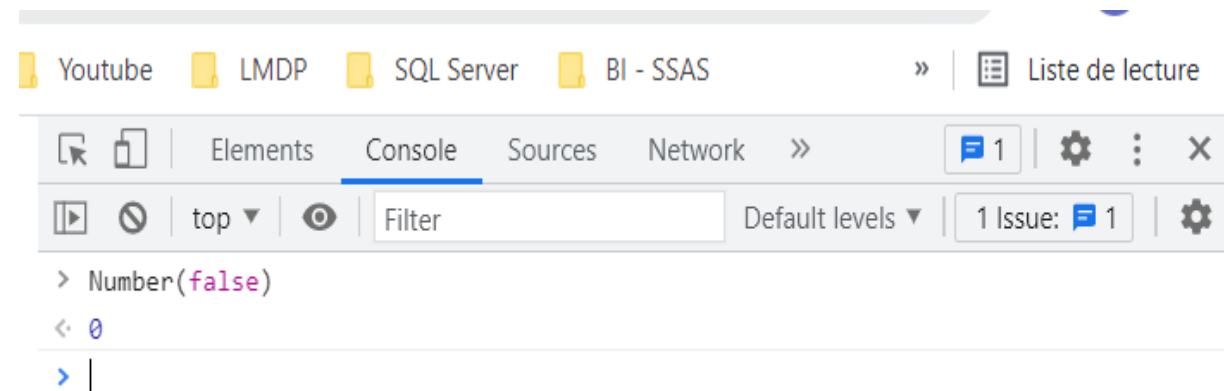
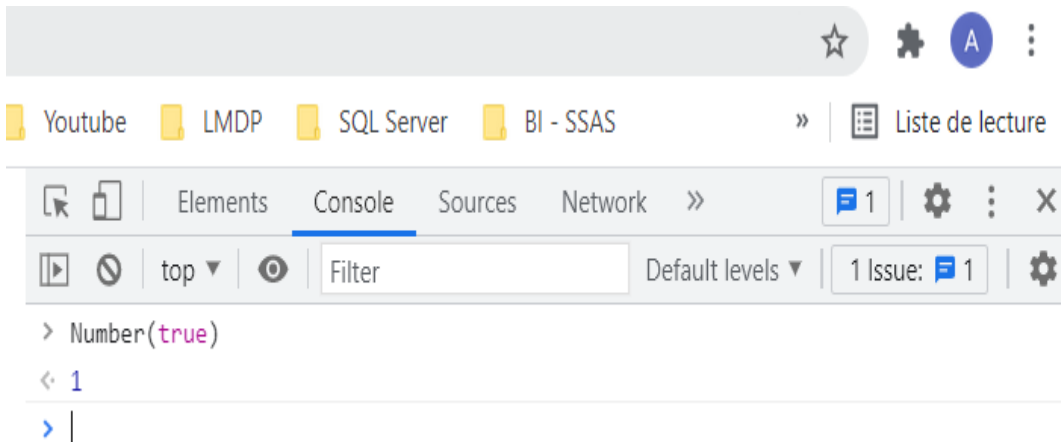
3. 4 Connaître les types de données de JavaScript

- Données numériques

- quelques fonctions intégrées utiles :

- *Number()* → convertit :

- les booléennes (true → 1, et false → 0)



Scripts clients - JavaScript

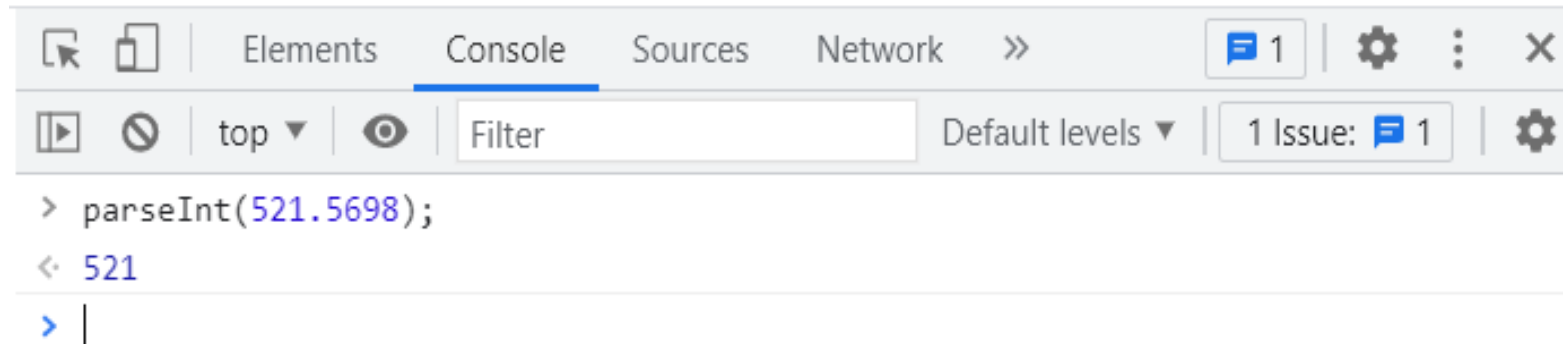
3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données numériques

- quelques fonctions intégrées utiles :

- *parseInt()* → retourne la partie entière d'un nombre



Scripts clients - JavaScript

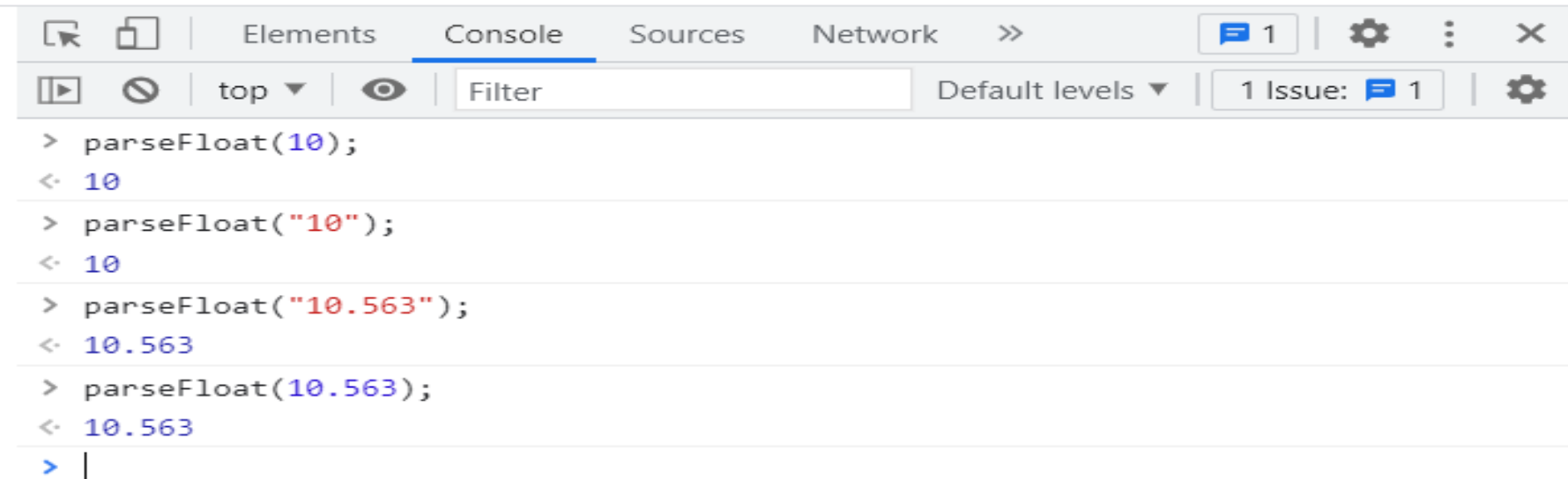
3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données numériques

- quelques fonctions intégrées utiles :

- *parseFloat()* → traite un nombre comme étant un décimal



```
> parseFloat(10);  
< 10  
  
> parseFloat("10");  
< 10  
  
> parseFloat("10.563");  
< 10.563  
  
> parseFloat(10.563);  
< 10.563  
> |
```

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données numériques

- Exercices :

- additionnez (dans la console) et expliquez :

- $12 + "12"$

- $"12" + 12$

- $"12" * 2$

Scripts clients - JavaScript

3. Les variables

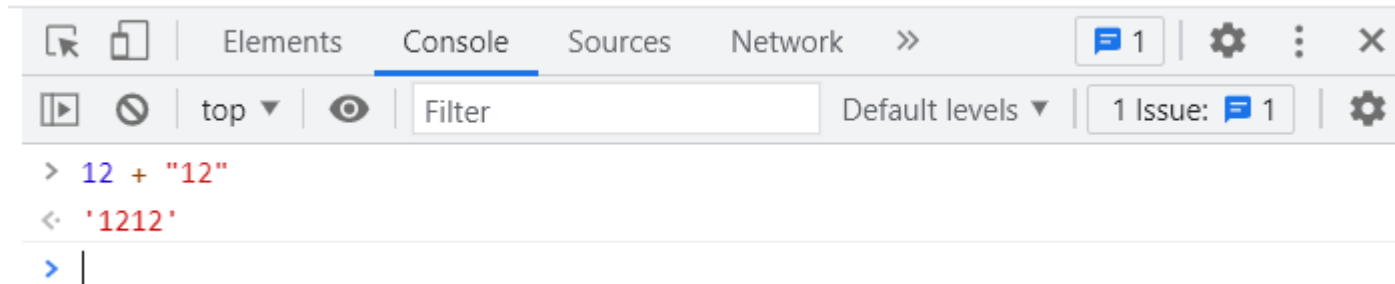
3. 4 Connaître les types de données de JavaScript

- Données numériques

- Exercices :

- additionnez (dans la console) et expliquez :

- $12 + "12"$ → le moteur JavaScript interprète « + » comme étant une concaténation, à cause de "12"



Scripts clients - JavaScript

3. Les variables

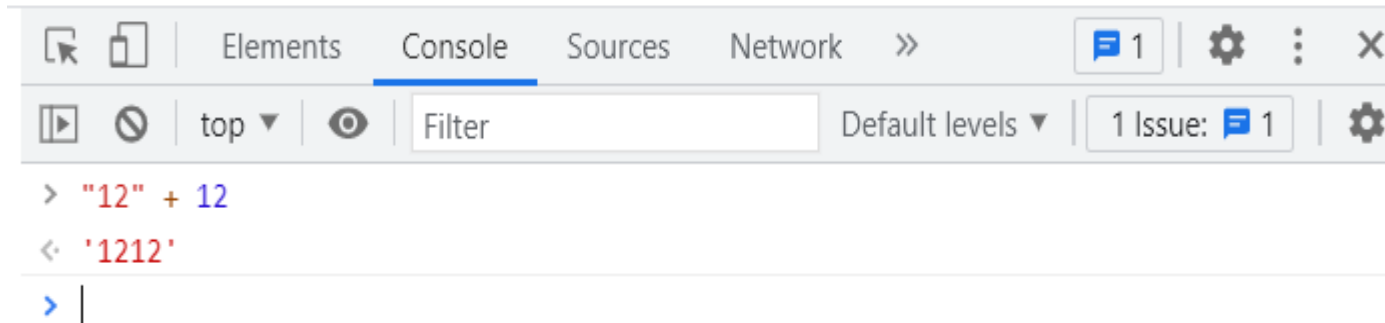
3. 4 Connaître les types de données de JavaScript

- Données numériques

- Exercices :

- additionnez (dans la console) et expliquez :

- "12" + 12 → idem que ci-haut !



Scripts clients - JavaScript

3. Les variables

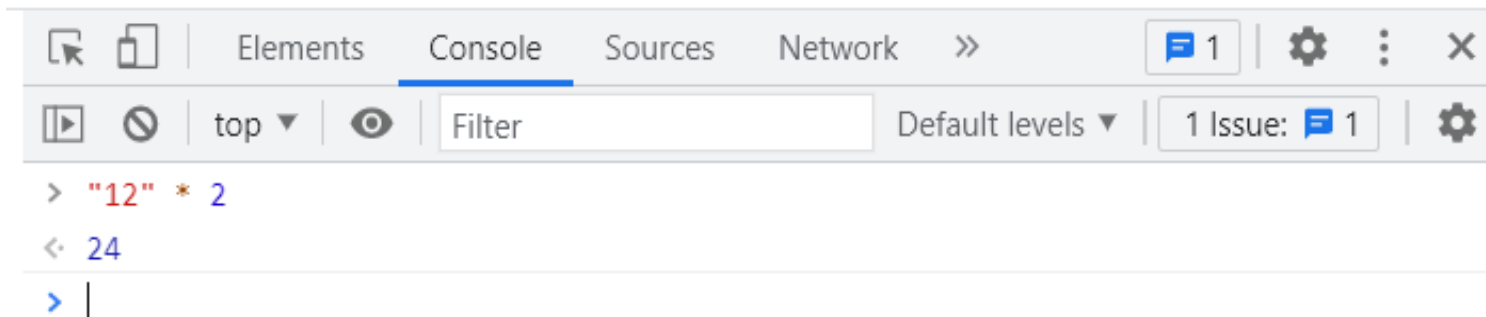
3. 4 Connaître les types de données de JavaScript

- Données numériques

- Exercices :

- additionnez (dans la console) et expliquez :

- "12" * 2 → le moteur JavaScript sait qu'on ne peut pas multiplier 2 mots → il convertit "12" en nombre, et applique la multiplication



Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type chaîne
 - Une chaîne peut contenir :
 - des lettres
 - des chiffres
 - des signes de ponctuation (virgule, point)
 - des caractères spéciaux (qui devront alors être précédés de la barre oblique inversée → caractère d'échappement)

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type chaîne

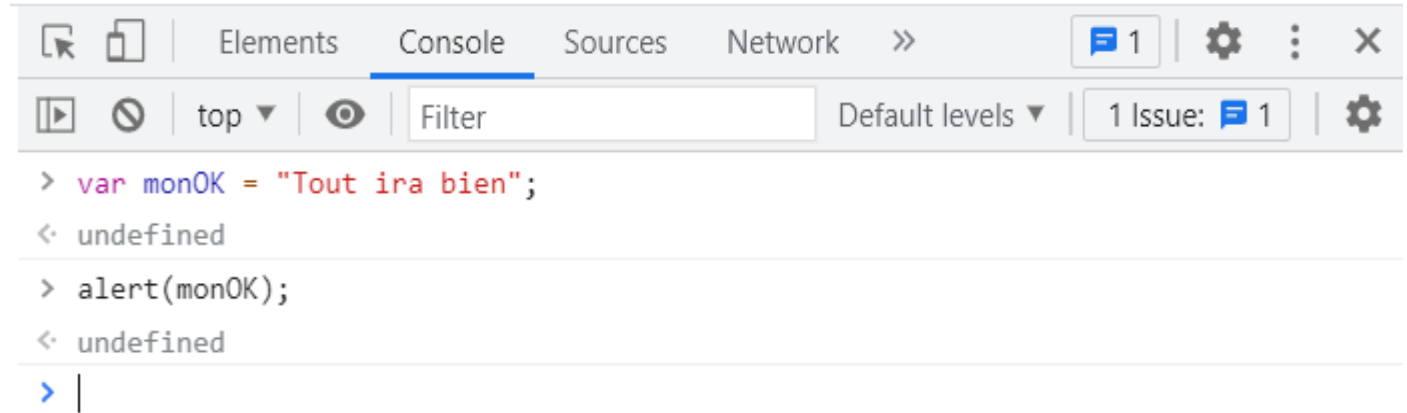
Caractères spéciaux	
Code	Signification
\'	apostrophe
\"	Guillement
\\	Barre oblique inverse
\n	Nouvelle ligne
\r	Retour chariot
\t	Tabulation
\b	Retour arrière
\f	Saut de page

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type chaîne
 - Exemples :



The screenshot shows the developer console with the following code and output:

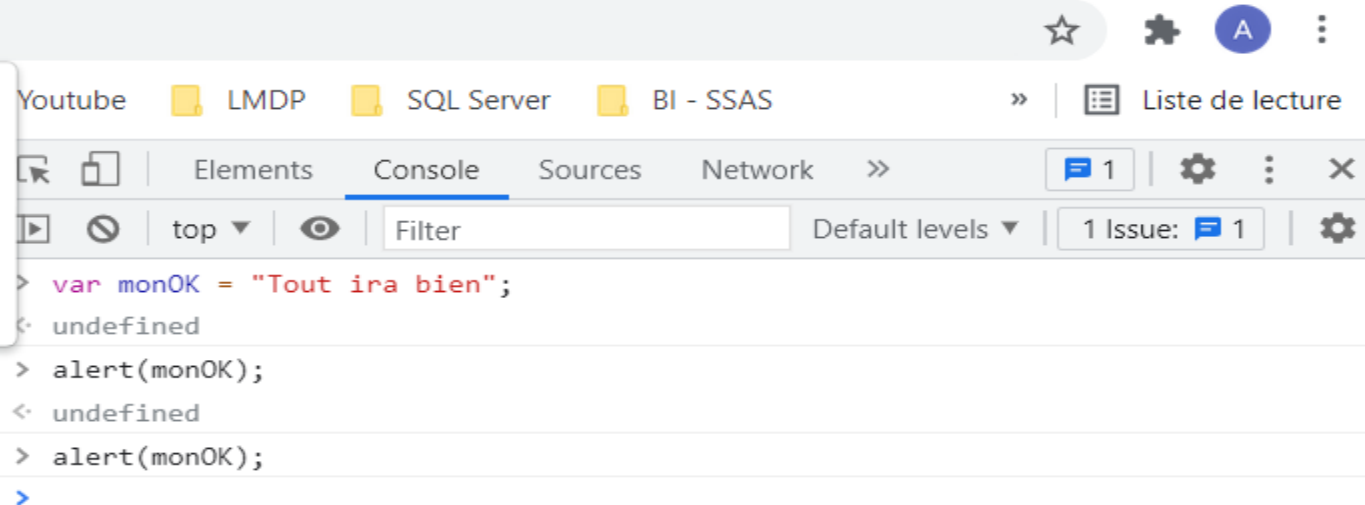
```
> var monOK = "Tout ira bien";  
< undefined  
> alert(monOK);  
< undefined  
> |
```

sez une URL

...page intégrée à l'adresse new-tab-page-third-party indique

Tout ira bien

OK



The screenshot shows the developer console with the following code and output:

```
> var monOK = "Tout ira bien";  
< undefined  
> alert(monOK);  
< undefined  
> alert(monOK);  
< undefined  
> |
```

Scripts clients - JavaScript

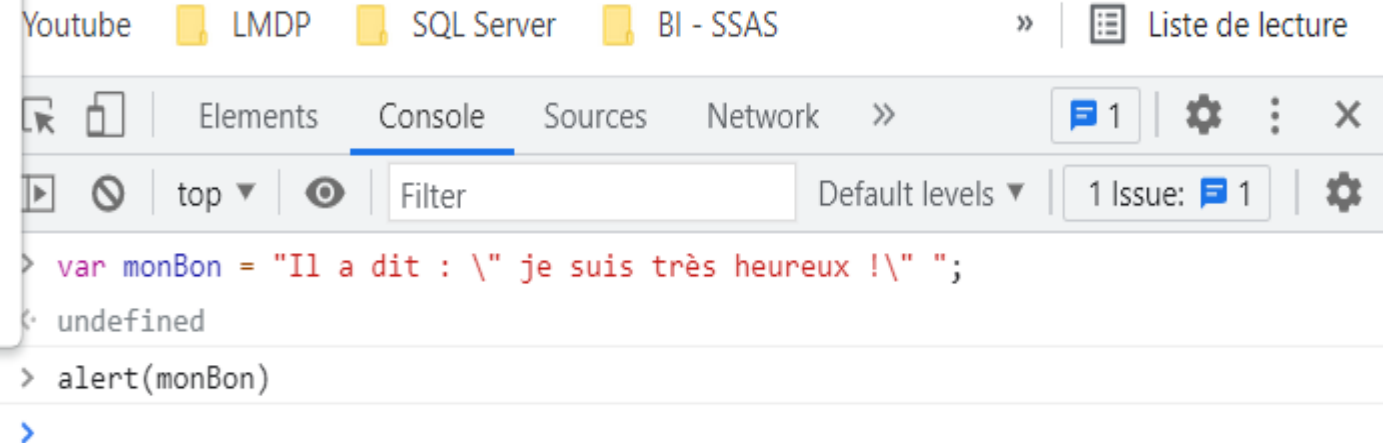
3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type chaîne
 - Exemples :

...page intégrée à l'adresse new-tab-page-third-party indique
Il a dit : " je suis très heureux !"

OK

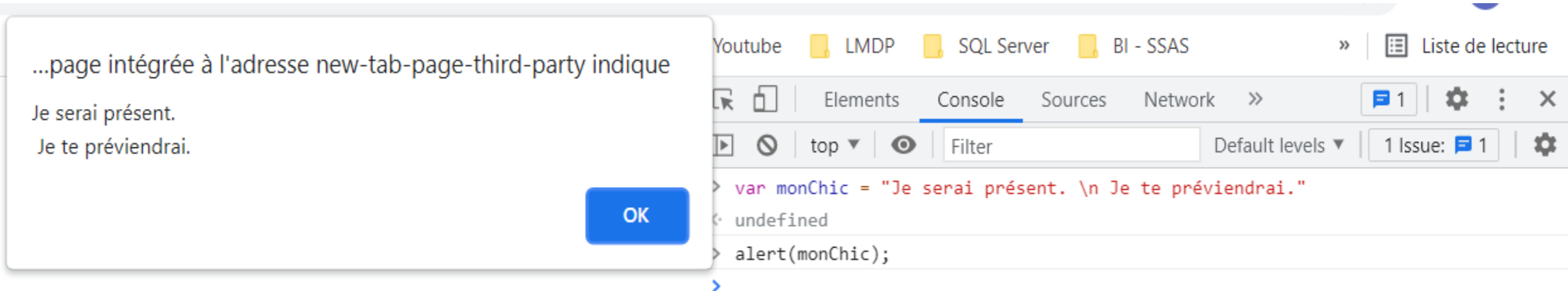


Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type chaîne
 - Exemples :



Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

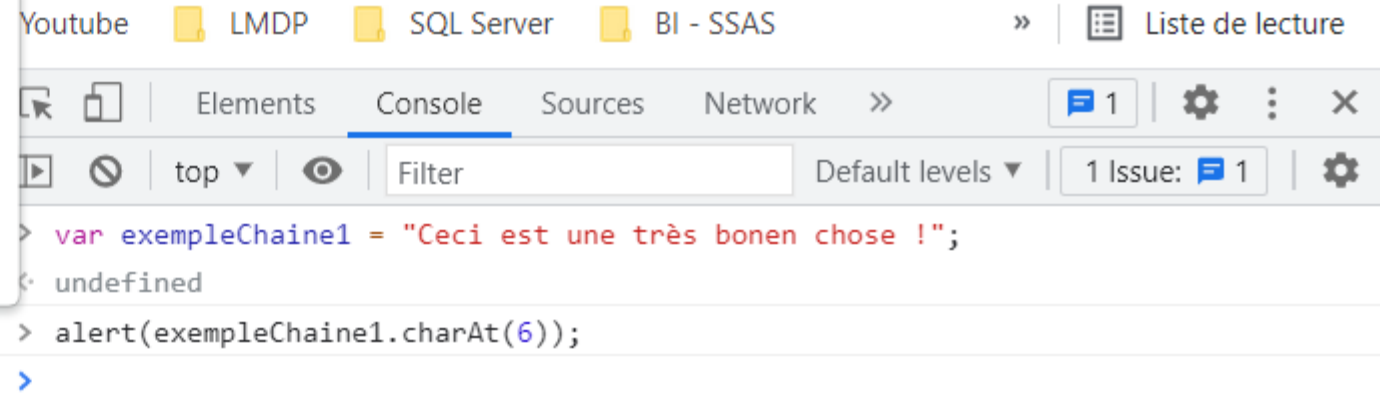
- Données de type chaîne
 - Quelques fonctions intégrées de chaîne :
 - *charAt()* → renvoie le caractère qui se trouve à la position indiquée
- JavaScript commence à compter à la position **0** et non à **1**

Scripts clients - JavaScript

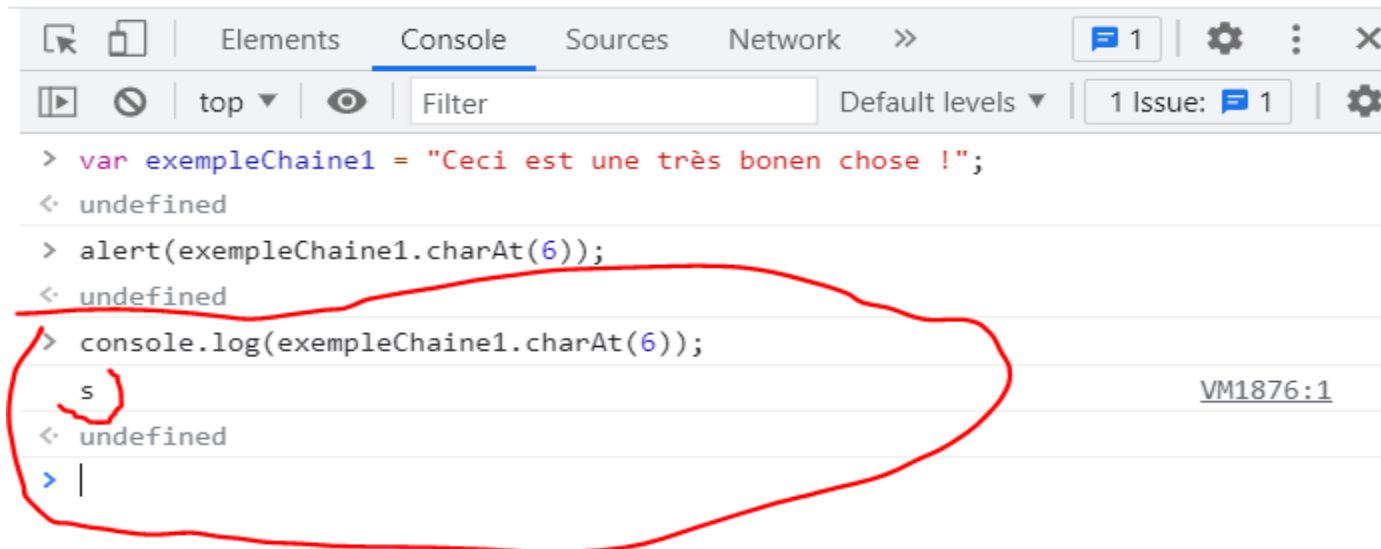
...page intégrée à l'adresse new-tab-page-third-party indique

s

OK



Ou encore, avec la méthode « console.log », qui affiche un message dans la console :

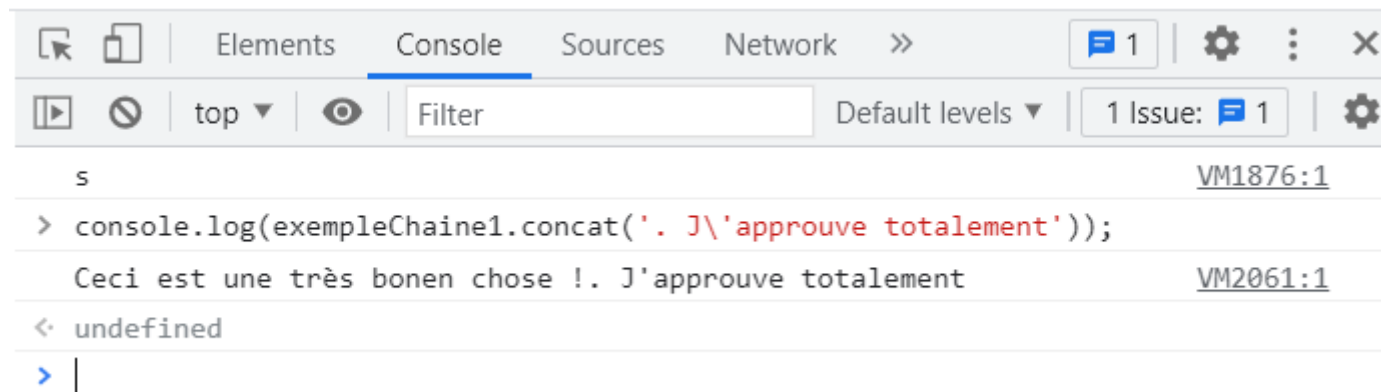


Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type chaîne
 - Quelques fonctions intégrées de chaîne :
 - *concat()* → combine une ou plusieurs chaînes en une nouvelle chaîne



```
s VM1876:1
> console.log(exempleChaine1.concat('. J\'approuve totalement'));
Ceci est une très bonen chose !. J'approuve totalement VM2061:1
< undefined
> |
```

Scripts clients - JavaScript

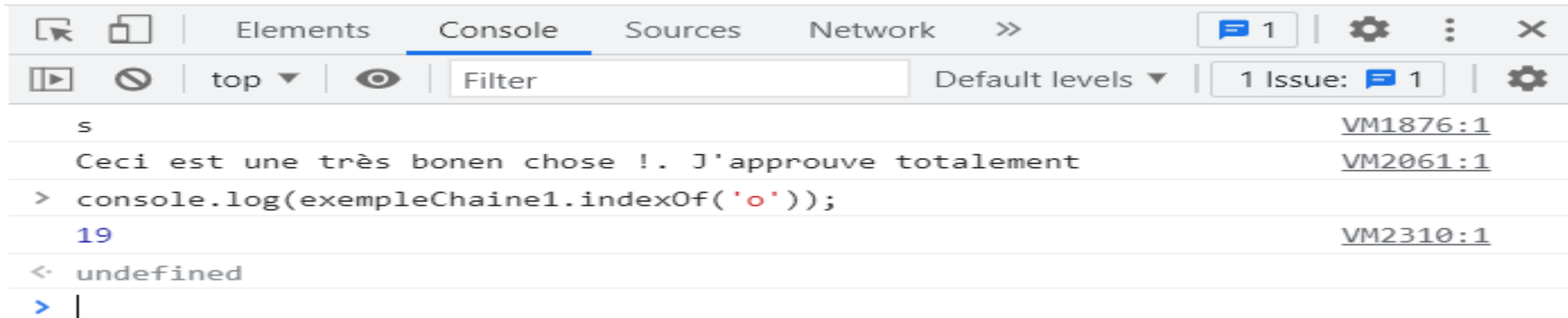
3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type chaîne
 - Quelques fonctions intégrées de chaîne :
 - *indexOf()* → recherche et renvoie la position de la première occurrence du caractère ou de la sous-chaîne recherchée

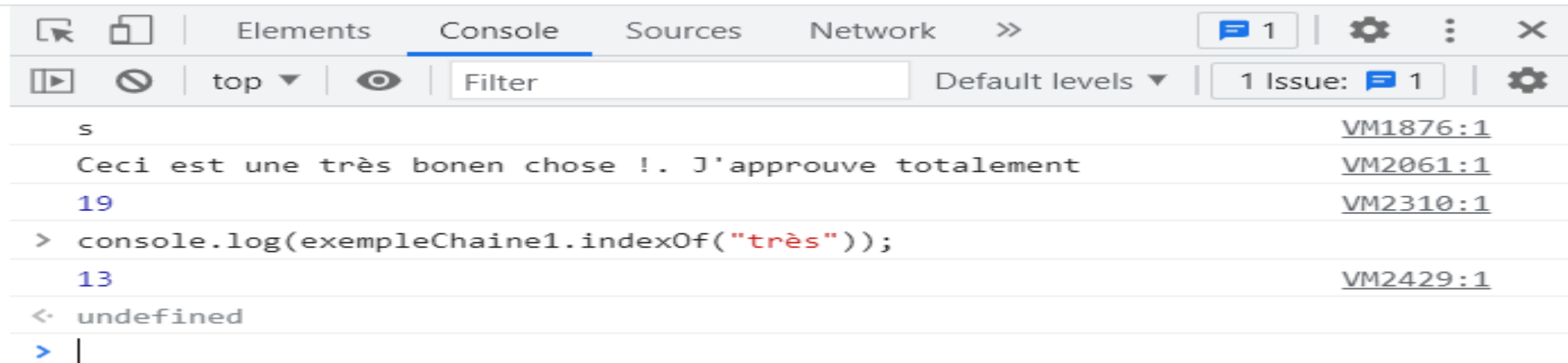
Scripts clients - JavaScript

3. Les variables



The screenshot shows the Chrome DevTools Console with the 'Console' tab selected. The console contains the following log entries:

```
s VM1876:1
Ceci est une très bonen chose !. J'approuve totalement VM2061:1
> console.log(exempleChaine1.indexOf('o'));
19 VM2310:1
< undefined
> |
```



The screenshot shows the Chrome DevTools Console with the 'Console' tab selected. The console contains the following log entries:

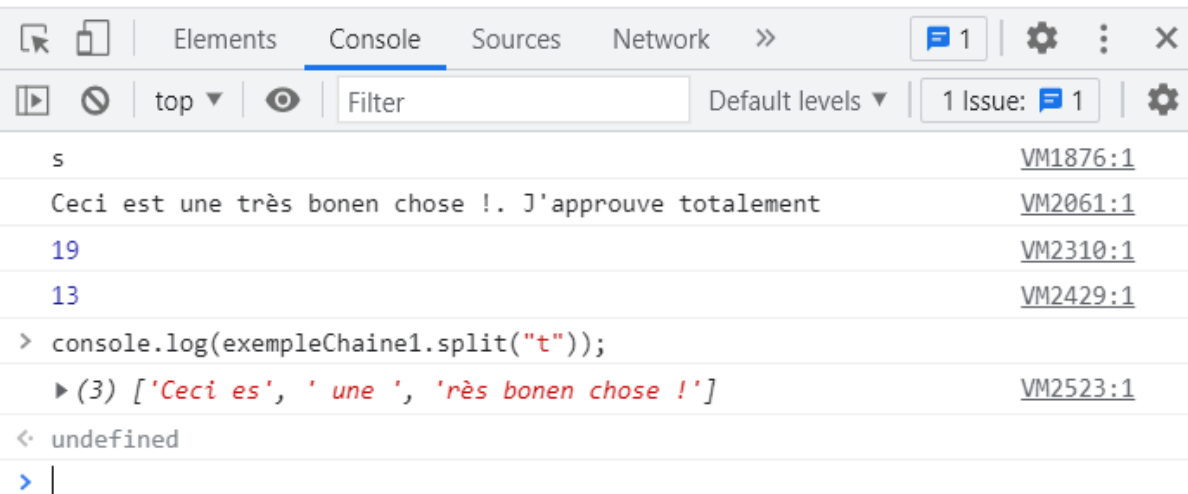
```
s VM1876:1
Ceci est une très bonen chose !. J'approuve totalement VM2061:1
19 VM2310:1
> console.log(exempleChaine1.indexOf("très"));
13 VM2429:1
< undefined
> |
```

Scripts clients - JavaScript

3. Les variables

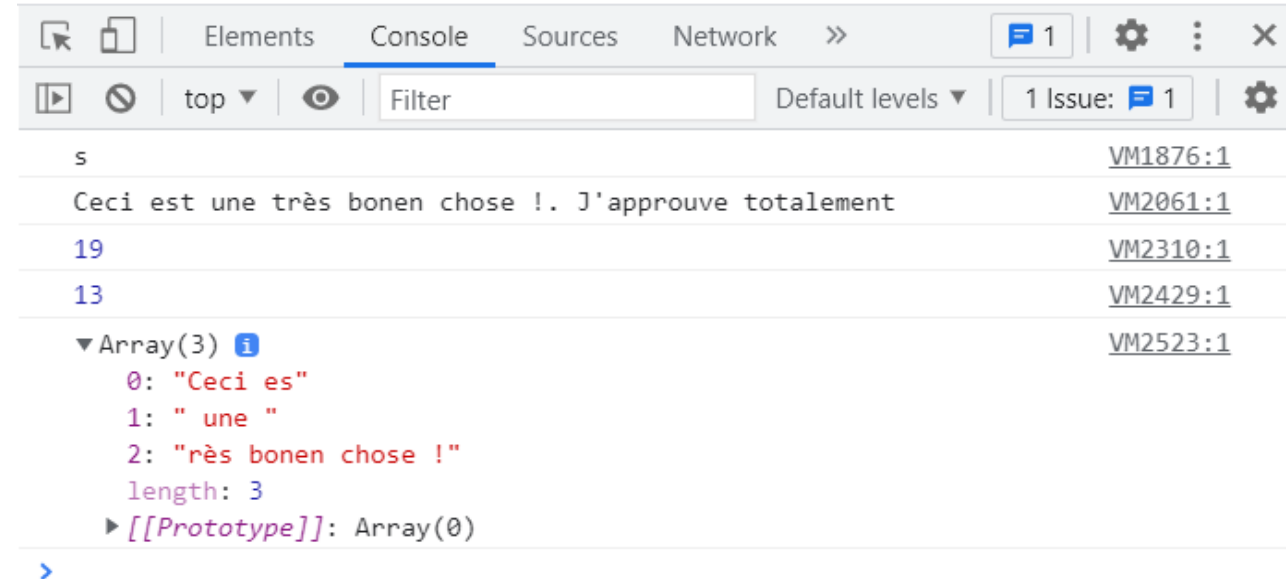
3. 4 Connaître les types de données de JavaScript

- Données de type chaîne
 - Quelques fonctions intégrées de chaîne :
 - *split()* → partage une chaîne en un tableau de sous-chaînes



The screenshot shows the Chrome DevTools Console with the 'Console' tab selected. The input field contains the text 'Ceci est une très bonen chose !. J'approuve totalement'. Below the input, the output of the `split()` function is displayed as an array of three strings: `['Ceci es', ' une ', 'rès bonen chose !']`. The console also shows the return value of `console.log()` as `undefined`.

```
s
Ceci est une très bonen chose !. J'approuve totalement
19
13
> console.log(exempleChaine1.split("t"));
  ▶ (3) ['Ceci es', ' une ', 'rès bonen chose !']
< undefined
> |
```



The screenshot shows the Chrome DevTools Console with the 'Console' tab selected. The input field contains the text 'Ceci est une très bonen chose !. J'approuve totalement'. Below the input, the output of the `split()` function is displayed as an array of three strings: `['Ceci es', ' une ', 'rès bonen chose !']`. The console also shows the return value of `console.log()` as `undefined`. The array structure is expanded, showing the elements and the `length` property.

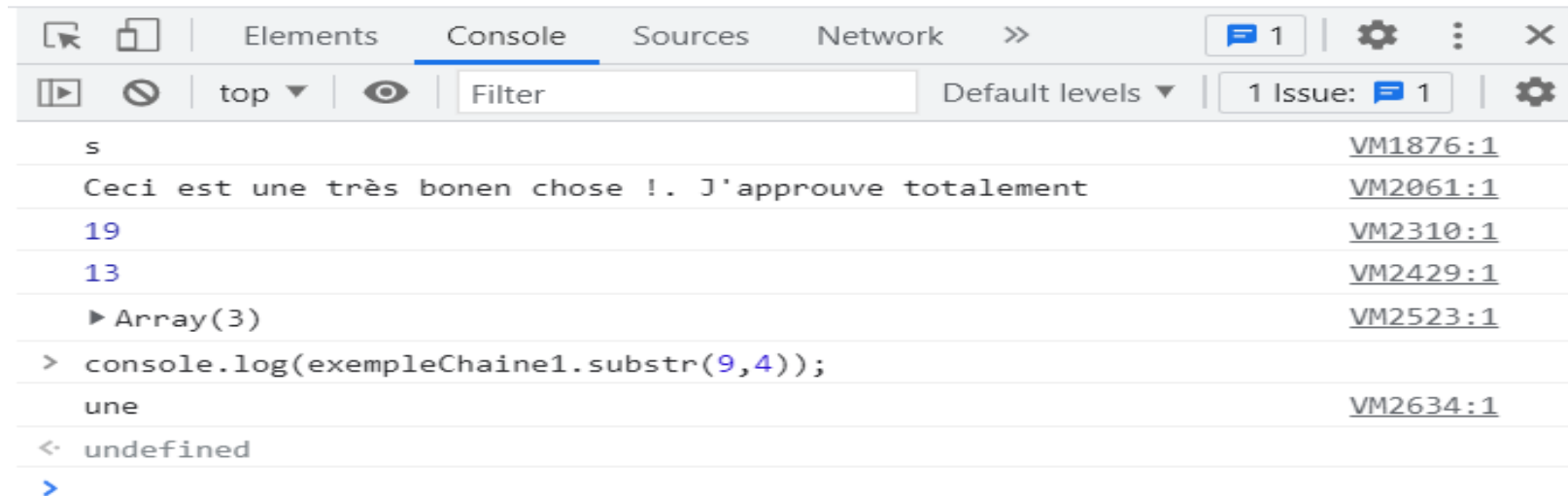
```
s
Ceci est une très bonen chose !. J'approuve totalement
19
13
▼ Array(3) ⓘ
  0: "Ceci es"
  1: " une "
  2: "rès bonen chose !"
  length: 3
  ▶ [[Prototype]]: Array(0)
>
```

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type chaîne
 - Quelques fonctions intégrées de chaîne :
 - *substr()* → extrait une partie de la chaîne commençant à la position indiquée et possédant la longueur spécifiée



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays the following sequence of operations and results:

- A variable `s` is assigned the value `Ceci est une très bonen chose !. J'approuve totalement`.
- The value of `s` is logged, showing the full string.
- The length of `s` is logged, showing `19`.
- The length of the substring is logged, showing `13`.
- An array of length 3 is created and logged.
- The `substr(9, 4)` method is called on `exempleChaine1`, and the result is logged.
- The result of the `substr` operation is logged, showing `une`.
- The console shows `undefined` and a prompt character `>`.

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type chaîne
 - Quelques fonctions intégrées de chaîne :
 - *substring()* → extrait les caractères de la chaîne compris entre 2 positions spécifiées



```
s
Ceci est une très bonen chose !. J'approuve totalement
19
13
Array(3)
une
est
> console.log(exempleChaine1.substring(5,10));
est u
< undefined
```


Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type chaîne

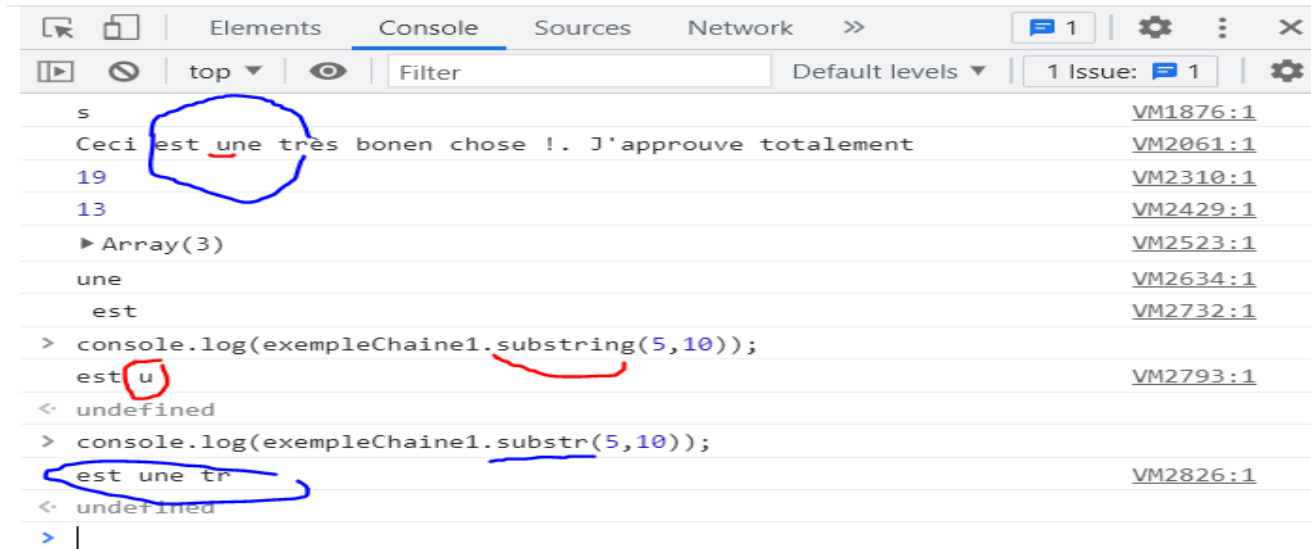
- Quelques fonctions intégrées de chaîne :

- *substring()* → extrait les caractères de la chaîne compris entre 2 positions spécifiées



```
s
Ceci est une très bonen chose !. J'approuve totalement
19
13
▶ Array(3)
une
est
> console.log(exempleChaine1.substring(5,10));
est u
< undefined
```

The screenshot shows the Chrome DevTools Console. The first log entry is the string "Ceci est une très bonen chose !. J'approuve totalement". The second log entry is the number 19. The third log entry is the number 13. The fourth log entry is an array of 3 elements. The fifth log entry is the string "une". The sixth log entry is the string "est". The seventh log entry is the result of the JavaScript code `console.log(exempleChaine1.substring(5,10));`, which is the string "est u". The eighth log entry is `< undefined`. A red arrow points to the string "est u".



```
s
Ceci est une très bonen chose !. J'approuve totalement
19
13
▶ Array(3)
une
est
> console.log(exempleChaine1.substring(5,10));
est u
< undefined
> console.log(exempleChaine1.substr(5,10));
est une tr
< undefined
```

The screenshot shows the Chrome DevTools Console. The first log entry is the string "Ceci est une très bonen chose !. J'approuve totalement". The second log entry is the number 19. The third log entry is the number 13. The fourth log entry is an array of 3 elements. The fifth log entry is the string "une". The sixth log entry is the string "est". The seventh log entry is the result of the JavaScript code `console.log(exempleChaine1.substring(5,10));`, which is the string "est u". The eighth log entry is `< undefined`. The ninth log entry is the result of the JavaScript code `console.log(exempleChaine1.substr(5,10));`, which is the string "est une tr". The tenth log entry is `< undefined`. A blue circle highlights the string "est u" and the string "est une tr".

Scripts clients - JavaScript

3. Les variables

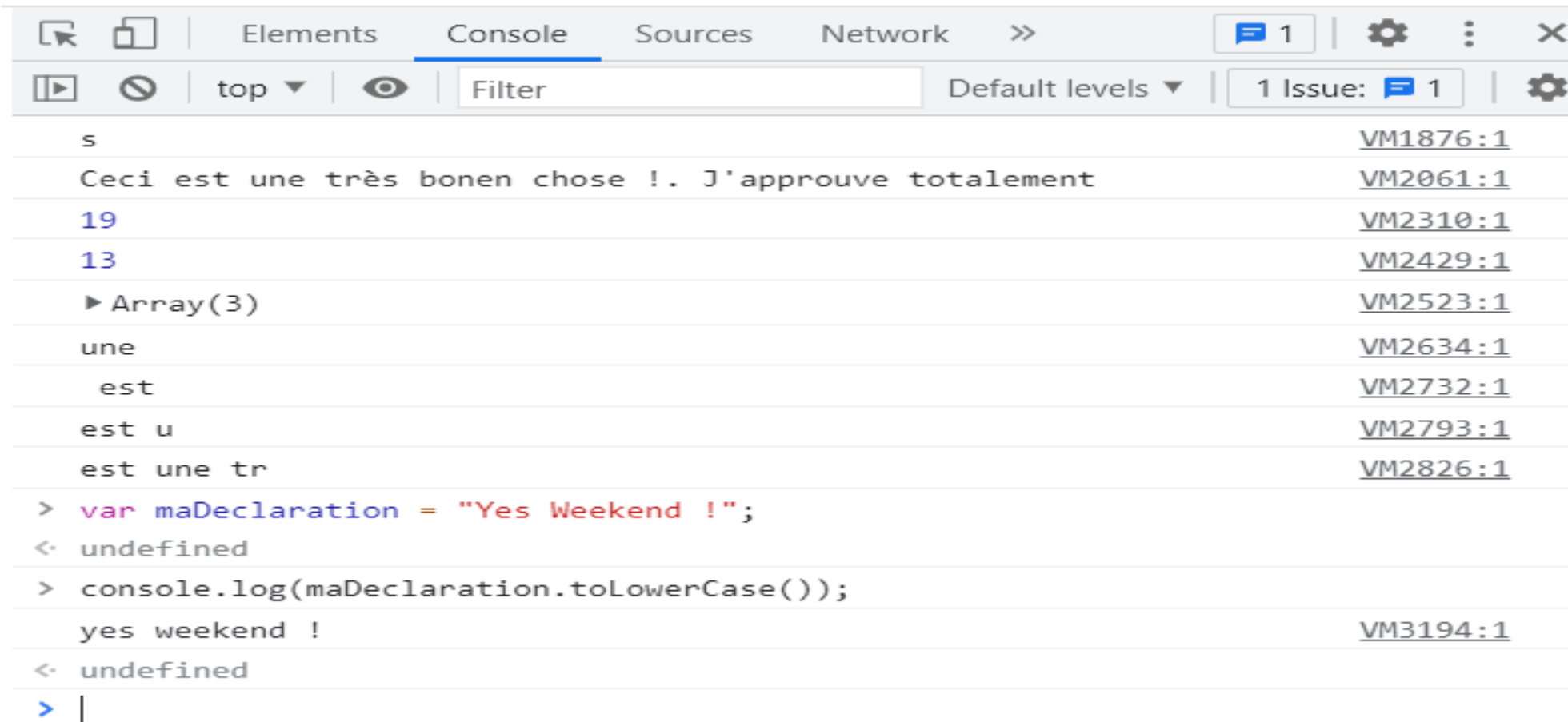
3. 4 Connaître les types de données de JavaScript

- Données de type chaîne
 - Quelques fonctions intégrées de chaîne :
 - *toLowerCase()* → convertit tous les caractères en minuscules

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays a series of log messages and code executions. The messages include a French sentence, a blue number '19', a blue number '13', a log of an array with 3 elements, and a log of the string 'yes weekend !'. The code shows a variable declaration and a log statement. The console interface includes a toolbar with icons for back, forward, and search, as well as a filter input field and a 'Default levels' dropdown. The right side of the console shows a '1 Issue' status and a settings gear icon.

```
s VM1876:1
Ceci est une très bonen chose !. J'approuve totalement VM2061:1
19 VM2310:1
13 VM2429:1
▶ Array(3) VM2523:1
une VM2634:1
est VM2732:1
est u VM2793:1
est une tr VM2826:1
> var maDeclaration = "Yes Weekend !";
< undefined
> console.log(maDeclaration.toLowerCase());
yes weekend ! VM3194:1
< undefined
> |
```

Scripts clients - JavaScript

3. Les variables

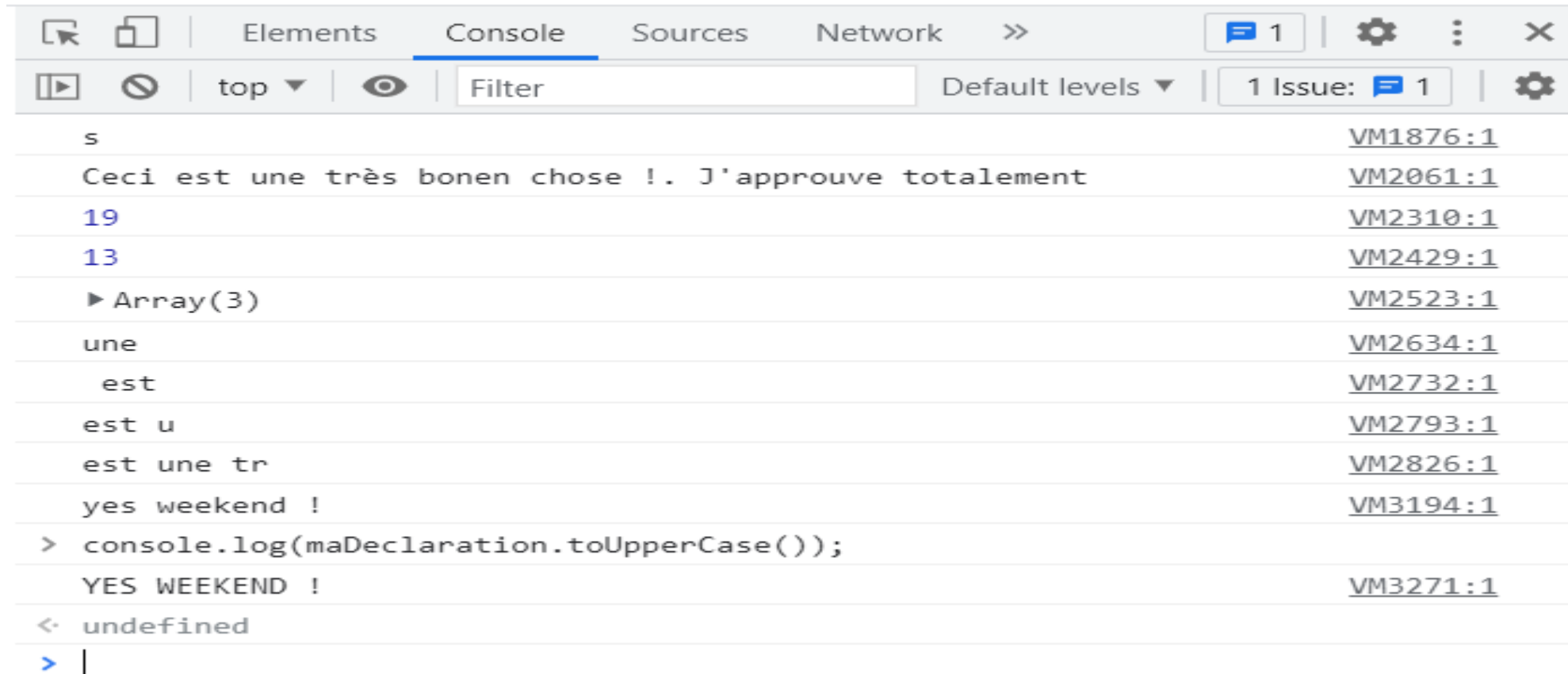
3. 4 Connaître les types de données de JavaScript

- Données de type chaîne
 - Quelques fonctions intégrées de chaîne :
 - *toUpperCase()* → convertit tous les caractères en majuscules

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays a series of log messages and a variable declaration. The messages are as follows:

Message	Source
s	VM1876:1
Ceci est une très bonen chose !. J'approuve totalement	VM2061:1
19	VM2310:1
13	VM2429:1
► Array(3)	VM2523:1
une	VM2634:1
est	VM2732:1
est u	VM2793:1
est une tr	VM2826:1
yes weekend !	VM3194:1
> console.log(maDeclaration.toUpperCase());	
YES WEEKEND !	VM3271:1
< undefined	
>	

Scripts clients - JavaScript

3. Les variables

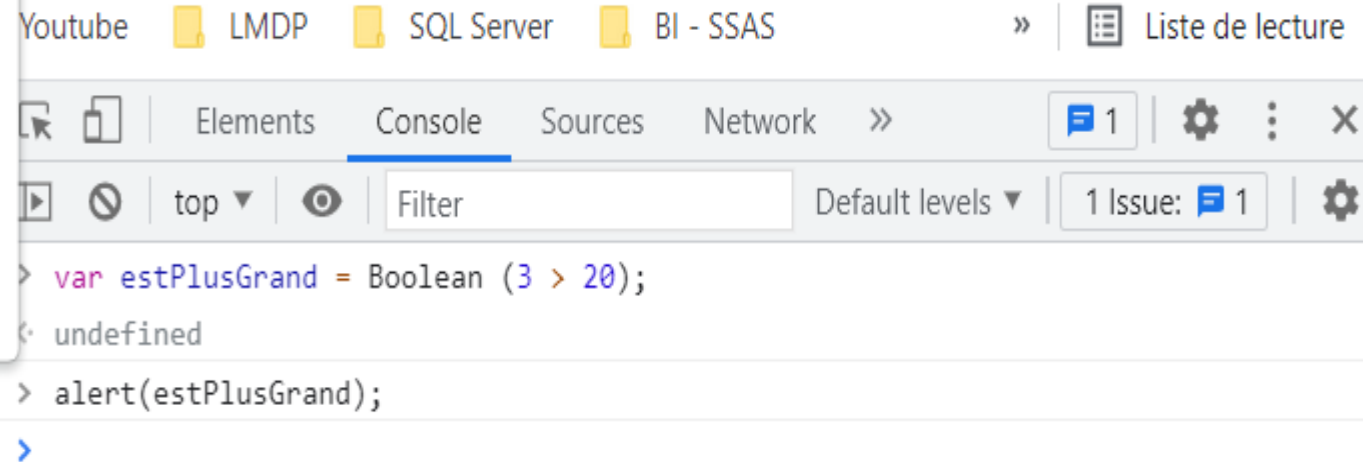
3. 4 Connaître les types de données de JavaScript

- Données de type booléen
 - ne peuvent contenir que 2 valeurs :
 - true
 - false
 - souvent employées pour enregistrer le résultat des comparaisons
 - que donnent les instructions suivantes :
var estPlusGrand = Boolean(3 > 20);
var estIdentique = Boolean('lfosup' == 'ifosup');

Scripts clients - JavaScript

...page intégrée à l'adresse new-tab-page-third-party indique
false

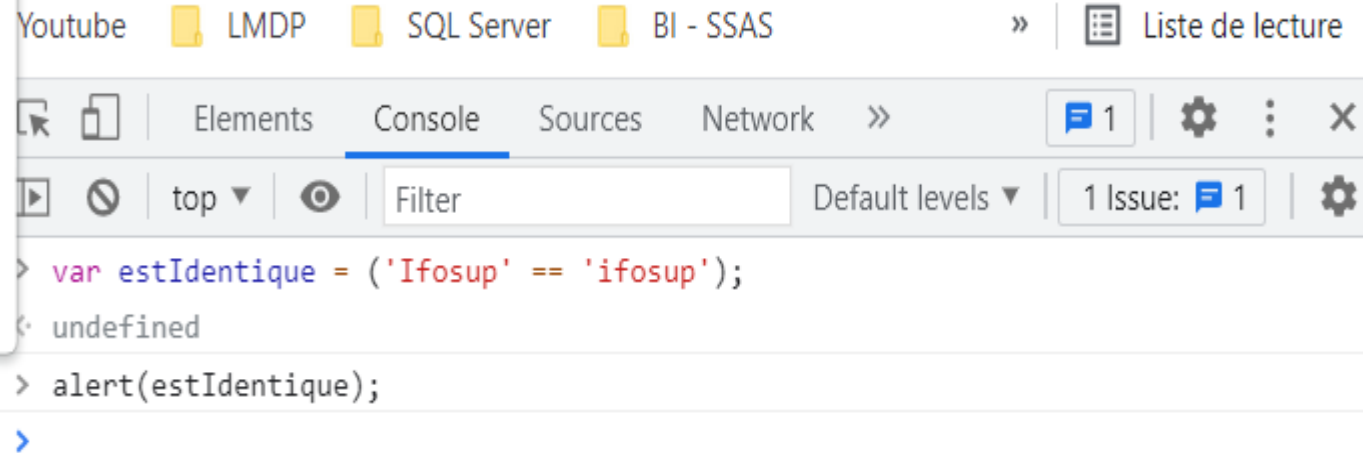
OK



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays a JavaScript error: 'var estPlusGrand = Boolean (3 > 20);' followed by 'undefined' and 'alert(estPlusGrand);'. The error message '1 Issue: 1' is visible in the top right corner of the console panel. The browser's address bar shows the URL 'new-tab-page-third-party'.

...page intégrée à l'adresse new-tab-page-third-party indique
false

OK



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays a JavaScript error: 'var estIdentique = ('Ifosup' == 'ifosup');' followed by 'undefined' and 'alert(estIdentique);'. The error message '1 Issue: 1' is visible in the top right corner of the console panel. The browser's address bar shows the URL 'new-tab-page-third-party'.

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type booléen
 - valeurs qui sont toujours évaluées comme **false** :
 - NaN
 - indéfini
 - 0
 - 0
 - "" ou " (chaîne vide)
 - false

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type booléen
 - toute autre valeur hors de celles-ci-dessus sont toujours évaluées comme **true** :

"NaN"

89

"89"

"0"

"toute chaine de caractères"

true

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type NaN
 - **Not A Number**
 - c'est le résultat obtenu si :
 - on fait des calculs avec une chaîne qui ne peut pas être convertie en valeur numérique
 - un calcul échoue
 - un calcul ne peut pas être effectué (exemple: racine carrée d'un nombre négatif)

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Données de type indéfini
 - valeur que JavaScript donne à une variable non initialisée (« **undefined** »)

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Exercices à exécuter dans le navigateur:

a) Créez les variables suivantes :

num qui contient le nombre 255

txt qui contient la chaîne de caractère 255

bin qui contient la valeur booléenne *vraie*

flt qui contient la valeur 7,23

→ Appliquez les opérations suivantes :

- incrémentez **num** de 1

- concaténez à la chaîne **txt** la chaîne suivante : « *est pris comme une chaîne de caractères* »

- ajoutez dans **flt** la valeur contenue dans **num**

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Exercices à exécuter dans le navigateur:

b) Créez les variables suivantes :

num qui contient le nombre 45

txt qui contient la chaîne de caractère 55

→ Appliquez les opérations suivantes :

- Affichez le résultat de **txt + num** avec la commande **console.log(txt + num);**
- Affichez le résultat de **num + txt**

Scripts clients - JavaScript

3. Les variables

3. 4 Connaître les types de données de JavaScript

- Exercices à exécuter dans le navigateur:

c) Créez les variables suivantes :

➔ Peut-on dire que cet exercice montre que :

- la variable **txt** a été traitée comme un nombre ?
- la variable **num** a été traitée comme une chaîne de caractères ?
- que JavaScript réalise des conversions automatiques ?
- qu'il n'est pas bien de faire des opérations avec des variables de types différents ?
- qu'il faut explicitement écrire les conversions dans les programmes pour éviter des mauvaises surprises ?
- que JavaScript est fortement typé ?
- que JavaScript est faiblement typé ?

Scripts clients - JavaScript

4. Les tableaux

4.1 Identifier et définir des tableaux

4.2 Construire des tableaux

4.3 Tableaux multidimensionnels

4.4 Travailler avec les éléments des tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

Scripts clients - JavaScript

4. Les tableaux

4.1 Identifier et définir des tableaux

➔ But de travailler avec des tableaux :

- Créer des listes
- Les ordonner
- Ajouter des éléments
- Supprimer des éléments

➔ Intérêt :

- enregistrer plusieurs données similaires sous un même nom !
- Ce qui n'est pas possible avec les variables ➔ nécessiterait de disposer d'une variable pour chaque donnée !
- Et faire une opération comme trier ces données dans l'ordre alphabétique demanderait beaucoup d'effort de programmation !

Scripts clients - JavaScript

4. Les tableaux

4.1 Identifier et définir des tableaux

- ➔ Un tableau = une variable contenant des valeurs multiples présentant une certaine cohérence
- ➔ Un ***élément de tableau*** est formé :
 - du nom du tableau
 - d'un numéro d'indice (ces derniers sont utilisés pour accéder aux éléments de tableau)
 - Exemple :
`tableauExemple[0] = "mazda" ;`
`tableauExemple[1] = "toyota";`
`tableauExemple[2] = "nissan";`

Scripts clients - JavaScript

4. Les tableaux

4.1 Identifier et définir des tableaux

- ➔ Nombre maximal des éléments de tableau : 4.294.967.295
- ➔ L'indice de base d'un élément de tableau = 0
- ➔ Les tableaux peuvent contenir n'importe quel type de donnée

- Exemple :

tableauExemple[0] = "mazda" ;

tableauExemple[1] = "fritte";

tableauExemple[2] = 898;

tableauExemple[3] = tableauExemple[0] + ' ' + tableauExemple[1] ;

Scripts clients - JavaScript

4. Les tableaux

4.2 Construire des tableaux

→ 2 manières :

- Avec le mot-clé *new* et *Array*
 - `var tableauExemple = new Array("épinards", "aubergine", "concombre");`
- Avec la notation littérale :
 - `var tableauExemple = ["épinards", "aubergine", "concombre "];`
 - Les crochets indiquent à JavaScript qu'il s'agit d'un tableau
 - Notation plus courte → utile lorsqu'on écrit beaucoup des lignes de code !

Scripts clients - JavaScript

4. Les tableaux

4.2 Construire des tableaux

➔ On peut créer un tableau sans l'initialiser, et lui affecter des valeurs par la suite

```
var tableauExemple = [];  
tableauExemple[0] = "épinards";  
tableauExemple[1] = "aubergine";  
tableauExemple[2] = "concombre";
```

➔ On peut rajouter une valeur dans un indice en dehors de l'ordre des indices :

```
tableauExemple[99] = "choux";
```

-> les valeurs des indices non renseignés sont considérées comme vides !

-> JavaScript reconnaît bien que le tableau a 100 positions :

```
tableauExemple.length; // retourne 100
```

Scripts clients - JavaScript

4. Les tableaux

4.3 Tableaux multidimensionnels

- ➔ Un tableau multidimensionnel = un tableau qui contient d'autres tableaux
- ➔ Exemple : un tableau des types de logements :

```
var typesLogements = [];  
typesLogements[0] = ["Appartements", "studio"];  
typesLogements[1] = ["Villas", "Villa avec piscine"];  
typesLogements[2] = ["Chateaux", "Chateau d'ax"];
```

Scripts clients - JavaScript

4. Les tableaux

4.4 Travailler avec les éléments des tableaux

➔ Pour accéder aux éléments d'un tableau ➔ écrire :

- nomDuTableau
- crochet
- indice

➔ Exemples :

```
alert(tableauExemple[0]);  
alert( tableauExemple[1]);  
alert( tableauExemple[2]);
```

```
alert(typesLogements[0][0]);  
alert(typesLogements[1][1]);  
alert( typesLogements[2][0]);
```

Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

➔ Pour accéder aux fonctions et propriétés d'un tableau : utiliser la **notation point**

- Taper le nom du tableau
- le faire suivre
 - d'un point
 - puis de la fonction ou propriété à laquelle on veut accéder

➔ Propriété très utilisée : **length**

```
var tableauExemple = [];  
tableauExemple[0] = "épinards";  
tableauExemple[1] = "aubergine";  
tableauExemple[2] = "concombre";
```

```
tableauExemple.length;
```

Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

➔ Pour accéder aux fonctions et propriétés d'un tableau : utiliser la **notation point**

- Taper le nom du tableau
- le faire suivre
 - d'un point
 - puis de la fonction ou propriété à laquelle on veut accéder

➔ Propriété très utilisée : **length**

```
var tableauExemple = [];  
tableauExemple[0] = "épinards";  
tableauExemple[1] = "aubergine";  
tableauExemple[2] = "concombre";
```

```
tableauExemple.length;
```


Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

→ Quelques méthodes des tableaux :

→ **concat()** : forme un nouveau tableau à partir du tableau courant concaténé avec un ou plusieurs autres tableaux, et/ ou valeurs

```
var texte = ["a", "b", "c"];  
var chiffre = [1, 2, 3];
```

```
texte.concat(chiffre); // → : ["a", "b", "c", 1, 2, 3]
```

→ **every()** : renvoie **true** si chaque élément du tableau satisfait aux conditions de la fonction spécifiée

```
function estAssezGrand(element, index, array) {  
    return element >= 10; }  
[12, 5, 8, 130, 44].every(estAssezGrand); // false  
[12, 54, 18, 130, 44].every(estAssezGrand); // true
```

Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

→ Quelques méthodes des tableaux :

→ ***filter()*** : forme un nouveau tableau contenant tous les éléments du tableau courant qui satisfont la propriété

```
function suffisammentGrand(element) {  
    return element >= 10; }  
var filtre = [12, 5, 8, 130, 44].filter(suffisammentGrand);  
  
Console.log(filtre); // → filtre vaut [12, 130, 44]
```

Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

➔ Quelques méthodes des tableaux :

➔ *forEach()* : exécute la fonction spécifiée pour chaque élément du tableau

```
// avec forEach()
```

➔ Copier et exécuter le code « `forEachTest.html` »

```
<!DOCTYPE html>
<html>
  <body>
    <h2>JavaScript Arrays</h2>

    <p>
      The Array.forEach() method calls a function for each element in an array.
    </p>

    <p id="demo"></p>

    <script>
      let text = "";
      const fruits = ["apple", "orange", "cherry"];

      function myFunction(item, index) {
        text += index + ": " + item + "<br>";
      }

      fruits.forEach(myFunction);

      document.getElementById("demo").innerHTML = text;
    </script>
  </body>
</html>
```

Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

➔ Quelques méthodes des tableaux :

➔ ***forEach()*** : exécute la fonction spécifiée pour chaque élément du tableau

```
// boucle for classique
// =====
var items = ["item1", "item2", "item3"];
var copie = [];

for (var i = 0; i < items.length; i++) {
    copie.push(items[i]);
}
console.log(copie); // vaut item1, item2, item3
```

Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

➔ Quelques méthodes des tableaux :

➔ ***forEach()*** : exécute la fonction spécifiée pour chaque élément du tableau

```
var items = ["item1", "item2", "item3"]  
var copie = [];  
  
items.forEach(function(item){  
    copie.push(item);  
});
```

Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

→ Quelques méthodes des tableaux :

→ ***indexOf()*** : renvoie la première occurrence de la valeur spécifiée à l'intérieur du tableau.
Renvoie **-1** si la valeur n'est pas trouvée

```
var tableau = [2, 9, 9];  
  
tableau.indexOf(2); // 0  
tableau.indexOf(7); // -1  
tableau.indexOf(9, 2); // 2  
tableau.indexOf(2, -1); // -1  
tableau.indexOf(2, -3); // 0
```

- peut prendre :

- 1 argument : une valeur du tableau
- 2 arguments : la valeur et l'indice du début de recherche

Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

→ Quelques méthodes des tableaux :

→ ***lastIndexOf()*** : renvoie la dernière occurrence de la valeur spécifiée à l'intérieur du tableau.
Renvoie **-1** si la valeur n'est pas trouvée

```
'canal'.lastIndexOf('a'); // renvoie 3  
'canal'.lastIndexOf('a', 2); // renvoie 1  
'canal'.lastIndexOf('a', 0); // renvoie -1  
'canal'.lastIndexOf('x'); // renvoie -1  
'canal'.lastIndexOf('c', -5); // renvoie 0  
'canal'.lastIndexOf('c', 0); // renvoie 0  
'canal'.lastIndexOf(''); // renvoie 5  
'canal'.lastIndexOf('', 2); // renvoie 2
```


Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

→ Quelques méthodes des tableaux :

→ **pop()** : supprime le dernier élément du tableau et retourne cette valeur

```
var mesPoissons = ["angel", "clown", "mandarin", "sturgeon"];  
var popped = mesPoissons.pop();  
  
console.table(mesPoissons); // angel, clown, madarin  
console.log(popped); // sturgeon
```

→ **push()** : ajoute de nouveaux éléments à la fin d'un tableau et retourne la nouvelle longueur du tableau

```
var sports = ["plongée", "baseball"];  
var total = sports.push("football", "tennis");  
console.log(sports); // ["plongée", "baseball", "football", "tennis"]  
console.log(total); // 4
```

Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

➔ Quelques méthodes des tableaux :

➔ ***reverse()*** : inverse l'ordre des éléments d'un tableau

```
var monArray = ["un", "deux", "trois"];  
monArray.reverse();  
  
console.log(monArray) // ["trois", "deux", "un"]
```

➔ ***shift()*** : supprime le premier élément d'un tableau et retourne cet élément, en modifiant la longueur du tableau

```
var mesPoissons = ["ange", "clown", "mandarin", "chirurgien"];  
console.log("mesPoissons avant : " + JSON.stringify(mesPoissons));  
// mesPoissons avant : ["ange","clown","mandarin","chirurgien"]  
var premierÉlément = mesPoissons.shift();  
console.log("mesPoissons après :", mesPoissons); // mesPoissons après : ["clown", "mandarin",  
"chirurgien"]  
console.log("Cet élément a été enlevé :", premierÉlément); // "Cet élément a été enlevé : ange"
```

Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

→ Quelques méthodes des tableaux :

→ ***unshift()*** : rajoute un ou plusieurs éléments au début d'un tableau et retourne la nouvelle longueur du tableau

```
let arr = [4, 5, 6]; arr.unshift(1, 2, 3);  
console.table(arr); // [1, 2, 3, 4, 5, 6]
```

```
let arr2 = [4, 5, 6];  
arr2.unshift(1);  
arr2.unshift(2);  
arr2.unshift(3);  
console.table(arr2); // [3, 2, 1, 4, 5, 6]
```

Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

➔ Quelques méthodes des tableaux :

➔ *slice()* : sélectionne une partie du tableau et retourne le résultat dans un nouveau tableau

```
var fruits = ["Banane", "Orange", "Citron", "Pomme", "Mangue"];  
var agrumes = fruits.slice(1, 3);
```

```
console.log(fruits); // fruits vaut --> ["Banane", "Orange", "Citron", "Pomme", "Mangue"]  
console.log(agrumes); // agrumes vaut --> ["Orange", "Citron"]
```

➔ Le tableau de départ, « fruits », n'a pas changé !

➔ *slice()* extrait :

- à partir du 1^{er} indice

- jusqu'au dernier indice, la valeur de cet indice étant exclue de l'extraction

➔ Les indices sont facultatifs

Scripts clients - JavaScript

4. Les tableaux

4.5 Utiliser les fonctions et les propriétés des tableaux

→ Quelques méthodes des tableaux :

→ ***toString()*** : convertit un tableau en chaîne de caractères et retourne la chaîne obtenue

Scripts clients - JavaScript

4. Les tableaux

4.5 Exercices :

a) Faire un script qui :

- va demander à l'utilisateur de rentrer des noms de langage de programmation
- va stocker ces noms dans un tableau
- la fin de la saisie va être indiquée par la valeur ""

Afficher ce tableau :

- dans la console
- dans le navigateur

Scripts clients - JavaScript

```
alert("Exerice a");  
while (true) {  
    var nomLangage = prompt("Entrez le nom d\'un lan  
gage, svp : ");  
    if (nomLangage != ' ') {  
        langagesProg.push(nomLangage);  
    }  
    else {  
        break;  
    }  
}
```

Scripts clients - JavaScript

```
var compteur = 0;
while (compteur <= langagesProg.length) {
    document.write(langagesProg[compteur]);
    document.write("<br \>");
    compteur = compteur + 1;
}
console.log(langagesProg);
```


Scripts clients - JavaScript

4. Les tableaux

4.5 Exercices :

b) A partir du tableau construit dans l'exercice (a) (→ recopier le script), écrire un script qui :

- lorsque l'utilisateur saisit un nom de langage de programmation, va indiquer si oui/non le langage recherché se trouve bien dans le tableau.

Scripts clients - JavaScript

```
alert("Exerice b");  
var nomLangage2 = prompt("Entrez le nom d\'un langage,  
svp : ");  
if (langagesProg.indexOf(nomLangage2) != 1) {  
    alert("Le langage existe bien !");  
}  
else {  
    alert("Le langage n\'existe pas !");  
}
```

Scripts clients - JavaScript

4. Les tableaux

4.5 Exercices :

c) Afficher dans un nouveau tableau toutes les occurrences d'un élément du tableau de départ

Scripts clients - JavaScript

```
alert("Exerice c");
var indices = [];
var tableau = ['a', 'b', 'a', 'c', 'a', 'd'];
var élément = 'a';
var idx = tableau.indexOf(élément);
console.log(idx);
while (idx != -1) {
    indices.push(idx);
    idx = tableau.indexOf(élément, idx + 1);
    console.log(idx);
}
console.log(indices); // [0, 2, 4]
```

Scripts clients - JavaScript

4. Les tableaux

4.5 Exercices :

d) Trouver si un élément existe et l'ajouter dans le tableau si ce n'est pas le cas

Scripts clients - JavaScript

```
alert("Exerice d");  
var indices = [];  
var tableau = ['a', 'b', 'a', 'c', 'a', 'd'];  
var élément = 'a';  
var idx = tableau.indexOf(élément);  
console.log(idx);  
while (idx != -1) {  
    indices.push(idx);  
    idx = tableau.indexOf(élément, idx + 1);  
    console.log(idx);  
}
```

Scripts clients - JavaScript

```
console.log(indices); // [0, 2, 4]
function mettreAJourLegumes(tabLegumes, légume) {
    if (tabLegumes.indexOf(légume) === -1) {
        tabLegumes.push(légume);
        console.log('Le nouveau tableau est : ' + tabLegumes);
    } else if (tabLegumes.indexOf(légume) > -1) {
        console.log(légume + ' existe déjà dans le tableau. ');
    }
}
```

Scripts clients - JavaScript

```
var tabLégumes = ['pomme de terre', 'tomate', 'poivron'];
```

```
mettreAJourLegumes(tabLégumes, 'épinard');
```

```
// Le nouveau tableau est : pomme de terre,tomate,poivron,épinard
```

```
mettreAJourLegumes(tabLégumes, 'épinard');
```

```
// épinard existe déjà dans le tableau.
```