

P00 en EcmaScript 6

- Les classes
- Héritage
- Méthodes getter, setter et static
- Fonctions fléchées (arrow functions)
- Structures Map, Set et boucle for of
- Portés des variables
- Déstructuration
- Promesses (promise)

DÉCLARATION D'UNE CLASSE

//Définition d'une classe

```
class nom_classe {  
    //Méthode constructeur  
    constructor(param1, param2 ...) {  
        this. attribut_1 = param1;  
        this. attribut_2 = param2;  
    }  
  
    //Méthode(s)  
    nomMéthode() {  
        //Traitements  
    }  
}
```

EXEMPLE

```
<!-- Script du JavaScript -->
<script>

    //Définition d'une classe
    class Voiture
    {
        //Méthode constructeur
        constructor (marque, modele, pays)
        {
            this.marque=marque;
            this.modele=modele;
            this.pays=pays;
        }

        //Méthode afficheVoiture
        afficheVoiture()
        {
            return "Marque : " + this.marque + " <br /> Modele : " + this.modele + " <br /> Pays : " + this.pays;
        }
    }

    //Instanciación d'un objet Peugeot de la classe Voiture
    let peugeot206 = new Voiture ("Peugeot", "206", "France");

    //Appel de la méthode afficheVoiture de la classe Voiture
    document.write (peugeot206.afficheVoiture());

</script>
```

EcmaScript 6 : POO (déclaration de classe)

Marque : Peugeot
Modele : 206
Pays : France

HÉRITAGE

```
//Définition d'une classe qui hérite d'une autre
classe

class nom_classe extends classe_parente
{
    //Méthode constructeur
    constructor(param1, param2, param3, param4...)
    {
        //Héritage des attributs de la classe parente
        super (param1, param2);

        //Attributs spécifiques de la classe
        this. attribut_3= param3;
        this. attribut_4= param4;
    }
}
```

```
//Autre(s) méthode(s)
nomMéthode()
{
    //Traitements

}
}
```

```

//Définition d'une classe VoitureSport (extension de la classe Voiture)
class VoitureSport extends Voiture
{
    //Méthode constructeur
    constructor (marque, modele, pays, vitessemax)
    {
        //Héritage des attributs de la classe parente Voiture
        super(marque, modele,pays);

        //Attribu spécifique de la classe VoitureSport
        this.vitessemax= vitessemax;
    }

    //Méthode afficheVoitureSport
    afficheVoitureSport()
    {
        return super.afficheVoiture() + " <br /> Vitesse Maximale " + this.vitessemax;
    }
}

//Instanciation d'un objet Peugeot de la classe Voiture
let peugeot206 = new Voiture ("Peugeot", "206", "France");

//Appel de la méthode afficheVoiture de la classe Voiture
document.write (peugeot206.afficheVoiture());

//Instanciation d'un objet Peugeot de la classe VoitureSport
let porsche911 = new VoitureSport ("Porsche", "911", "France", 290);

//Appel de la méthode afficheVoitureSport de la classe VoitureSport
document.write ( "<br /> <br />" +porsche911.afficheVoitureSport());

```

EcmaScript 6 : POO (déclaration de classe)

Marque : Peugeot

Modele : 206

Pays : France

Marque : Porsche

Modele : 911

Pays : France

Vitesse Maximale 290

```

//Définition d'une classe VoitureSport (extension de la classe Voiture)
class VoitureSport extends Voiture
{
    //Méthode constructeur
    constructor (marque, modele, pays, vitessemax)
    {
        //Héritage des attributs de la classe parente Voiture
        super(marque, modele,pays);

        //Attribu spécifique de la classe VoitureSport
        this.vitessemax= vitessemax;
    }

    //Méthode afficheVoitureSport
    afficheVoitureSport()
    {
        return super.afficheVoiture() + " <br /> Vitesse Maximale " + this.vitessemax;
    }
}

//Instanciation d'un objet Peugeot de la classe Voiture
let peugeot206 = new Voiture ("Peugeot", "206", "France");

//Appel de la méthode afficheVoiture de la classe Voiture
document.write (peugeot206.afficheVoiture());

//Instanciation d'un objet Peugeot de la classe VoitureSport
let porsche911 = new VoitureSport ("Porsche", "911", "France", 290);

//Appel de la méthode afficheVoitureSport de la classe VoitureSport
document.write ( "<br /> <br />" +porsche911.afficheVoitureSport());

```

EcmaScript 6 : POO (déclaration de classe)

Marque : Peugeot
 Modele : 206
 Pays : France

Marque : Porsche
 Modele : 911
 Pays : France
 Vitesse Maximale 290

MÉTHODES GETTER, SETTER, STATIC

- Une méthode getter permet d'accéder à la valeur d'un attribut d'un objet instancié par cette classe.
- Une méthode setter permet de modifier la valeur d'un attribut de l'objet et ceci l'instanciation initiale éventuellement.
- Une méthode static est une méthode particulière qui porte sur la classe et non pas sur les instances d'objets de la classe. La déclaration débute avec le mot-clé static.

```
<!-- Script du JavaScript -->
```

```
<script>
```

```
//Définition d'une classe
```

```
class Voiture
```

```
{
```

```
    //Méthode constructeur
```

```
    constructor(marque, modele, pays)
```

```
    {
```

```
        this.marque = marque;
```

```
        this.modele = modele;
```

```
        this.pays = pays;
```

```
    }
```

```
    // Méthode de type getter (accès à la valeur de l'attribut marque)
```

```
    get getMarque()
```

```
    {
```

```
        //Retour de la marque
```

```
        return this.marque;
```

```
    }
```

```
    // Méthode de type getter (accès à la valeur de l'attribut modele)
```

```
    get getModele()
```

```
    {
```

```
        //Retour du modele
```

```
        return this.modele;
```

```
    }
```

```
    // Méthode de type getter (accès à la valeur de l'attribut pays)
```

```
    get getPays()
```

```
    {
```

```
        //Retour du pays
```

```
        return this.pays;
```

```
    }
```



```
// Méthode de type setter (modification de l'attribut marque)
set setMarque(value)
{
    this.marque=value;
}

// Méthode de type setter (modification de l'attribut modele)
set setModele(value)
{
    this.modele=value;
}

// Méthode de type setter (modification de l'attribut pays)
set setPays(value)
{
    this.pays=value;
}

// Méthode statique comptant le nombre d'objets créés par l'intermédiaire de cette classe
static getNombreVoitures()
{
    if (!this.compteurVoiture && this.compteurVoiture !== 0)
    {
        //Initialisation du compteur d'objets instanciés s'il n'existe pas déjà
        this.compteurVoiture = 0;
    }
    else {
        //Incrémementation du nombre d'objets instanciés
        this.compteurVoiture++;
    }

    //Valeur de retour
    return this.compteurVoiture + 1;
}
}
```

```
//Instanciation d'un objet porsche911 de la classe Voiture
let porsche911 = new Voiture ("Porsche", "911", "France");
document.write("Marque : " + porsche911.getMarque + "<br />");
document.write("Modèle : " + porsche911.getModele + "<br />");
document.write("Pays : " + porsche911.getPays + "<br />");

document.write( "<br />");
document.write("Nombre des voitures instanciées : " + Voiture.getNombreVoitures() + "<br />");
document.write( "<br /> <br />");

//Modification de l'objet porsche911
porsche911.setModele = "911 Turbo";
document.write("Modèle (après modification): " + porsche911.getModele + "<br />");

//Instanciation d'un objet testarossa de la classe Voiture
let testarossa = new Voiture ("Ferrarri", "testarossa", "Italie");
document.write("Marque : " + testarossa.getMarque + "<br />");
document.write("Modèle : " + testarossa.getModele + "<br />");
document.write("Pays : " + testarossa.getPays + "<br />");

document.write( "<br />");
document.write("Nombre des voitures instanciées : " + Voiture.getNombreVoitures() + "<br />");

</script>
```

EcmaScript 6 : Méthodes getter, setter, static

Marque : Porsche
Modèle : 911
Pays : Allemagne

Nombre des voitures instanciées : 1

Modèle (après modification): 911 Turbo
Marque : Ferrarri
Modèle : testarossa
Pays : Italie

Nombre des voitures instanciées : 2

FONCTION FLÉCHÉES (ARROW FUNCTION)

Les fonction fléchées ont deux particularités:

- La syntaxe est allégée par rapport aux fonction EcmaScript 5.

Argument => valeur

- La non création d'un nouveau scope associé.

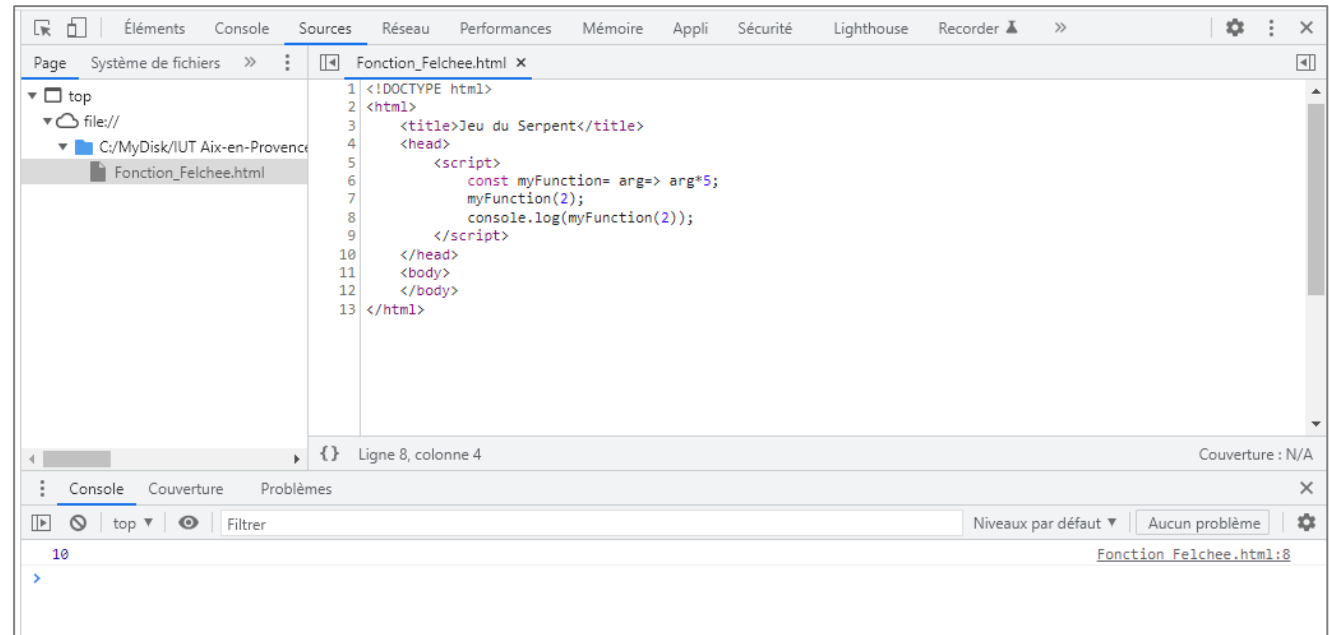
EXEMPLE

`arg => arg*5` C'est une fonction anonyme! **C'est pas pratique!**

Comment on va faire?

On va stocker dans une variable

```
const myFunction = arg=> arg*5;  
myFunction(2);  
Console.log(myFunction(2));
```



EXEMPLE — F^{CT} FLÉCHÉE COMME UNE MÉTHODE SUR UN OBJET

```
<!-- Début script JavaScript -->
<script>
  var personne={
    nom : "Jouini",
    prenom : "Rim",
    connaissance : function(friend)
    {
      return "Ravi de votre connaissance" + friend + "!";
    }
  };
  console.log(personne.connaissance(" Eric "));
  document.write("Rencontre <br />");
  document.write(personne.connaissance(" Eric "));
</script>
```

EXEMPLE — F^{CT} FLÉCHÉE COMME UNE MÉTHODE SUR UN OBJET

The screenshot displays a web browser window with the following components:

- Page Content:** A single line of text: "Rencontre Ravi de votre connaissance Eric !".
- Developer Tools:**
 - Sources Panel:** Shows the file "Fonction_Felchee_comme_des_objets.html". The code is as follows:

```
8 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10 <!-- Titre du script HTML -->
11 <title> Exemple - Fonction fléchée comme une méthode sur un objet </title>
12 </head>
13
14 <!-- Début section body du script HTML -->
15 <body>
16 <!-- Début script JavaScript -->
17 <script>
18   var personne={
19     nom : "Jouini",
20     prenom : "Rim",
21     connaissance : function(friend)
22     {
23       return "Ravi de votre connaissance" + friend + "!";
24     }
25   };
26
```
 - Console Panel:** Shows the output "Ravi de votre connaissance Eric !" and the execution location "Fonction Felchee com... des objets.html:27".

```
<!-- Début script JavaScript -->
<script>
  var personne={
    nom : "Jouini",
    prenom : "Rim",
    connaissance : function( friend )
    {
      return "Ravi de votre connaissance" + friend + "!";
    }
  };
  console.log(personne.connaissance(" Eric "));
  document.write("Rencontre <br />");
  document.write(personne.connaissance(" Eric "));
</script>
```

```
var personne={
  nom : "Jouini",
  prenom : "Rim",
  connaissance : friend => "Ravi de votre connaissance" + friend + "!"
};

console.log(personne.connaissance(" Eric "));
```

argument valeur

EXEMPLE — F^{CT} FLÉCHÉE COMME UNE MÉTHODE SUR UN OBJET

The screenshot displays a web browser window and its developer tools. The browser shows the text "Rencontre" and "Ravi de votre connaissance Eric !". The developer tools are open to the "Sources" panel, showing a file named "Fonction_Felchee_comme_un_methode.html". The code in the file defines a JavaScript object with a function arrow as a method. The console shows the output of the code execution.

```
16 <!-- Début script JavaScript -->
17 <script>
18   var personne={
19     nom : "Jouini",
20     prenom : "Rim",
21     /*connaissance : function(friend)
22     {
23       return "Ravi de votre connaissance" + friend + "!";
24     }*/
25     connaissance : friend => "Ravi de votre connaissance" + friend + "!"
26   };
27
28   console.log(personne.connaissance(" Eric "));
29   document.write("Rencontre <br />");
30   document.write(personne.connaissance(" Eric "));
31 </script>
32 </body>
33
34 </html>
```

Console output: Ravi de votre connaissance Eric !

Pour les fonctions qui ne prennent pas des arguments Comment on va les écrire en fonctions fléchées?

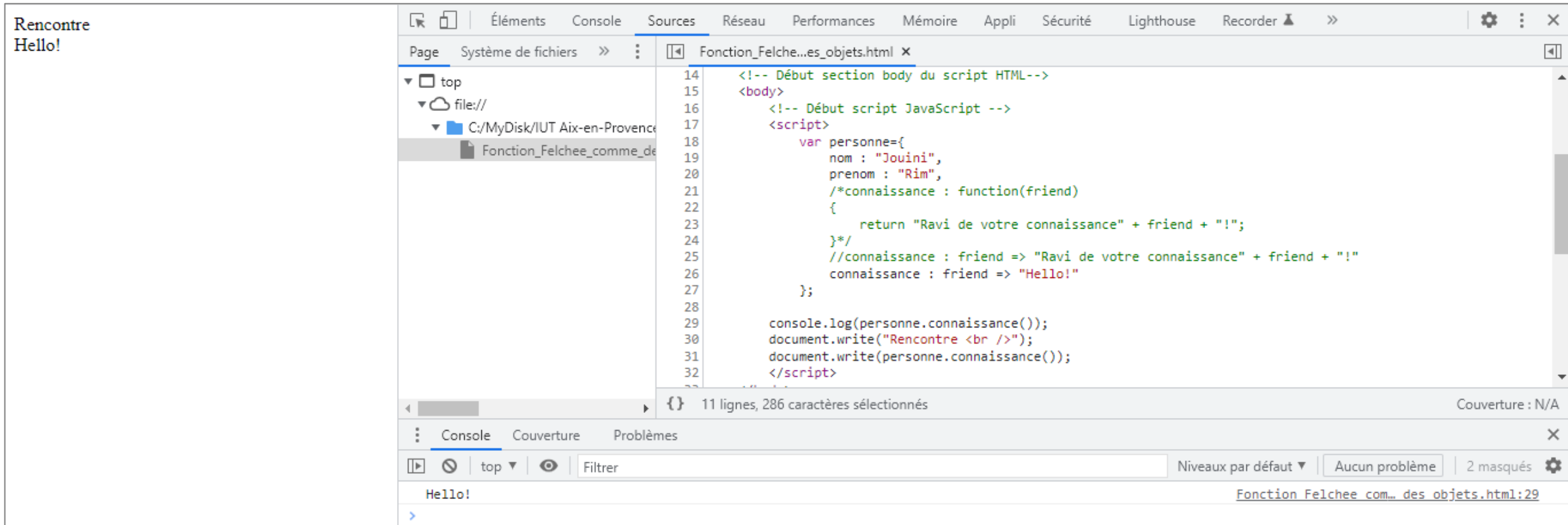

```
<!-- Début script JavaScript -->
<script>
  var personne={
    nom : "Jouini",
    prenom : "Rim",
    connaissance : function()
    {
      return "Hello!";
    }
  };
  console.log(personne.connaissance());
  document.write("Rencontre <br />");
  document.write(personne.connaissance(" Eric "));
</script>
```

```
var personne={
  nom : "Jouini",
  prenom : "Rim",
  connaissance : () => "Hello!"
};

console.log(personne.connaissance());
```

Sans argument valeur

EXEMPLE — F^{CT} FLÉCHÉE SANS ARGUMENT SUR UN OBJET



Pour une fonction qui reçoit plusieurs arguments

Comment on va l'écrire en fonctions fléchées?

```
<!-- Début script JavaScript -->
```

```
<script>
```

```
var personne={
```

```
  nom : "Jouini",
```

```
  prenom : "Rim",
```

```
  connaissance : function( friend_1, friend_2, friend_3 )
```

```
{
```

```
  return "Ravi de votre connaissance" + friend1 + ", " + friend2 + ", " + friend3 + "!"
```

```
}
```

```
};
```

```
console.log(personne.connaissance(" Marie-Francoise "," Eric "," Jean-Micheal "));
```

```
document.write("Rencontre <br />");
```

```
document.write(personne.connaissance(" Eric "));
```

```
</script>
```

```
var personne={
```

```
  nom : "Jouini",
```

```
  prenom : "Rim",
```

```
  connaissance:(friend1, friend2, freind3 ) => "Ravi de votre connaissance" + friend1 + ", " + friend2 + ", " + friend3 + "!"
```

```
};
```

arguments

valeur

```
console.log(personne.connaissance(" Marie-Francoise "," Eric "," Jean-Micheal "));
```

EXEMPLE — F^{CT} FLÉCHÉE AVEC PLUSIEURS ARGUMENTS SUR UN OBJET

The screenshot displays a web browser window and its developer tools. The browser shows a page titled "Rencontre" with the text "Ravi de votre connaissance Marie-Francoise , Eric et Jean-Micheal !". The developer tools are open to the "Sources" tab, showing the file "Fonction_Felchee...es_objets.html". The code in the editor defines a JavaScript object with a function arrow method.

```
8      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10     <!-- Titre du script HTML -->
11     <title> Exemple - Fonction fléchée comme une méthode sur un objet </title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16     <!-- Début script JavaScript -->
17     <script>
18         var personne={
19             nom : "Jouini",
20             prenom : "Rim",
21             /*connaissance : function(friend)
22             {
23                 return "Ravi de votre connaissance" + friend + "!";
24             }*/
25             //connaissance : friend => "Ravi de votre connaissance" + friend + "!"
26             connaissance : (friend1, friend2, friend3) => "Ravi de votre connaissance " + friend1 + " , " +
27         };
28
29     console.log(personne.connaissance(" Marie-Francoise", "Eric", "Jean-Micheal"));
30     document.write("Rencontre <br />");
31     document.write(personne.connaissance(" Marie-Francoise", "Eric", "Jean-Micheal"));
32 </script>
33 </body>
34
```

The console shows the output of the function call: "Ravi de votre connaissance Marie-Francoise , Eric et Jean-Micheal !". The status bar at the bottom indicates "Ligne 31, colonne 40" and "Couverture : N/A".

EXEMPLE - F^{CT} FLÉCHÉE AVEC BLOC

```
<!-- Début script JavaScript -->
<script>
  var personne={
    nom : "Jouini",
    prenom : "Rim",
    connaissance : friend => {
      const presentation ="Ravi de votre connaissance " + friend + " !";
      console.log(presentation);
      return presentation;
    }
  };
  personne.connaissance("Marie-Francoise"));
</script>
```

EXEMPLE - F^{CT} FLÉCHÉE AVEC BLOC

The screenshot shows a web browser's developer tools interface. The 'Sources' tab is active, displaying a file named 'Fonction_Felchee_comme_de..._fonction.html'. The code is as follows:

```
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     var personne={
19       nom : "Jouini",
20       prenom : "Rim",
21       connaissance : friend => {
22         const presentation ="Ravi de votre connaissance " + friend + " !";
23         console.log(presentation);
24         return presentation;
25       }
26     };
27     personne.connaissance("Marie-Francoise");
28   </script>
29 </body>
30
31 </html>
32
```

The status bar at the bottom indicates 'Ligne 27, colonne 13' and 'Couverture : N/A'. The 'Console' tab is also visible, showing a log message: 'Ravi de votre connaissance Marie-Francoise !' with the source 'Fonction_Felchee_comme_de..._ck fonction.html:23'.

EXEMPLE - F^{CT} FLÉCHÉE & .THIS

The screenshot displays a web browser's developer tools interface. The top pane shows the source code of a file named 'Fonction_Felchee_This.html'. The code defines three functions: a classic function, an arrow function, and a classic function bound to 'this'. A 'personne' object is created with properties for name, first name, and the three functions. The object's methods are called, and the results are logged to the console.

```
15 <body>
16 <!-- Début script JavaScript -->
17 <script>
18   const classicfunction =function()
19   {
20     console.log(this);
21   }
22
23   //une fonction fléchée qui est l'équivalente de la fonction classique
24   const arrowfunction = () => console.log(this);
25
26   const classicFunctionBind = classicfunction.bind(this);
27   var personne={
28     nom : "Jouini",
29     prenom : "Rim",
30     presentclassique : classicfunction,
31     presentarrow : arrowfunction,
32     presentclassicBind : classicFunctionBind
33   };
34   personne.presentclassique();
35   personne.presentarrow();
36   personne.presentclassicBind();
37 </script>
```

The bottom pane shows the console output. It contains three entries, all of which are the same object: `{nom: 'Jouini', prenom: 'Rim', presentclassique: [Function], presentarrow: [Function], presentclassicBind: [Function]}`. The first two entries are from the classic function and the arrow function, and the third is from the bound function.

Object	Source
{nom: 'Jouini', prenom: 'Rim', presentclassique: [Function], presentarrow: [Function], presentclassicBind: [Function]}	Fonction_Felchee_This.html:20
{nom: 'Jouini', prenom: 'Rim', presentclassique: [Function], presentarrow: [Function], presentclassicBind: [Function]}	Fonction_Felchee_This.html:24
{nom: 'Jouini', prenom: 'Rim', presentclassique: [Function], presentarrow: [Function], presentclassicBind: [Function]}	Fonction_Felchee_This.html:20

LA GESTION DES STRUCTURES DE DONNÉES

Pour la gestion des structures de données en EcmaScript 6 :

- Les tableaux (déjà disponibles dans les versions antérieurs)
- Les structures `Map`
- Les structures `Set`

EXEMPLE — LES TABLEAUX ET BOUCLE FOR ... OF

```
//  
//Définition d'un tableau JavaScript et affichage de son contenu  
//  
document.write("<h4> Tableau  JavaScript </h4>");  
  
//Définition d'un tableau JavaScript de voitures  
//let tabVoitures=["Porsche", "Ferrari", "BMW"];  
let tabVoitures= new Array("Porsche", "Ferrari", "BMW");  
  
//Ajout d'une 4 ème voiture  
tabVoitures.push("Peugeot");  
  
//Affichage de tableau voiture via for  
document.write("Affichage de tableau voiture (via for) : <br /> ");  
for (i=0; i<tabVoitures.length; i++)  
{  
    document.write(tabVoitures[i] + "<br />");  
}  
  
//Affichage des marques du tableau tabVoitures (via for of)  
document.write("<br /> Marques du tableau tabVoitures");  
document.write(" (via for of) : <br /> ");  
for ( let marque of tabVoitures)  
{  
    document.write(marque + "<br />");  
}  
  
//Affichage du nombre de voitures présentes dans le  tableau tabvoiture  
document.write("<br />Nombre de voitures du tableau tabvoiture : " + tabVoitures.length);
```

Tableau JavaScript

Affichage de tableau voiture (via for) :

Porsche
Ferrari
BMW
Peugeot

Marques du tableau tabVoitures (via for of) :

Porsche
Ferrari
BMW
Peugeot

Nombre de voitures du tableau tabvoiture : 4

EXEMPLE — LES MAP ET BOUCLE FOR ... OF

```
//  
//Définition d'un map JavaScript et affichage de son contenu  
//  
document.write("<h4> Map en JavaScript </h4>");  
//Définition d'un tableau JavaScript de voitures  
//Code en tant que clé, libelle en tant que valeur  
let mapVoitures = new Map([ ["porsche", "Porsche 911"],  
                             ["bmwwm5", "BMW 5"],  
                             ["renaut", "Clio Campus "]  
                             ] );  
  
//Ajout d'une 4 ème voiture  
mapVoitures.set("peugeot", "peugeot 206");  
  
//Affichage des codes voitures du mapVoitures  
document.write("Clés du Map mapVoitures: <br /> ");  
for ( let cle of mapVoitures.keys() )  
{  
    document.write("Code voiture : " + cle + " <br />");  
}  
  
//Affichage des libelle voitures du mapVoitures  
document.write("<br /> Valeurs (libelle)du Map mapVoitures: <br /> ");  
for ( let valeur of mapVoitures.values() )  
{  
    document.write("Libellé voiture : " + valeur + " <br />");  
}
```

Map en JavaScript

Clés du Map mapVoitures:

Code voiture : porsche

Code voiture : bmwwm5

Code voiture : renaut

Code voiture : peugeot

Valeurs (libelle)du Map mapVoitures:

Libellé voiture : Porsche 911

Libellé voiture : BMW 5

Libellé voiture : Clio Campus

Libellé voiture : peugeot 206

EXEMPLE — LES MAP ET BOUCLE FOR ... OF

```
//Affichage des voitures (cle, libelle) du mapVoitures
// via entries en tant qu'itérable
document.write("<br />");
document.write("Voitures du Map mapVoitures via entries en tant qu'itérable");
document.write("<br />");
for ( let voiture of mapVoitures.entries())
{
    document.write(voiture [0] + "- " + voiture[1] + " "<br />");
}

//Affichage des voitures (cle, libelle) du mapVoitures
// via entries destructuré
document.write("<br />");
document.write("Voitures du map mapVoitures via entries destructuré");
document.write("<br />");
for ( let [cle,valeur] of mapVoitures.entries())
{
    document.write(cle + "- " + valeur + " "<br />");
}

//Affichage du nombre de voitures présentes dans mapVoitures
document.write("<br /> Nombre de voitures du map mapVoitures : " + mapVoitures.size);
```

Voitures du Map mapVoitures via entries en tant qu'itérable
porsche- Porsche 911
bmwwm5- BMW 5
renaut- Clio Campus
peugeot- peugeot 206

Voitures du map mapVoitures via entries destructuré
porsche- Porsche 911
bmwwm5- BMW 5
renaut- Clio Campus
peugeot- peugeot 206

Nombre de voitures du map mapVoitures : 4

EXEMPLE — LES SET

```
//  
//Définition d'un Set JavaScript et affichage de son contenu  
//  
document.write("<h4> Set en JavaScript </h4>");  
//Définition d'un setJavaScript de voitures  
let setVoitures = new Set(["Porsche 911",  
                           "BMW 5",  
                           "Clio Campus"  
]);  
  
//Ajout d'une voiture supplémentaire dans le Set setVoiture  
setVoitures.add("peugeot 206");  
//Affichage du contenu du Set setVoiture  
document.write("Contenu su Set setVoitures <br />");  
for (voiture of setVoitures)  
{  
    document.write(voiture + "<br />");  
}  
//Affichage du nombre de voitures présentes dans le setVoitures  
document.write("<br />Nombre de voitures du Set mapVoitures : " + setVoitures.size);
```

Set en JavaScript

Contenu su Set setVoitures

Porsche 911

BMW 5

Clio Campus

peugeot 206

Nombre de voitures du Set mapVoitures : 4

PORTÉE DES VARIABLES

- La gestion de la portée (scope en anglais) des variables était problématique dans les versions antérieures de JavaScript.
- En EcmaScript6 Les variables ont une portée limitée et n'existent que dans le bloc où elles sont définies.
- Un bloc (`{ }`) enfant a un accès aux variables du bloc parents.

EXEMPLE — SCOPE DE BLOC

The screenshot shows the Chrome DevTools interface with the 'Sources' tab selected. The file 'les_variables.html' is open, showing the following code:

```
6 <head>
7 <!-- Balise meta -->
8 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10 <!-- Titre du script HTML -->
11 <title> Les variables </title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16 <!-- Début script JavaScript -->
17 <script>
18   function myFunction()
19   {
20     let a =10 ;
21     if (a>5)
22     {
23       console.log(a);
24     }
25   }
26   myFunction();
27 </script>
28 </body>
29
30 </html>
31
```

Annotations on the code:

- A purple bracket from line 18 to line 25 is labeled **Bloc parent**.
- A red bracket from line 22 to line 24 is labeled **Bloc enfant**.

Below the code, a text box states: **Le bloc enfant a accès aux variables du bloc parents**

The bottom of the image shows the 'Console' tab with a single log entry: `10`, indicating the value of `a` printed by `console.log(a)`.

EXEMPLE —SCOPE DE BLOC

```
1 <!DOCTYPE html>
2 <!-- Début script HTML -->
3 <html>
4
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10  <!-- Titre du script HTML -->
11  <title> Les variables </title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     function myFunction()
19     {
20       {
21         let a =10 ;
22       }
23       console.log(a);
24     }
25     myFunction();
26   </script>

```

Diagram illustrating block scope:

- Bloc parent**: The function `myFunction()` block (lines 18-25).
- Bloc enfant**: The inner block `{ let a =10 ; }` (lines 20-22).

The error message in the console is:

```
Uncaught ReferenceError: a is not defined
    at myFunction (file:///C:/MyDisk/IUT Aix-en-Provence/Anne%20Universitaire%202022-2023/R4.A.10%20Complement%20Web%20(JavaScript)/Amphi/Approche%20Objet%20en%20JavaScript/les_variables.html:23:17)
    at file:///C:/MyDisk/IUT Aix-en-Provence/Anne%20Universitaire%202022-2023/R4.A.10%20Complement%20Web%20(JavaScript)/Amphi/Approche%20Objet%20en%20JavaScript/les_variables.html:25:4
```

Console.log est dans le bloc parent et il essaye d'avoir accès à une variable dans le bloc enfant

EXEMPLE — SCOPE DE BLOC

Page >> les_variables.html x

```
1 <!DOCTYPE html>
2 <!-- Début script HTML -->
3 <html>
4
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10  <!-- Titre du script HTML -->
11  <title> Les variables </title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     function myFunction()
19     {
20       let a =10 ;
21       {
22         console.log(a);
23       }
24     }
25   }
26   myFunction();
```

Bloc enfant

Bloc parent

Le bloc enfant a accès aux variables du bloc parents

Ligne 26, colonne 17

Couverture : N/A

Console Couverture Problèmes

top Filtre Niveaux par défaut Aucun problème

10 les_variables.html:22

EXEMPLE — SCOPE DE FONCTION

The screenshot shows the Chrome DevTools interface with the 'Sources' panel open. The file 'les_variables.html' is loaded, and the code is visible. The code defines a function 'myFunction' which contains an 'if' block with a 'var a=5;' declaration. Annotations include a blue text box stating 'var répond à la règle de Scope de fonction' and a pink box labeled 'Bloc enfant' pointing to the 'if' block, with a green box labeled 'Bloc parent' pointing to the function body. The status bar at the bottom indicates 'Ligne 26, colonne 17' and 'Couverture : N/A'.

```
1 <!DOCTYPE html>
2 <!-- Début script HTML -->
3 <html>
4
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10  <!-- Titre du script HTML -->
11  <title> Les variables </title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     function myFunction()
19     {
20       if(true)
21       {
22         var a=5;
23       }
24       console.log(a);
25     }
26     myFunction();
```

var répond à la règle de Scope de fonction

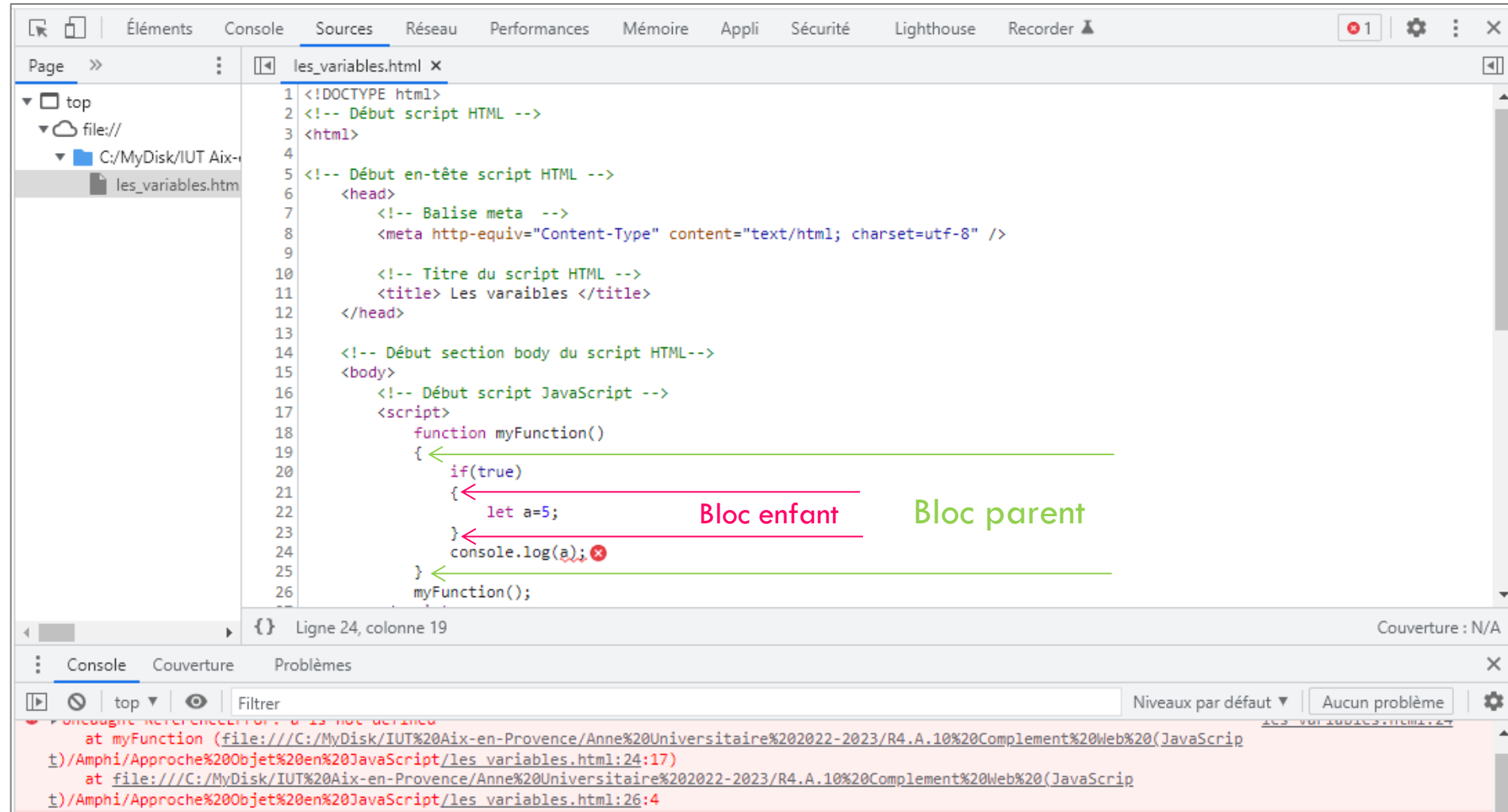
Bloc enfant

Bloc parent

Ligne 26, colonne 17

Couverture : N/A

EXEMPLE — SCOPE DE FONCTION



Le bloc parent n'a pas accès aux variable déclarées dans le bloc enfant

EXEMPLE

```
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10  <!-- Titre du script HTML -->
11  <title> Les variables </title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     var i=50;
19     for (var i=0; i<10;i++)
20     {
21       console.log(i);
22     }
23     console.log(i);
24   </script>
25 </body>
26
```

⋮ Console Couverture Problèmes Affichage

▶ 🔇 top 👁 Filtrer

Niveaux par défaut ▼ | Aucun problème | ⚙

0	les_variables.html:21
1	les_variables.html:21
2	les_variables.html:21
3	les_variables.html:21
4	les_variables.html:21
5	les_variables.html:21
6	les_variables.html:21
7	les_variables.html:21
8	les_variables.html:21
9	les_variables.html:21
10	les_variables.html:23
>	

EXEMPLE

```
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10  <!-- Titre du script HTML -->
11  <title> Les variables </title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     let i=50;
19     for (let i=0; i<10;i++)
20     {
21       console.log(i);
22     }
23     console.log(i);
24   </script>
25 </body>
```

Bloc enfant Bloc parent

Console		Couverture	Problèmes	Affichage		
▶	🔇	top ▼	👁	Filtrer	Niveaux par défaut ▼	Aucun problème ⚙
0					les_variables.html:21	
1					les_variables.html:21	
2					les_variables.html:21	
3					les_variables.html:21	
4					les_variables.html:21	
5					les_variables.html:21	
6					les_variables.html:21	
7					les_variables.html:21	
8					les_variables.html:21	
9					les_variables.html:21	
50					les_variables.html:23	
>						

→ La chaîne des Scope a un ordre de préférence

EXEMPLE — VAR HOISTING

The screenshot displays the Chrome DevTools interface. The 'Sources' panel shows a file named `les_variables.html` with the following code:

```
1 <!DOCTYPE html>
2 <!-- Début script HTML -->
3 <html>
4
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10  <!-- Titre du script HTML -->
11  <title> Les variables </title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     console.log(x);
19     var x;
20   </script>
21 </body>
22
23 </html>
```

The status bar at the bottom of the Sources panel indicates the cursor is at `Ligne 17, colonne 11` and `Couverture : N/A`.

The 'Console' panel at the bottom shows a single log entry: `undefined`, which occurred at `les_variables.html:18`. This is the result of the `console.log(x);` statement being executed before the variable `x` is declared.

EXEMPLE — LET HOISTING

The screenshot displays a web browser's developer tools interface. The top pane shows the source code of a file named `les_variables.html`. The code is an HTML document with a script section. The script contains a `console.log(x);` statement on line 18, followed by a `let x;` declaration on line 19. The error message in the bottom pane indicates an `Uncaught ReferenceError: Cannot access 'x' before initialization` at line 18, column 16. The error message also includes the file path: `file:///C:/MyDisk/IUT%20Aix-en-Provence/Anne%20Universitaire%202022-2023/R4.A.10%20Complement%20Web%20(JS`.

The source code in the top pane is as follows:

```
1 <!DOCTYPE html>
2 <!-- Début script HTML -->
3 <html>
4
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10   <!-- Titre du script HTML -->
11   <title> Les variables </title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     console.log(x);
19     let x;
20   </script>
21 </body>
22
23 </html>
```

The error message in the bottom pane is:

```
Uncaught ReferenceError: Cannot access 'x' before initialization
    at file:///C:/MyDisk/IUT%20Aix-en-Provence/Anne%20Universitaire%202022-2023/R4.A.10%20Complement%20Web%20(JS
    t)/Amphi/Approche%20Objet%20en%20JavaScript/les_variables.html:18:16
```

DESTRUCTURING

L'affectation par décomposition (*destructuring*) est une expression JavaScript qui permet d'extraire des données d'un tableau ou d'un objet.

La syntaxe ressemble à la structure du tableau.

```
Let [variable1 , variable2 , variable3 ] = [val1, val2, val3, val4 , val5]
```

```
Let [variable1 , variable2 , variable3 ] = nom_tbaleau
```

```
Let [variable1 , variable2 , variable3 ] = nom_Objet
```

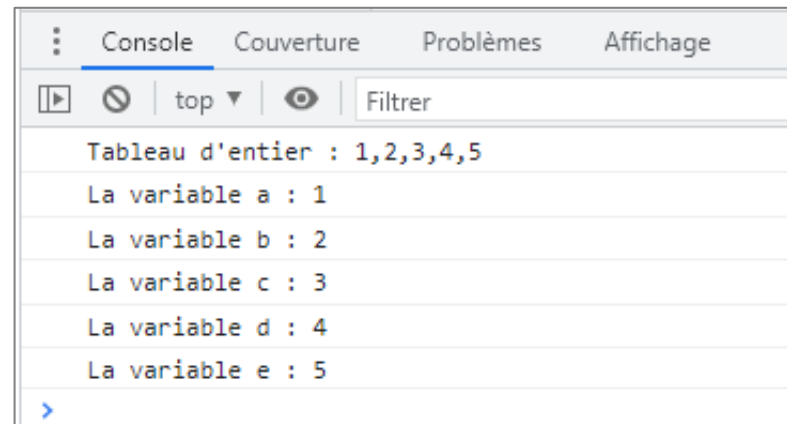
EXEMPLE — DESTRUCTURIN ARRAY

```
<!-- Début section body du script HTML-->
<body>
  <!-- Début script JavaScript -->
  <script>
    const tabEntier = [1, 2, 3, 4, 5];
    const a = tabEntier [0];
    const b = tabEntier [1];
    const c = tabEntier [2];
    const d = tabEntier [3];
    const e = tabEntier [4];
    console.log(" Tableau d'entier : " + tabEntier);
    console.log(" La variable a : " + a);
    console.log(" La variable b : " + b);
    console.log(" La variable c : " + c);
    console.log(" La variable d : " + d);
    console.log(" La variable e : " + e);
  </script>
</body>
```

≠

```
<!-- Début section body du script HTML-->
<body>
  <!-- Début script JavaScript -->
  <script>
    const tabEntier = [1, 2, 3, 4, 5];
    const [a,b,c,d,e] = tabEntier;
    console.log(" Tableau d'entier : " + tabEntier);
    console.log(" La variable a : " + a);
    console.log(" La variable b : " + b);
    console.log(" La variable c : " + c);
    console.log(" La variable d : " + d);
    console.log(" La variable e : " + e);
  </script>
</body>
</html>
```

Destructuring array



EXEMPLE - DESTRUCTURIN

The screenshot displays a web browser window with a file explorer on the left and a console at the bottom. The file explorer shows the path `C:/MyDisk/IUT Aix-les-variables.html`. The main content area shows the HTML code for `les_variables.html`, which includes a meta tag for content type and charset, a title "Les variables", and a body section containing a JavaScript script. The script defines an array `tabEntier` with values [1, 2, 3, 4, 5] and uses destructuring to assign values to variables `a`, `b`, `c`, `d`, and `e`. It then logs these values to the console. The console output shows the following messages:

- La variable a : 1
- La variable b : 2
- La variable c : 3
- La variable d : 4
- La variable e : 5
- La variable e : undefined

The console also shows the source file and line number for each log statement: `les_variables.html:21`, `les_variables.html:22`, `les_variables.html:23`, `les_variables.html:24`, `les_variables.html:25`, and `les_variables.html:26`.

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<!-- Titre du script HTML -->
<title> Les variables </title>
</head>
<!-- Début section body du script HTML-->
<body>
  <!-- Début script JavaScript -->
  <script>
    const tabEntier = [1, 2, 3, 4, 5];
    const [a,b,c,d,e, f] = tabEntier;
    console.log(" Tableau d'entier : " + tabEntier);
    console.log(" La variable a : " + a);
    console.log(" La variable b : " + b);
    console.log(" La variable c : " + c);
    console.log(" La variable d : " + d);
    console.log(" La variable e : " + e);
    console.log(" La variable e : " + f);
  </script>
</body>
```

EXEMPLE - DESTRUCTURING AVEC PARAMÈTRE PAR DÉFAUT

The screenshot shows a web browser's developer tools interface. The 'Sources' panel is open, displaying a file named `les_variables.html`. The code in the file includes a meta tag for content type, a title, and a JavaScript script. The script defines an array `tabEntier` and uses destructuring with a default value for `f` to log the values of `a`, `b`, `c`, `d`, `e`, and `f`.

```
7 <!-- Balise meta -->
8 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10 <!-- Titre du script HTML -->
11 <title> Les variables </title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16 <!-- Début script JavaScript -->
17 <script>
18   const tabEntier = [1, 2, 3, 4, 5];
19   const [a,b,c,d,e, f=10] = tabEntier;
20   console.log(" Tableau d'entier : " + tabEntier);
21   console.log(" La variable a : " + a);
22   console.log(" La variable b : " + b);
23   console.log(" La variable c : " + c);
24   console.log(" La variable d : " + d);
25   console.log(" La variable e : " + e);
26   console.log(" La variable e : " + f);
27 </script>
28
```

The 'Console' panel at the bottom shows the output of the script, displaying the values of the variables `a` through `e` and `f` (labeled as `e` in the log).

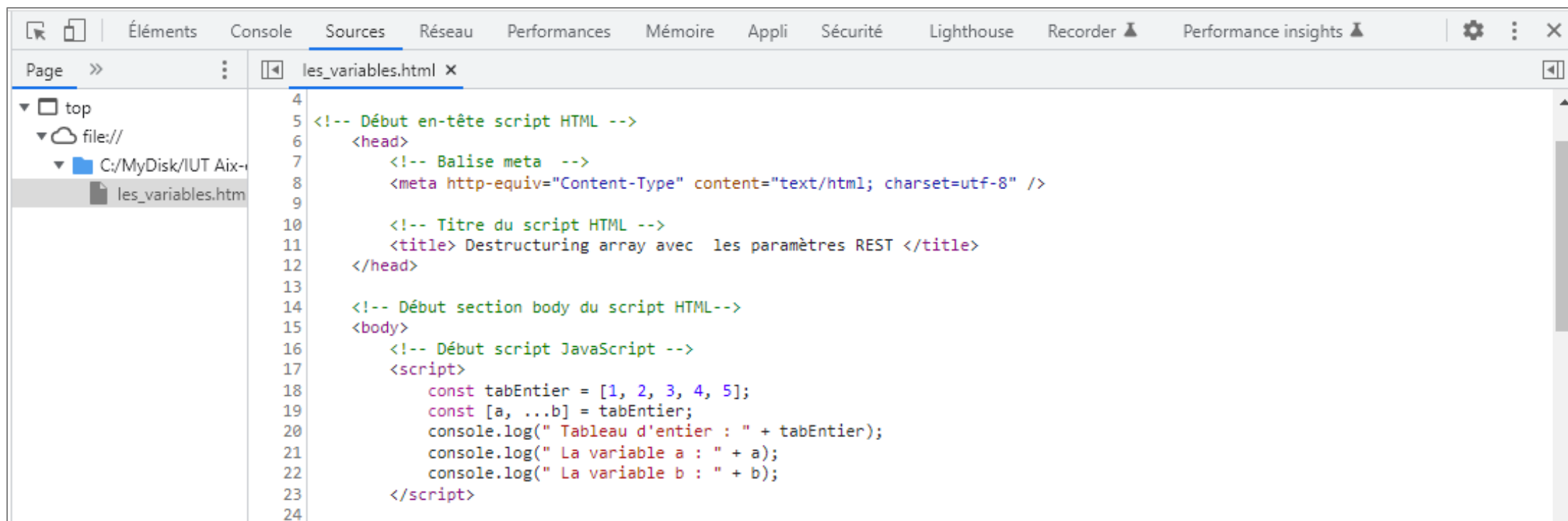
Message	Source
La variable a : 1	les_variables.html:21
La variable b : 2	les_variables.html:22
La variable c : 3	les_variables.html:23
La variable d : 4	les_variables.html:24
La variable e : 5	les_variables.html:25
La variable e : 10	les_variables.html:26

EXEMPLE

The screenshot displays a web browser window with a single tab titled 'les_variables.html'. The browser's address bar shows the file path 'C:/MyDisk/IUT Aix-les-variables.html'. The page content is an HTML document with a title 'Destructuring array' and a JavaScript script. The script defines an array 'tabEntier' with values [1, 2, 3, 4, 5] and then uses destructuring to assign values to variables 'a', 'b', 'c', 'd', 'e', and 'f'. The console output shows the following messages:

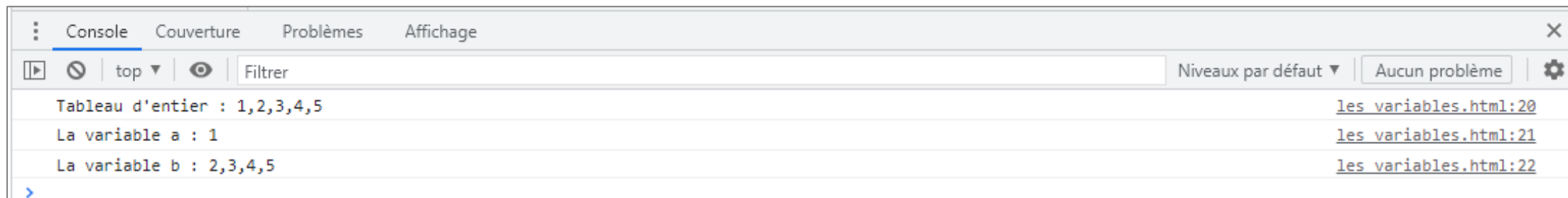
Message	Source
Tableau d'entier : 1,2,3,4,5	les_variables.html:20
La variable a : 1	les_variables.html:21
La variable b : 2	les_variables.html:22
La variable c : 3	les_variables.html:23
La variable d : 4	les_variables.html:24
La variable e : 5	les_variables.html:25
La variable e : 10	les_variables.html:26

EXEMPLE



The screenshot shows the 'Sources' panel of a web browser's developer tools. The file 'les_variables.html' is selected, and its content is displayed. The code is an HTML document with a head section containing a meta tag for content type and charset, and a title 'Destructuring array avec les paramètres REST'. The body section contains a JavaScript script that defines a constant array 'tabEntier' with values [1, 2, 3, 4, 5], deconstructs it into 'a' and 'b', and logs them to the console.

```
4
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10  <!-- Titre du script HTML -->
11  <title> Destructuring array avec les paramètres REST </title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     const tabEntier = [1, 2, 3, 4, 5];
19     const [a, ...b] = tabEntier;
20     console.log(" Tableau d'entier : " + tabEntier);
21     console.log(" La variable a : " + a);
22     console.log(" La variable b : " + b);
23   </script>
24
```



The screenshot shows the 'Console' panel of the browser's developer tools. It displays the output of the JavaScript code: 'Tableau d'entier : 1,2,3,4,5', 'La variable a : 1', and 'La variable b : 2,3,4,5'. Each line is linked to its corresponding line in the source file.

Console	Couverture	Problèmes	Affichage
top	Filter	Niveaux par défaut	Aucun problème
Tableau d'entier : 1,2,3,4,5			les_variables.html:20
La variable a : 1			les_variables.html:21
La variable b : 2,3,4,5			les_variables.html:22

EXEMPLE

The image shows a web browser's developer tools interface. The top panel displays the source code of a file named `les_variables.html`. The code is an HTML document with a head section containing a meta tag for content type and charset, and a title "Destructuring - sauter des valeurs". The body section contains a JavaScript script that defines an array `tabEntier` with values [1, 2, 3, 4, 5], deconstructs it into variables `a` and `b`, and logs them to the console.

```
2 <!-- Début script HTML -->
3 <html>
4
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10  <!-- Titre du script HTML -->
11  <title> Destructuring - sauter des valeurs</title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     const tabEntier = [1, 2, 3, 4, 5];
19     const [a, ,b] = tabEntier;
20     console.log(" Tableau d'entier : " + tabEntier);
21     console.log(" La variable a : " + a);
22     console.log(" La variable b : " + b);
23   </script>
```

The bottom panel shows the console output of the script. It displays three log messages: "Tableau d'entier : 1,2,3,4,5", "La variable a : 1", and "La variable b : 3". Each message is linked to its corresponding line in the source code.

Message	Source
Tableau d'entier : 1,2,3,4,5	les_variables.html:20
La variable a : 1	les_variables.html:21
La variable b : 3	les_variables.html:22

EXEMPLE

The image shows a web browser's developer tools interface. The top panel displays the source code of a file named `les_variables.html`. The code is an HTML document with a head section containing a meta tag for content type and charset, and a title "Destructuring - sauter des valeurs". The body section contains a JavaScript script that uses array destructuring to assign values to variables `a` and `b`, and then logs them to the console.

```
1 <!DOCTYPE html>
2 <!-- Début script HTML -->
3 <html>
4
5 <!-- Début en-tête script HTML -->
6   <head>
7     <!-- Balise meta -->
8     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10    <!-- Titre du script HTML -->
11    <title> Destructuring - sauter des valeurs</title>
12  </head>
13
14  <!-- Début section body du script HTML-->
15  <body>
16    <!-- Début script JavaScript -->
17    <script>
18      const [a, ,b] = [1, 2, 3, 4, 5];
19      console.log(" La variable a : " + a);
20      console.log(" La variable b : " + b);
21    </script>
```

The bottom panel shows the console output, which displays the results of the JavaScript execution:

```
La variable a : 1
La variable b : 3
```

The console also shows the source file and line numbers for each log statement: `les_variables.html:19` and `les_variables.html:20`.

EXEMPLE - DESTRUCTURING & PERMUTATION

```
<!-- Titre du script HTML -->
<title> Permutation </title>
</head>

<!-- Début section body du script HTML-->
<body>
  <!-- Début script JavaScript -->
  <script>
    let a= 5
    let b=10
    console.log("La valeur des variables avant la permutation");
    console.log(a);
    console.log(b);
    let c=a
    a=b
    b=c
    console.log("La valeur des variables après la permutation");
    console.log(a);
    console.log(b);

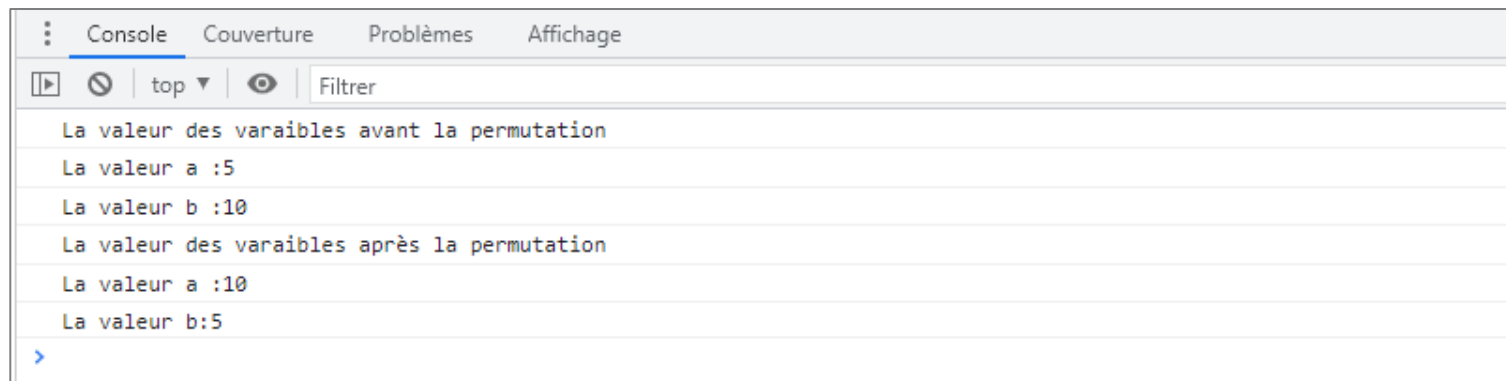
  </script>
```

```
<!-- Titre du script HTML -->
<title> Destructuring & Permutation </title>
</head>

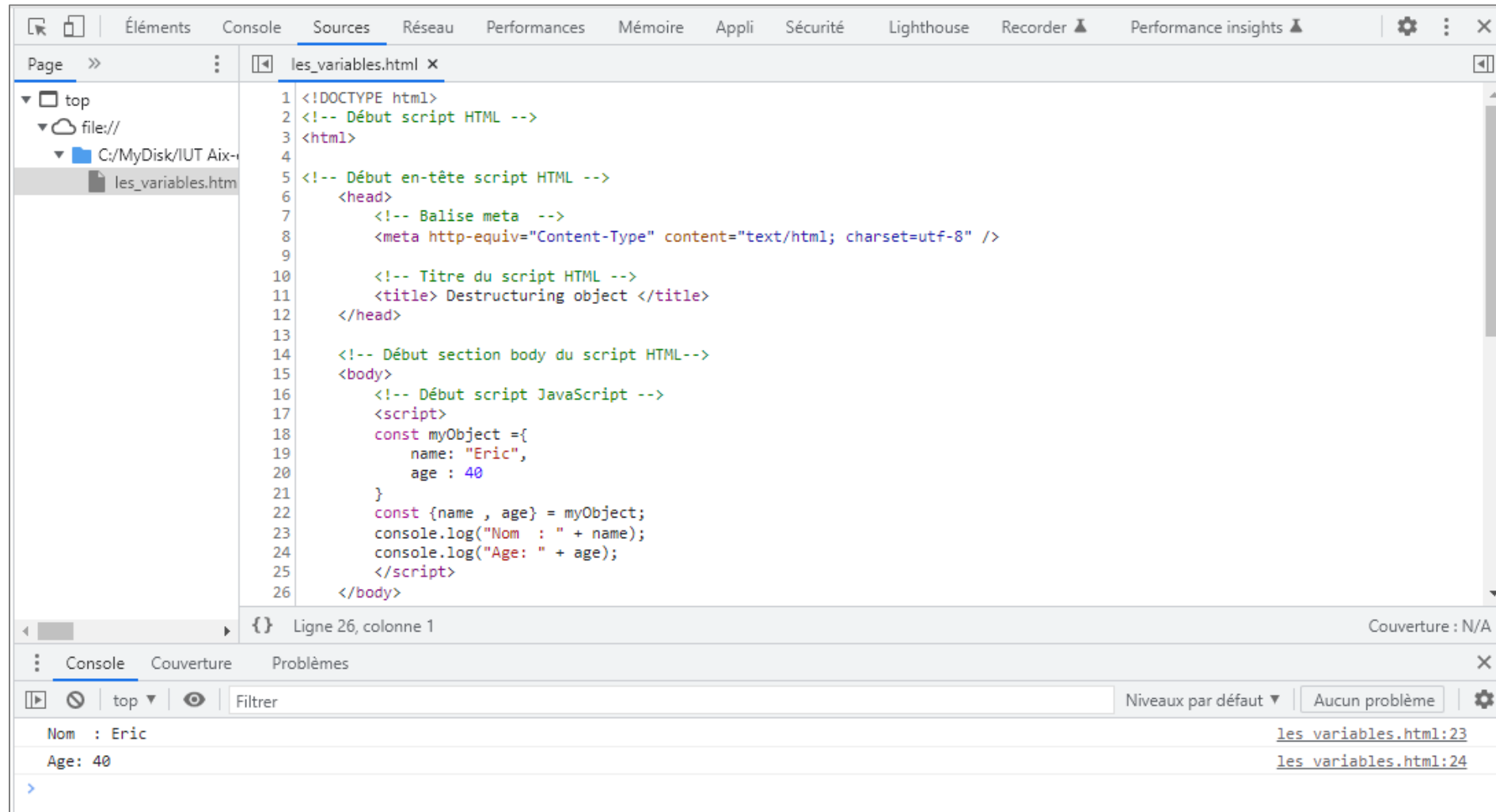
<!-- Début section body du script HTML-->
<body>
  <!-- Début script JavaScript -->
  <script>
    let a = 5
    let b = 10
    console.log("La valeur des variables avant la permutation");
    console.log("La valeur a :" + a);
    console.log("La valeur b :" + b);
    console.log("La valeur des variables après la permutation");
    [b,a]=[a,b];
    console.log("La valeur a :" + a);
    console.log("La valeur b:" + b);
  </script>

</body>

</html>
```



EXEMPLE - DESTRUCTURING OBJETS POUR CRÉER DES VARIABLES



The screenshot displays a web browser's developer tools interface. The top panel shows the 'Sources' tab with a file named 'les_variables.html' open. The code in the editor is as follows:


```
1 <!DOCTYPE html>
2 <!-- Début script HTML -->
3 <html>
4
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10   <!-- Titre du script HTML -->
11   <title> Destructuring object </title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     const myObject = {
19       name: "Eric",
20       age : 40
21     }
22     const {name , age} = myObject;
23     console.log("Nom : " + name);
24     console.log("Age: " + age);
25   </script>
26 </body>
```

The bottom panel shows the 'Console' tab with the following output:

```
Nom : Eric
Age: 40
```

The console also shows the source files for the log statements: 'les_variables.html:23' and 'les_variables.html:24'.

EXEMPLE - DESTRUCTURING OBJETS POUR CRÉER DES VARIABLES



```
1 <!DOCTYPE html>
2 <!-- Début script HTML -->
3 <html>
4
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10  <!-- Titre du script HTML -->
11  <title> Destructuring object </title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     const myObject = {
19       name: "Eric",
20       age : 40
21     }
22     const {varName , varAge} = myObject;
23     console.log("Nom : " + varName);
24     console.log("Age: " + varAge);
25   </script>
26 </body>
```

Ligne 23, colonne 11

Couverture : N/A

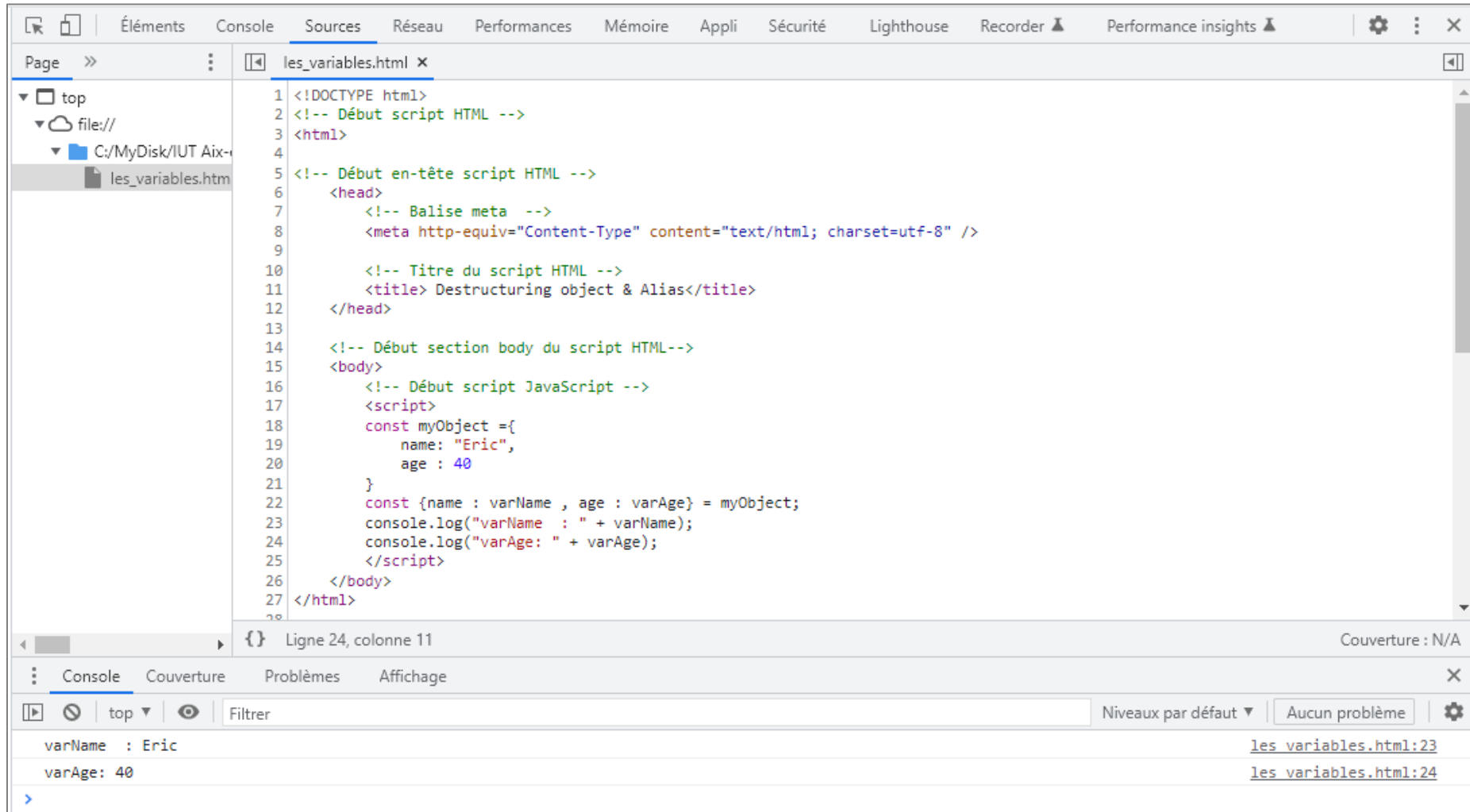
Console

Niveau par défaut

Aucun problème

Nom : undefined	les_variables.html:23
Age: undefined	les_variables.html:24

EXEMPLE - DESTRUCTURING OBJETS POUR CRÉER DES VARIABLES



The screenshot displays a web browser's developer tools interface. The 'Sources' panel is open, showing a file named 'les_variables.html'. The code in the file defines a constant object 'myObject' with properties 'name' (value 'Eric') and 'age' (value 40). It then uses destructuring to create variables 'varName' and 'varAge' from 'myObject' and logs their values to the console. The 'Console' panel at the bottom shows the output of these logs.

```
1 <!DOCTYPE html>
2 <!-- Début script HTML -->
3 <html>
4
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10  <!-- Titre du script HTML -->
11  <title> Destructuring object & Alias</title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     const myObject = {
19       name: "Eric",
20       age : 40
21     }
22     const {name : varName , age : varAge} = myObject;
23     console.log("varName : " + varName);
24     console.log("varAge: " + varAge);
25   </script>
26 </body>
27 </html>
```

Ligne 24, colonne 11

Couverture : N/A

Console

varName : Eric

varAge: 40

les_variables.html:23

les_variables.html:24

EXEMPLE

The screenshot shows a web browser's developer tools interface. The 'Sources' panel is active, displaying the file `les_variables.html`. The code in the editor is as follows:

```
9
10     <!-- Titre du script HTML -->
11     <title> Destructuring object pour créer des fonctions</title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16     <!-- Début script JavaScript -->
17     <script>
18         const myObject = {
19             name: "Eric",
20             age : 40,
21
22             presentYourSelf : function()
23             {
24                 console.log("Bonjour! Je me présente: je m'appelle " + myObject.name + " et j'ai " + myObject.age);
25             }
26         }
27         myObject.presentYourSelf();
28     </script>
29 </body>
30 </html>
31
```

The status bar at the bottom of the editor indicates "Ligne 11, colonne 39" and "Couverture : N/A".

Below the editor, the 'Console' panel is open, showing a single log message: "Bonjour! Je me présente: je m'appelle Eric et j'ai 40". The message is linked to the source file at `les_variables.html:24`.

EXEMPLE - DESTRUCTURING OBJETS POUR CRÉER DES FONCTIONS

```
6 <head>
7 <!-- Balise meta -->
8 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10 <!-- Titre du script HTML -->
11 <title> Destructuring object pour créer des fonctions</title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16 <!-- Début script JavaScript -->
17 <script>
18   const myObject = {
19     name: "Eric",
20     age : 40,
21
22     presentYourSelf : function()
23     {
24       console.log("Bonjour! Je me présente: je m'appelle " + myObject.name + " et j'ai " + myObject.age);
25     }
26   }
27   const {name, presentYourSelf} = myObject;
28   presentYourSelf();
29 </script>
30 </body>
31 </html>
```

Ligne 11, colonne 39

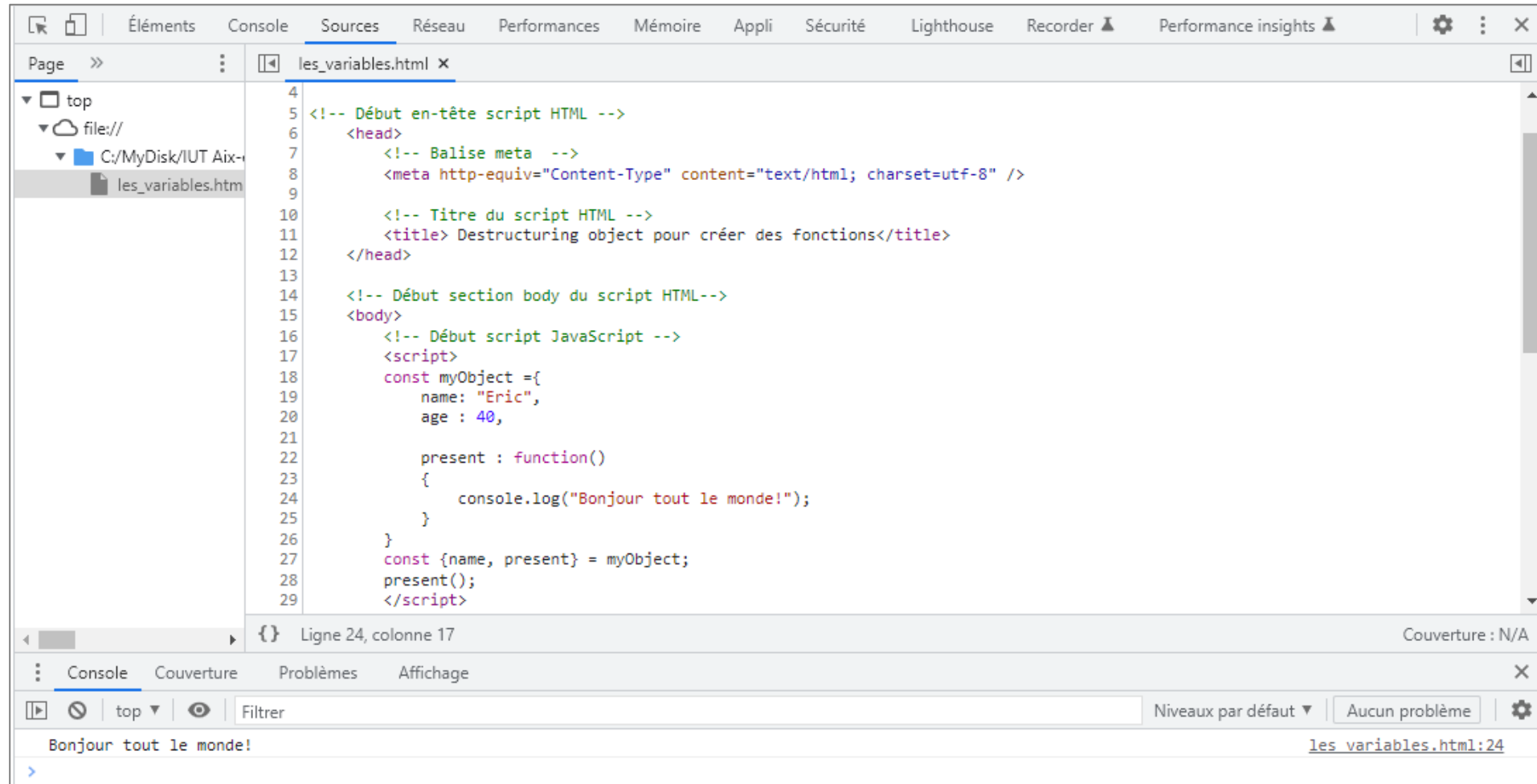
Couverture : N/A

Console Couverture Problèmes Affichage

top Filtre Niveaux par défaut Aucun problème

Bonjour! Je me présente: je m'appelle Eric et j'ai 40 [les_variables.html:24](#)

EXEMPLE - DESTRUCTURING OBJETS POUR CRÉER DES FONCTIONS



```
4
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10  <!-- Titre du script HTML -->
11  <title> Destructuring object pour créer des fonctions</title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     const myObject = {
19       name: "Eric",
20       age : 40,
21
22       present : function()
23       {
24         console.log("Bonjour tout le monde!");
25       }
26     }
27     const {name, present} = myObject;
28     present();
29   </script>
```

Ligne 24, colonne 17

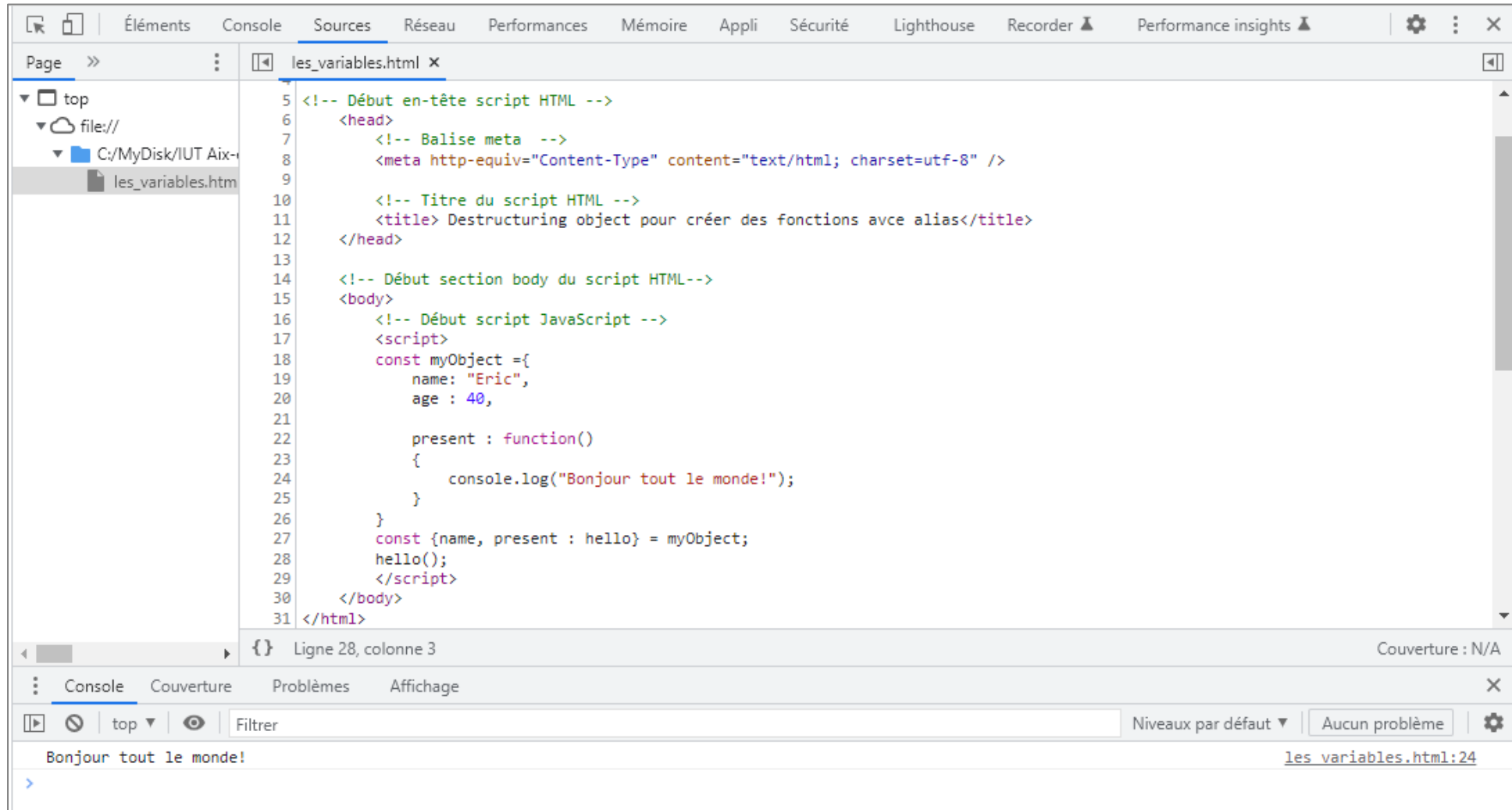
Couverture : N/A

Console Couverture Problèmes Affichage

top Filtre Niveaux par défaut Aucun problème

Bonjour tout le monde! les_variables.html:24

EXEMPLE - DESTRUCTURING OBJETS POUR CRÉER DES FONCTIONS

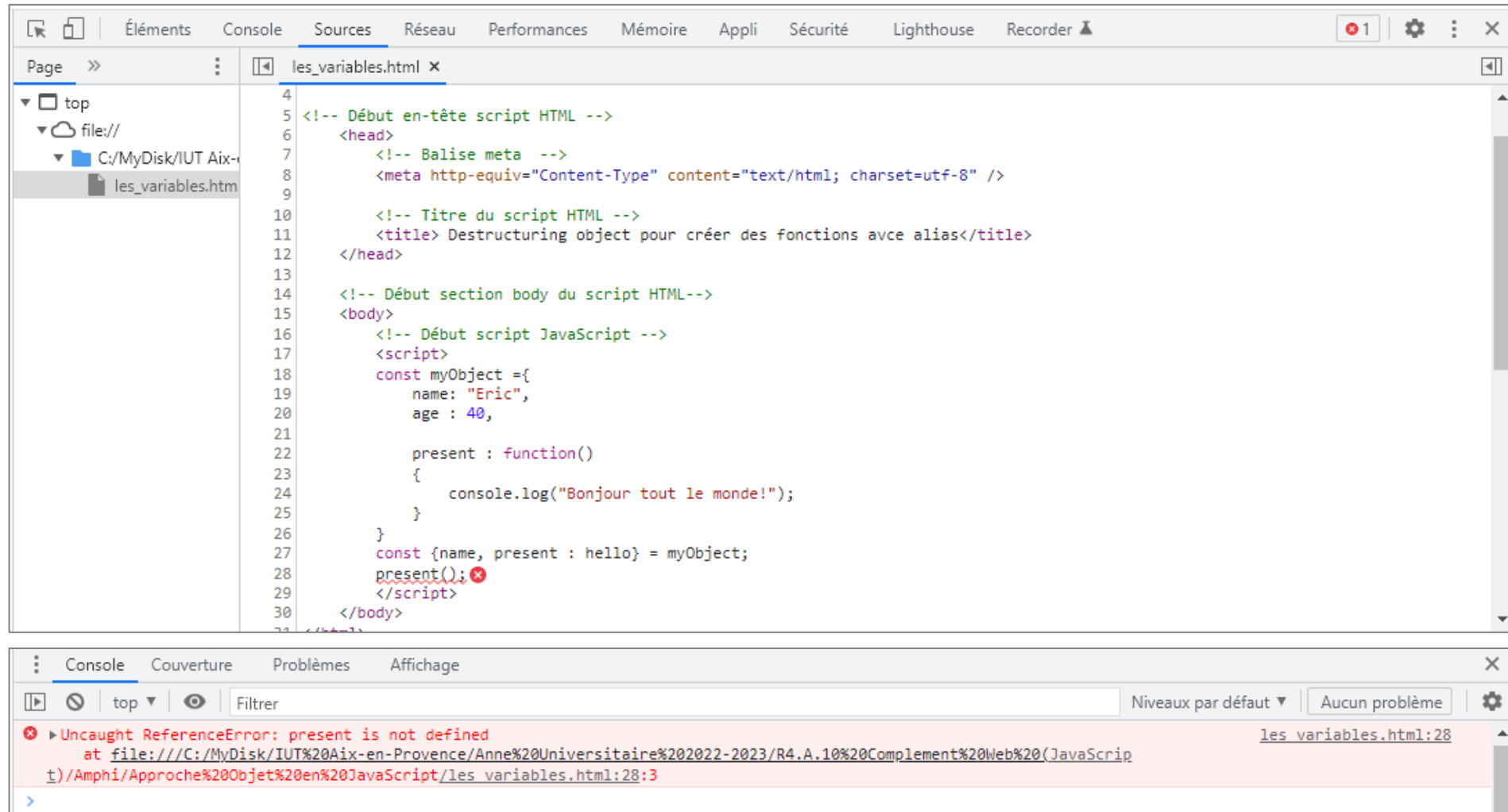


The screenshot shows a web browser's developer tools interface. The 'Sources' panel is active, displaying the file 'les_variables.html'. The code in the file is as follows:

```
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10  <!-- Titre du script HTML -->
11  <title> Destructuring object pour créer des fonctions avec alias</title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     const myObject = {
19       name: "Eric",
20       age : 40,
21
22       present : function()
23       {
24         console.log("Bonjour tout le monde!");
25       }
26     }
27     const {name, present : hello} = myObject;
28     hello();
29   </script>
30 </body>
31 </html>
```

The status bar at the bottom indicates 'Ligne 28, colonne 3' and 'Couverture : N/A'. The 'Console' panel is also visible, showing the output 'Bonjour tout le monde!' from the log statement in the script.

EXEMPLE - DESTRUCTURING OBJETS POUR CRÉER DES FONCTIONS



```
4
5 <!-- Début en-tête script HTML -->
6 <head>
7   <!-- Balise meta -->
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9
10  <!-- Titre du script HTML -->
11  <title> Destructuring object pour créer des fonctions avec alias</title>
12 </head>
13
14 <!-- Début section body du script HTML-->
15 <body>
16   <!-- Début script JavaScript -->
17   <script>
18     const myObject = {
19       name: "Eric",
20       age : 40,
21
22       present : function()
23       {
24         console.log("Bonjour tout le monde!");
25       }
26     }
27     const {name, present : hello} = myObject;
28     present();
29   </script>
30 </body>
```

Console

Uncaught ReferenceError: present is not defined
at file:///C:/MyDisk/IUT%20Aix-en-Provence/Anne%20Universitaire%202022-2023/R4.A.10%20Complement%20Web%20(JS...
t)/Amphi/Approche%20Objet%20en%20JavaScript/les_variables.html:28:3

PROMESSE (PROMISE)

- L'objet **Promise** (pour « promesse ») est utilisé pour réaliser des traitements de façon asynchrone.
- Une promesse représente une opération qui n'a pas encore été complétée, mais qui est attendue dans la futur.
- La promesse est une **fonction** utilisant deux paramètres **resolve** et **reject** (aussi de type **fonction**).
- L'exploitation de la promesse est réalisée à l'aide des deux méthodes qu'elle possède :
 - **then** : cas exécuté si la promesse aboutit (**resolve**)
 - **catch** : cas exécuté si la promesse est en échec (**reject**)

EXEMPLE — PROMISE : DÉFINITION

```
//Définition d'une promesse
//NB 1 : le traitement consiste à tester la vitesse atteinte
// par 2 voitures de sport au bout de 5 secondes
//NB 2 : la voiture qui sera gagnante est désignée de manière aléatoire par Math.random
//Math.random fournit une réponse entre 0 et 1
document.write("<h4> Démarrage de deux voitures</h4>");
document.write("Porsche 911 vs Ferrari Testarossa<br />");
var maPromesse = new Promise (function(resolve, reject){
    //Délai de 5 secondes
    //setTimeout(function(){traitement}, 5000);
    setTimeout(function()
    {
        if (Math.random()>0.5){

            // Porsche 911
            // NB : le choix de mettre cette voiture dans le cas de resolve est arbitraire
            resolve("La Porsche 911 est la plus rapide ");

        }
        else
        {
            //Ferrari Testarossa
            // NB : le choix de mettre cette voiture dans le cas de reject est arbitraire
            reject("La Ferrari Testarossa est la plus rapide ");

        }
    }, 5000);
});
```

EXEMPLE — PROMISE : EXPLOITATION

```
// Affichage que la promesse n'a pas encore été traitée
// (traitement aynchrone)
document.write(" Controle radar dans 5 secondes");

//Compte rendu du fonctionnement de la promesse
maPromesse
    .then(function (verdict) {
        //Cas Porsche 911
        alert(" verdict : " + verdict);
    })
    .catch(function (verdict){
        //Cas Ferrari Testarossa
        alert(" verdict : " + verdict);
    });
```

EXEMPLE —PROMISE

EcmaScript 6 : Utilisation des promesses

Démarrage de deux voitures

Porsche 911 vs Ferrari Testarossa
Controle radar dans 5 secondes

Cette page indique

verdict : La Ferrari Testarossa est la plus rapide

OK