

TECHNOLOGIES WEB AVANCÉES



R. TOMCZAK
vendredi 14 février 2025

Présentation Générale

- ◎ Robert TOMCZAK

- ◎ Cours complet sur moodle

- ◎ <https://moodle.uphf.fr/course/view.php?id=6353>

- ◎ Mdp = tomczak

- ◎ Diapo = screenshots

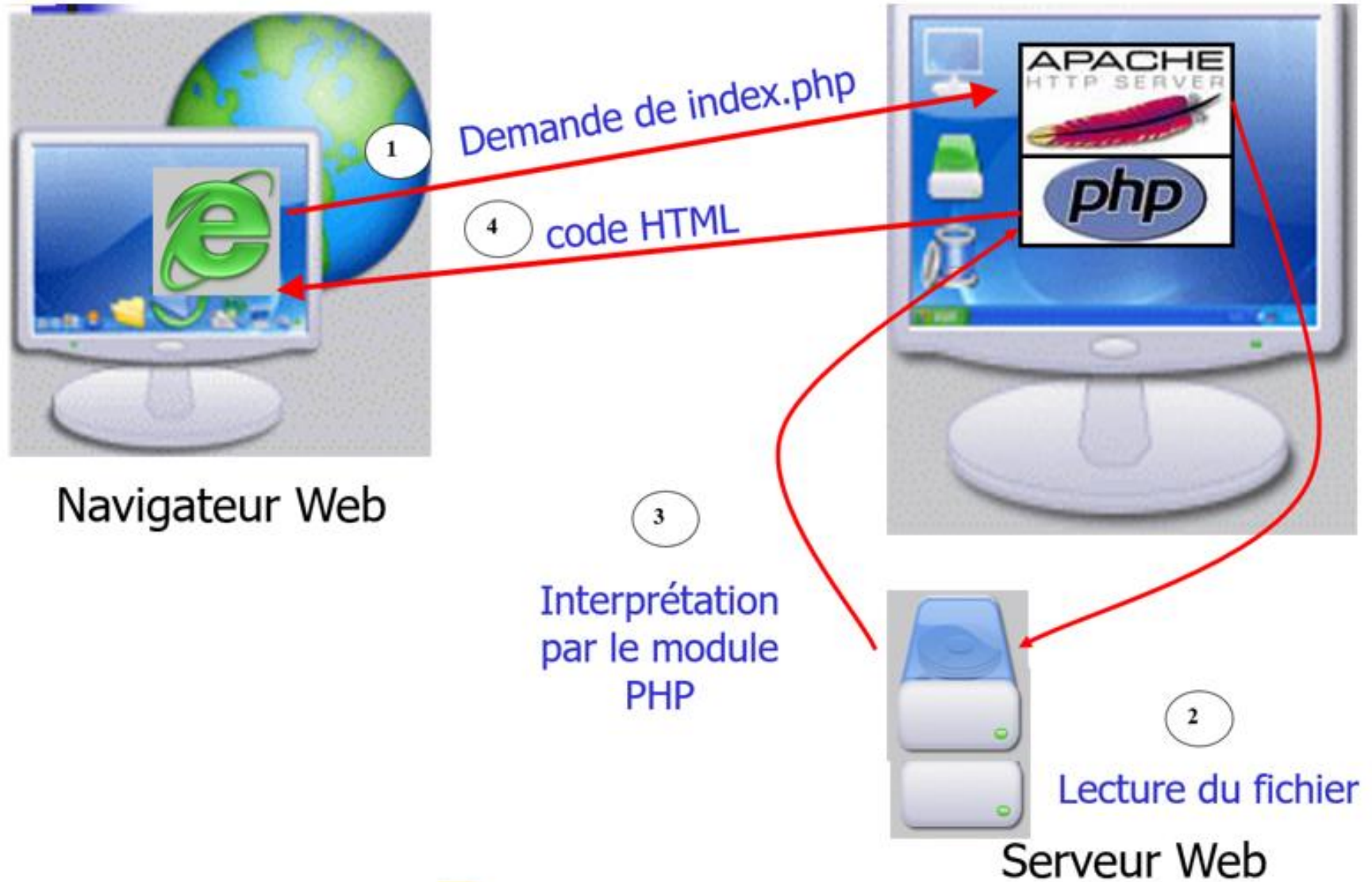
Plan

Plan du premier Cours

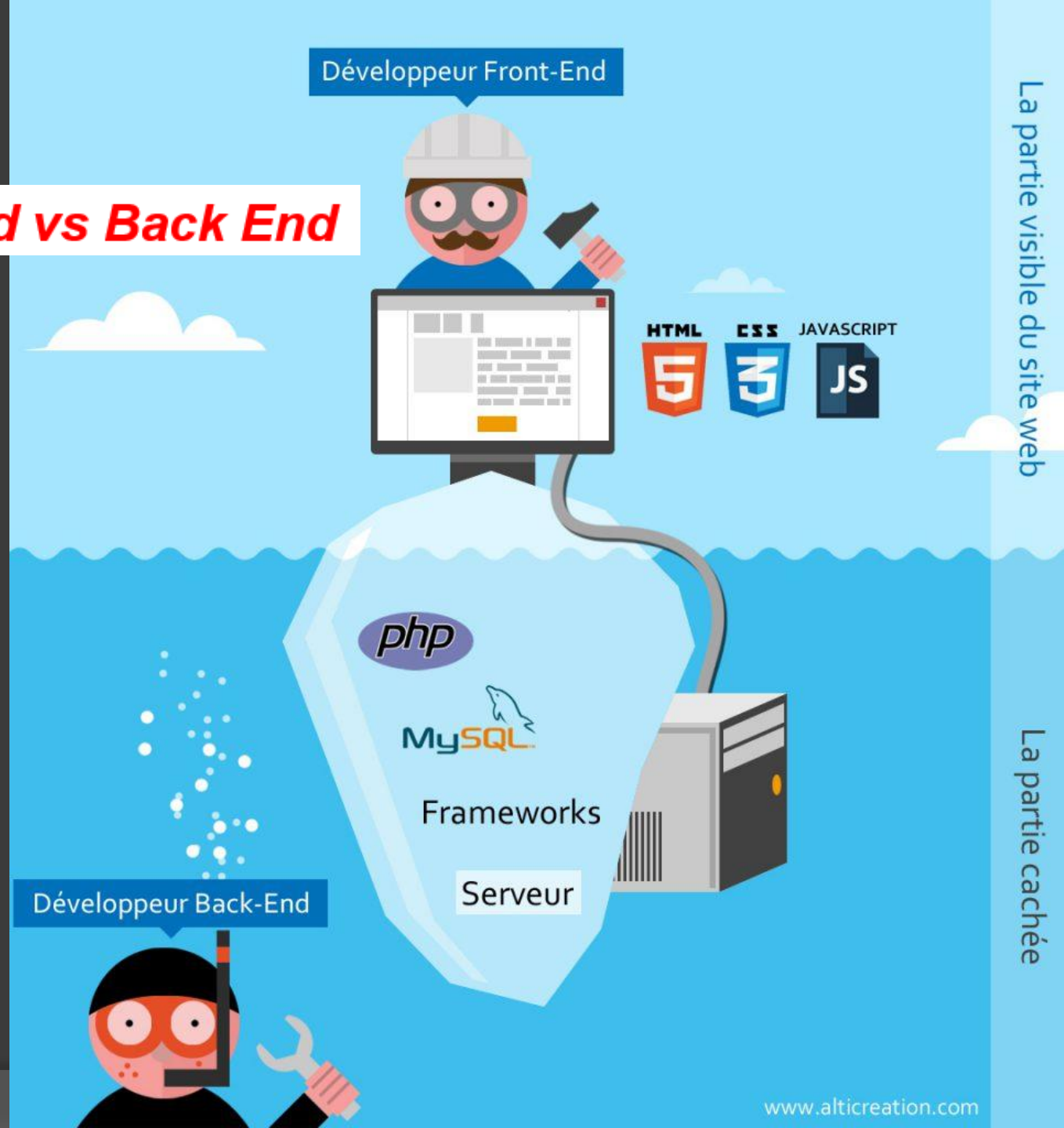
- ⦿ Présentation de JS
- ⦿ Interaction avec PHP
- ⦿ Tour d'horizon
- ⦿ Le DOM

I. Présentation de Javascript

1. Comment fonctionne un site web



2. *Front End vs Back End*










3. *Qu'est-ce que Javascript ?*

- C'est un langage de programmation comme le langage C ou Python orienté objet et il peut être utilisé en Back-End ou en Front-End.
- Dans ce cours, nous parlerons uniquement de Front-End.
- Pour la petite histoire, Javascript (JS pour les intimes) a été créé en seulement 10 jours par Brendan Eich afin de donner de l'interactivité aux pages web qui étaient jusqu'à présent uniquement statique.
- Une utilisation de JavaScript classique et de vérifier un formulaire dans votre page web -> plus efficace

👉 A retenir : Javascript est un langage utilisé pour les scripts côté client pour apporter un côté « dynamique » à la page. Il est interprété par le navigateur.

👉 A retenir : Javascript fait parti du trio HTML/CSS/Javascript indispensable pour la l'écriture de page Web dynamique et intégrative côté client.

Aug 2022	Aug 2021	Change	Programming Language		Ratings	Change
1	2	▲		Python	15.42%	+3
2	1	▼		C	14.59%	+2
3	3			Java	12.40%	+1
4	4			C++	10.17%	+2
5	5			C#	5.59%	+0
6	6			Visual Basic	4.99%	+0
7	7			JavaScript	2.33%	-0

4. *Avantages / inconvénients*

a) *Avantages*

- Ce langage est très populaire -> très facile à apprendre. La courbe d'apprentissage est très rapide.
- D'ailleurs beaucoup de jeunes codeurs ont débuté avec ce langage.
- JavaScript peut être utilisé côté serveur notamment avec node.js
- Il est dit que l'on peut tout faire avec JavaScript y compris des applications pour smartphone
- Un framework est un ensemble d'outils, de bibliothèques, qui facilite la vie du programmeur : AngularJS, React, Node.js, React Native, sont massivement utilisés en entreprise.

4. *Avantages / inconvénients*

b) Inconvénients

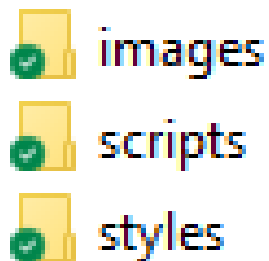
- C'est un langage de script qui est interprété, exécuté par le navigateur.
- Les langages interprétés (comme le Python) sont censés être moins rapides que les langages compilés (comme le langage C)
- Pour pouvoir utiliser toutes les possibilités de JS -> il vous faut connaître HTML et le CSS
- Son nom : cela porte très souvent en confusion JavaScript n'a rien à voir avec le langage Java.
- Souvent, certains recruteurs pensent que les programmeurs Javascript peuvent également développer en Java.
- Comme il est interprété par le navigateur donc problème d'affichage

5. Démarrer avec Javascript

- Je préfère le déclarer dans l'entête avec le css ainsi :

```
<head>
  <meta charset="utf-8">
  <title>Premiers Pas</title>
  <link rel="stylesheet" href="style.css">
  <script src="script.js"></script>
</head>
```

⚠ Attention, le fichier *script.js* (ainsi que *style.css*) doit être dans le même répertoire que votre fichier *html*.



```

<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Premiers Pas</title>
    <link rel="stylesheet" href="style.css">
    <script src="script.js"></script>
  </head>
  <body>
    <h1 id="msg">Hello World!
    <input type="text" value="Premiers pas avec Javascript"
    <br><br>
    <input type="radio" id="c1"> Choix N°1
    <input type="radio" id="c2"> Choix N°2
    <input type="radio" id="c3"> Choix N°3
    <br><br>
    <input type="password" value="....."
    <br><br>
    <input type="button" value="Ok" id="bp">
  </body>
</html>

```

Hello World!

☐ Choix N°1
 ☒ Choix N°2
 ☐ Choix N°3

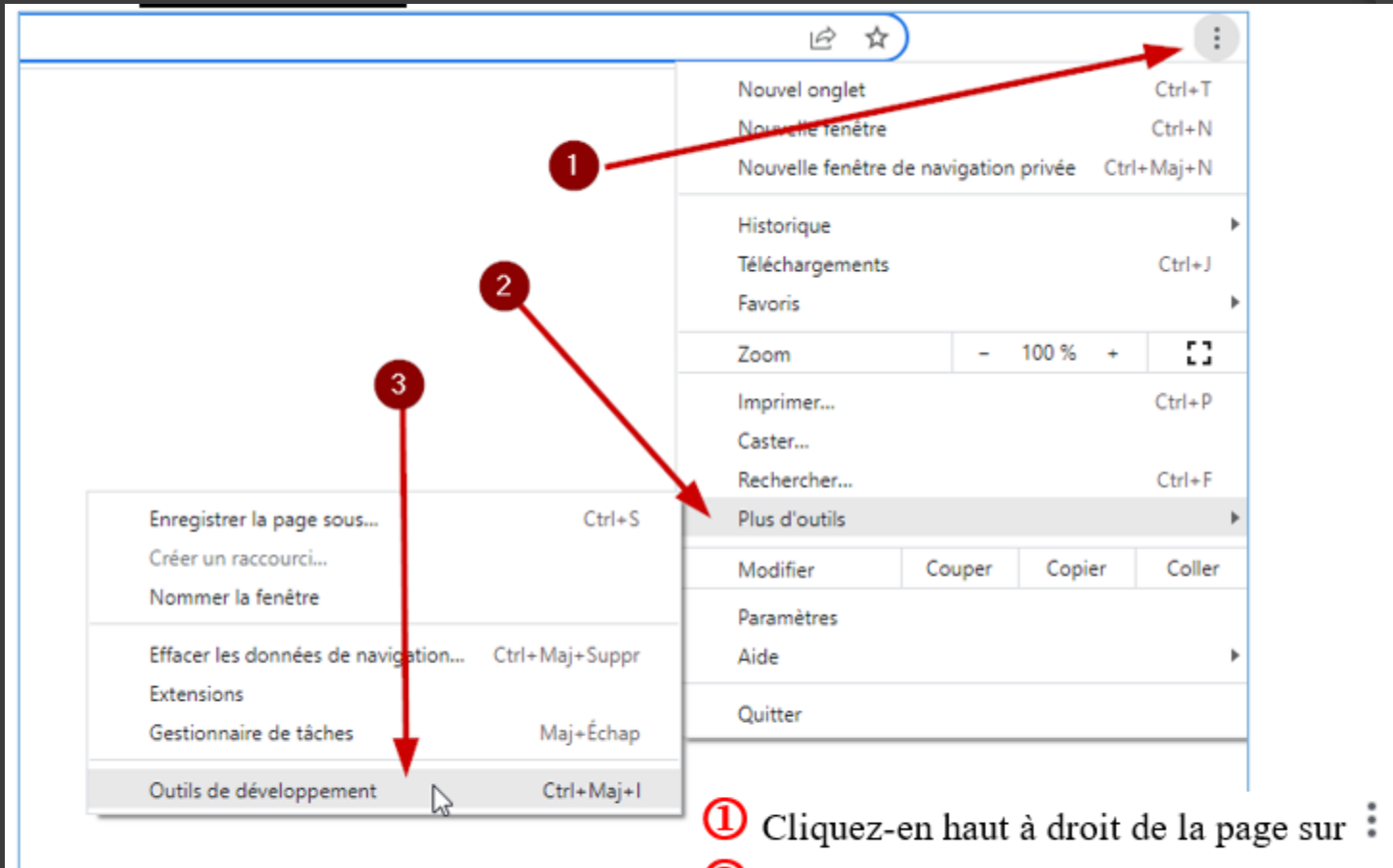

```
script" size="30" id="label">
```

```
:1"> Choix N°1
```

```
:d id="c2"> Choix N°2
```

6. Accès aux outils de développement

a) Sous Chrome

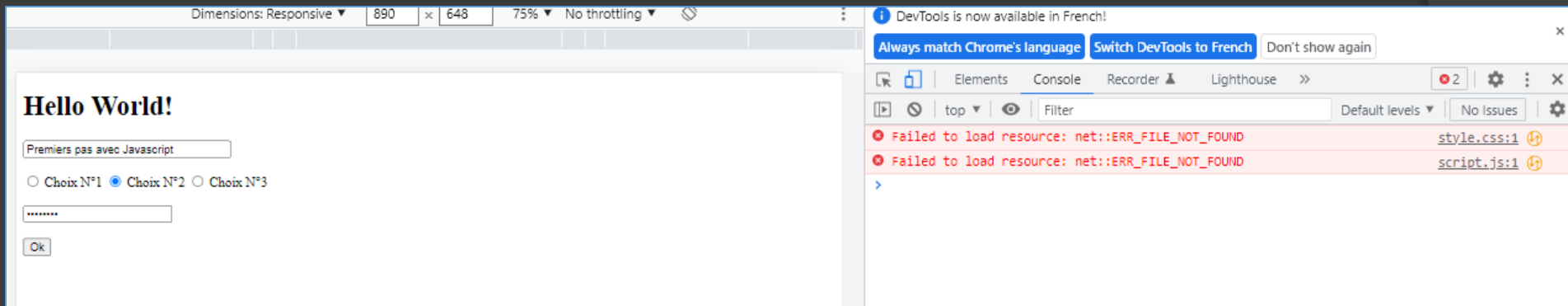


The screenshot shows the Chrome browser interface with the menu open. Red arrows and numbers indicate the steps to access Developer Tools:

- ① Cliquez-en haut à droit de la page sur ⋮
- ② Choisissez *Plus d'outils*
- ③ Outils de développement

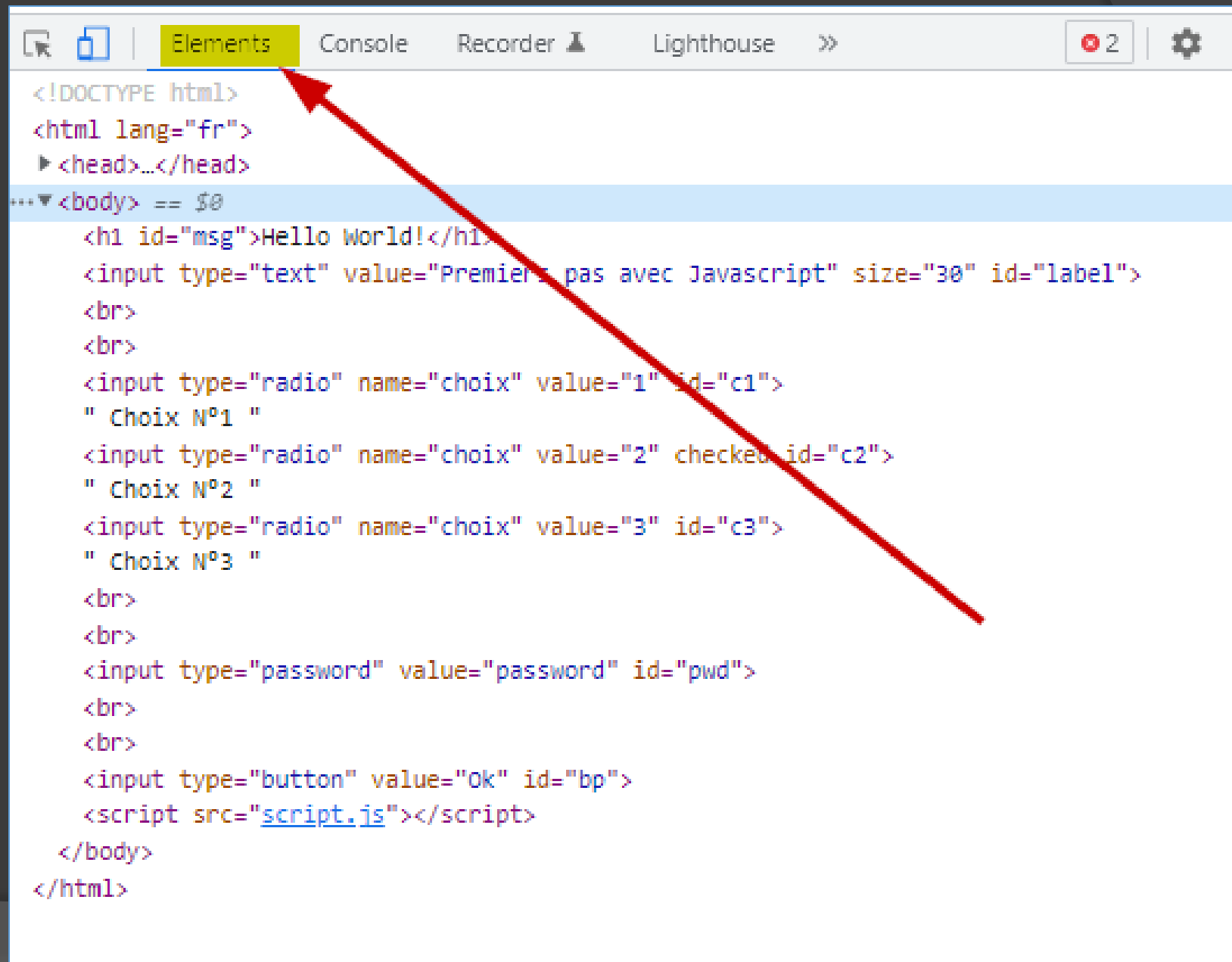
6. Accès aux outils de développement

a) Sous Chrome



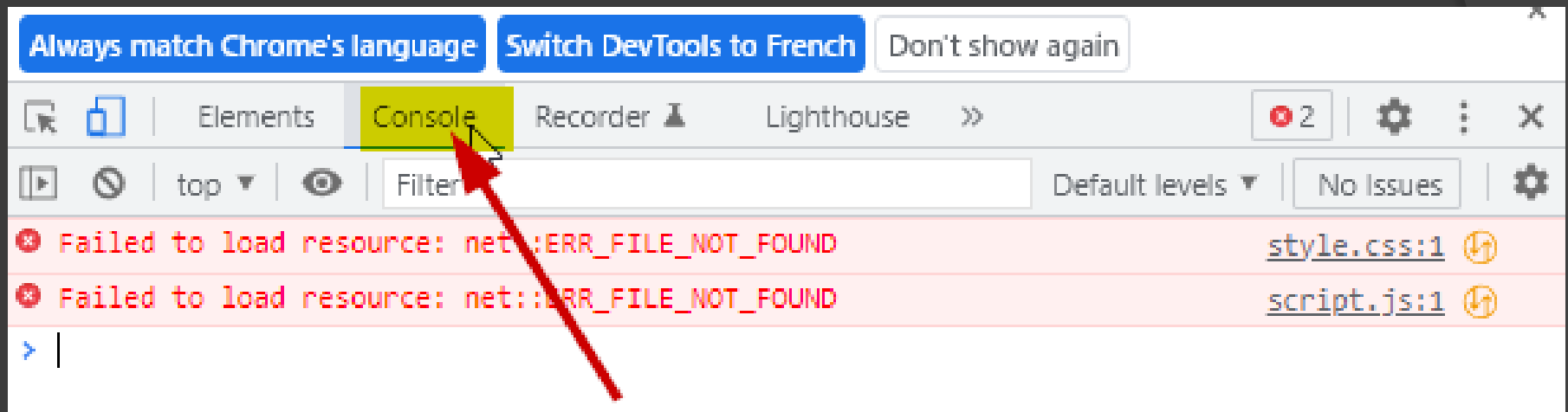
6. Accès aux outils de développement

a) Sous Chrome



6. Accès aux outils de développement

a) Sous Chrome



6. Accès aux outils de développement

b) Avec Firefox

Hello World!

Premiers pas avec Javascript

☐ Choix N°1 ☒ Choix N°2 ☐ Choix N°3

.....

Ok

Inspecteur Console Débugueur Éditeur de style Performances Mémoire Réseau Stockage Adblock Plus Accessibilité Applications

Rechercher dans le HTML

Filtrer les styles

élément { font-size: 15px; }

inline

Mise en page Calculé Modifications Compatil

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <title>Premiers pas avec Javascript</title>
  </head>
  <body style="font-size: 15px;">
    <h1 id="msg">Hello World!</h1>
    <input id="label" type="text" value="Premiers pas avec Javascript" size="30">
    <br>
    <br>
    <input id="c1" type="radio" name="choix" value="1">
    Choix N°1
    <input id="c2" type="radio" name="choix" value="2" checked="">
    Choix N°2
    <input id="c3" type="radio" name="choix" value="3">
    Choix N°3
    <br>
    <br>
    <input id="pwd" type="password" value="password">
    <br>
    <br>
    <input id="bp" type="button" value="Ok">
    <script src="script.js"></script>
    <script src="moz-extension://f0760e31-df9a-46fd-8300-51a7eb430c0e/js/app.js" type="text/javascript"></script>
    <div id="grammalecte_menu_main_button_shadow_host" style="width: 0px; height: 0px;"></div>
  </body>
  <script src="moz-extension://e96fcd8-0a54-43c9-984e-37b518e8abaa/content_scripts/api.js"></script>
</html>
```

Polices utilisées

Tahoma

Times New Roman

Times New Roman Bold, Times New Roman

Taille 15 px

Hauteur de... normal

Espacement normal

Graisse 400

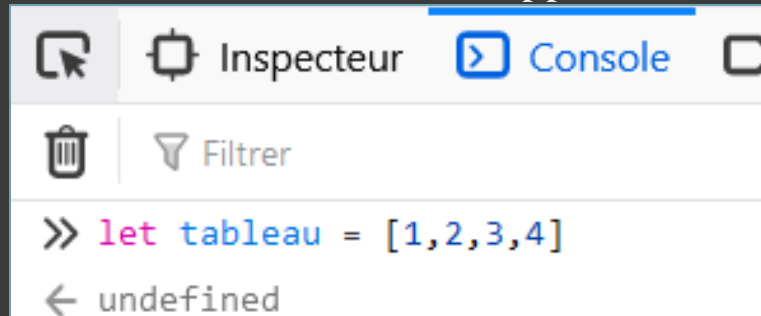
Italique

Toutes les polices sur la page

7. Mode console

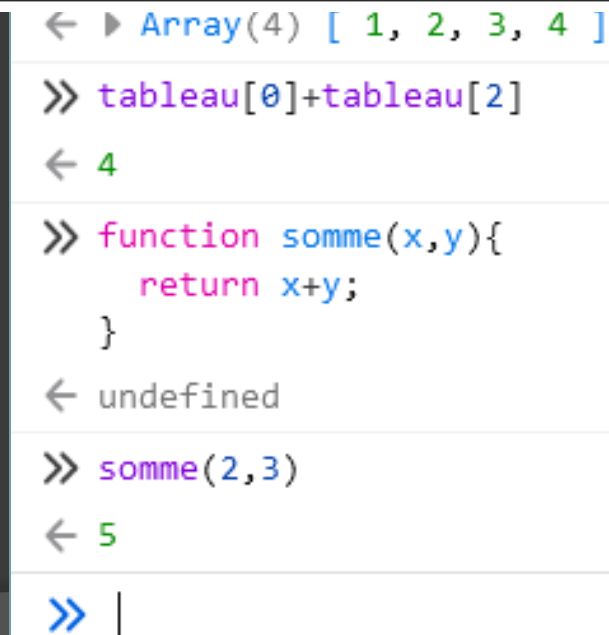
Le mode console est un mode interactif : vous entrez une commande elle s'exécute directement.

Pour cela, mettez-vous en mode *Outils de développement* :



```
>> let tableau = [1,2,3,4]  
← undefined
```

⚡ Dans ce mode il n'est pas obligatoire de mettre le point virgule à la fin de la commande. Ce n'est pas le cas lorsque le code se trouve dans le fichier !!!

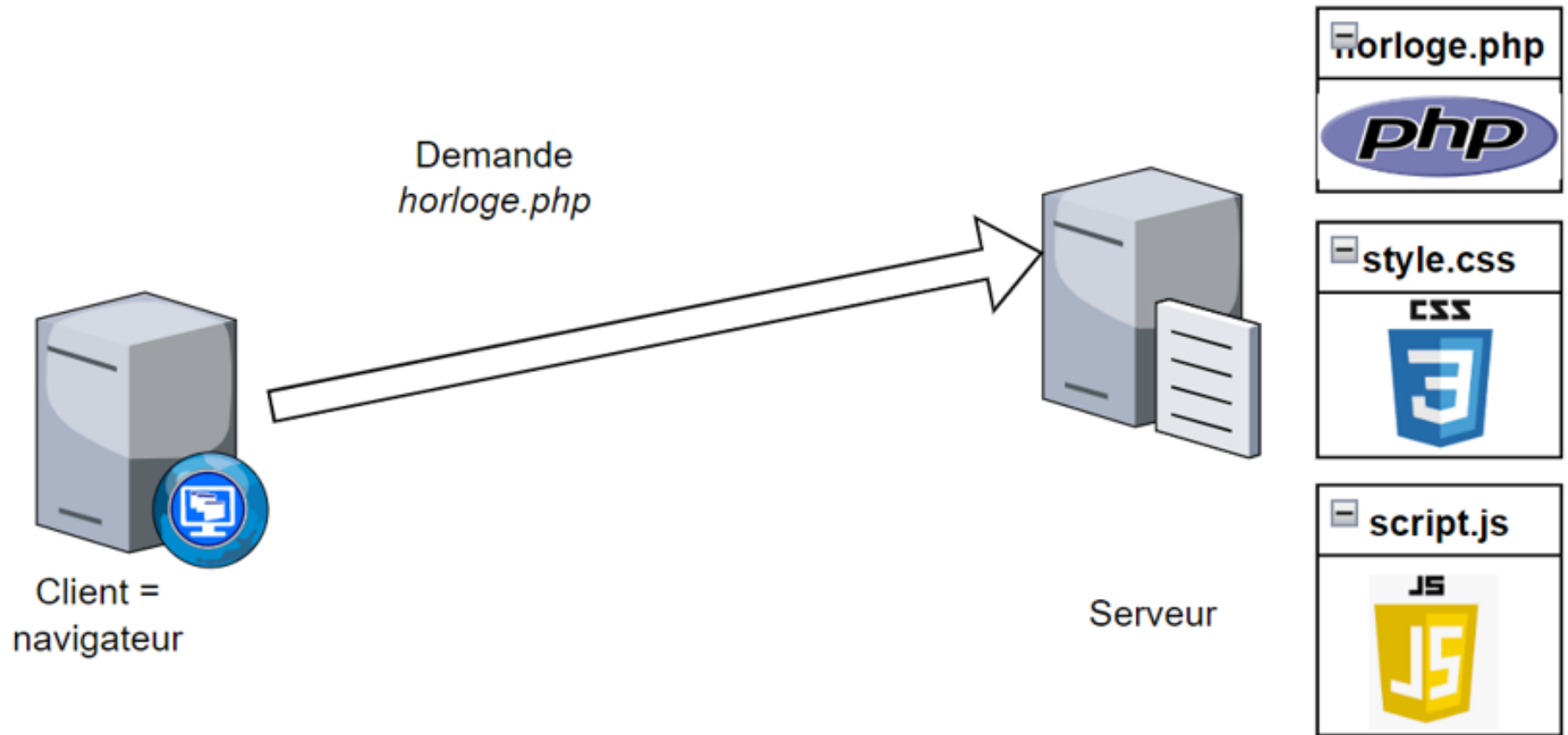


```
← ▶ Array(4) [ 1, 2, 3, 4 ]  
>> tableau[0]+tableau[2]  
← 4  
>> function somme(x,y){  
    return x+y;  
}  
← undefined  
>> somme(2,3)  
← 5  
>> |
```

IV. Exemple d'interaction entre PHP et Javascript

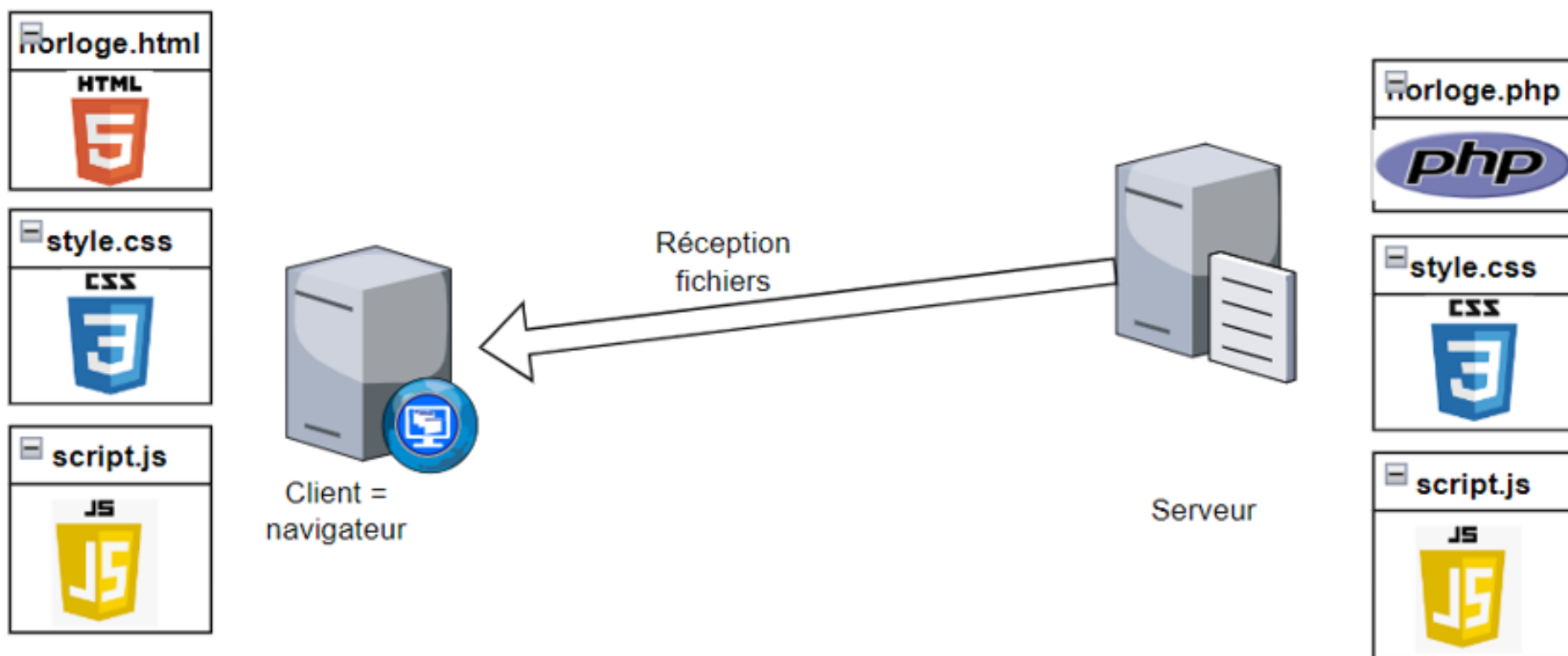
3. Avant la demande du fichier horloge.php/

Tous les fichiers se trouvent sur le serveur web :



4. Après la demande

Le client reçoit les fichiers qu'ils gardent en cache, mais pas le *PHP* qui lui est « transformé » par le serveur en fichier *html* à partir du fichier *PHP*.



II. Tour d'horizon du langage Javascript

En résumé : variables

Types de variables	
Déclaration	<code>let unEntier = 3;</code>
Affectation	<code>somme = 1.2 + 1.3;</code>
Chaînes	<code>let nom = "toto";</code> ou <code>let nom2 = 'tata'</code>
Booléens	<code>true</code> ou <code>false</code>
Opérateurs ou et non	<code> </code> && <code>!</code>
Tableaux	<code>t = [1,2,3,4]</code> et <code>t[1]</code> ou <code>t.lenght</code>
Dictionnaires	<code>dico = {a:1, b:2, c:3}</code> d.a

☛ Notez le point virgule à la fin de la ligne.

Éléments du langage

Commentaire

```
// Ceci est un commentaire
```

Conditionnelle : si
alors sinon

```
if (condition)
{
    instructions
}
else
{
    instructions
}
```

Boucle pour

```
if (condition)
{
    instructions
}
else
{
    instructions
}
```

Éléments du langage

Boucle tant que

```
while (condition)
{
    instructions
}
```

Fonction avec retour

```
function Mafonction(paramètres)
{
    instructions
    return qqchose
}
```

👉 L'opérateur permettant la division entière (`//` en Python) n'existe pas en Javascript

1. Les variables

La déclaration se fait par `let` :

```
let x; // Déclaration d'une variable x.
```

☛ il est possible de déclarer une variable comme ceci : `var x`; Dans ce cas cette variable est globale, cad qu'elle est visible dans tous les fichiers. Ce comportement est à éviter.

Voici un exemple d'affectation et de type des variable :

```
x = 0; // Maintenant la variable vaut zéro  
x // retourne sa valeur cad 0
```

JavaScript supporte plusieurs types de variables :

```
x = 1; // des entiers.  
x = 0.01; // réels.  
x = "hello world"; // chaîne de caractères entre des guillemets doubles  
x = 'JavaScript'; // ou des simples  
x = true; // valeur booléenne vrai  
x = false; // ou faux  
x = null; // null est une valeur spéciale qui signifie "pas de valeur"  
x = undefined; // Presque la même chose, non définie
```

QCM Variables

○ Laquelle de ces variables sera déclarée correctement ?

```
var var = 4;  
text = 'Hello !';  
var variable = 5.781e+8;  
var 1variable = 10;
```

☐ var

☐ text

☐ variable

☐ 1variable

○ Par quoi est encadrée une chaîne de caractères ?

☐ Par des guillemets : " "

☐ Par des chevrons : < >

☐ Par des apostrophes : ' '

☐ Il est possible d'utiliser les trois ci-dessus.

☐ Par des apostrophes ou des guillemets, cela a peu d'importance.

○ Je viens de réaliser une concaténation, est-elle correcte : `var text = "J'aime " - 'le JavaScript !';` ?

☐ Oui

☐ Non, il faut utiliser le signe + au lieu du - !

☐ Non, car on ne peut pas faire une concaténation dès la déclaration d'une variable.

☐ Non, il faut utiliser les apostrophes sur les deux chaînes de caractères.

○ Remplissez les champs laissés blancs

Complétez le code suivant pour réaliser une concaténation correcte :

```
var mois = "octobre",  
    annee = "2015";
```

```
var str = mois  " " + annee; // octobre 2015
```

○ Est-il possible de raccourcir la troisième ligne de ce code ?

```
var number1 = 60, number2 = 2;
```

```
number1 = number2 + 40;
```

☐ Oui, il suffit d'utiliser l'opérateur +=

☐ Non

○ Quel est le résultat de ce code ?

```
var number1 = "2", number2 = "3", resultat;  
resultat = number1 + number2;  
alert(resultat);
```

☐ -1

☐ 5

☐ 23

☐ Rien, le script rencontre une erreur

2. Les tableaux - listes

```
let prix = [212.5, 301, 125, 250]; // un tableau de valeurs délimités par []
prix[0] // retourne 212.5 cad le premier élément à l'indice 0
prix[3] // retourne 250 le dernier élément à l'indice 3
prix[prix.length-1] // Même résultat
prix.length // retourne 4 car le nombre d'éléments est 4.
prix[4] = 119; // Ajoute un nouvel élément.
prix // retourne Array(4) [ 212.5, 301, 125, 250, 119 ]
prix[4] = 0; // modification d'un éléments du tableau.
let tableauVide = []; // [] indique un tableau vide
tableauVide.length; // retourne 0
```

Remarque : un tableau peut contenir différent types de données

```
let tab = [1, "e"]
```

3. *Des dictionnaires (appelés JavaScript Objects)*

Les dictionnaires sont une sorte de collection d'Object

L'object est une collection de paires nom/valeur comme en Python qui se trouvent entre crochets {}

```
let ang2fr ={"one":"un"} ;  
ang2fr["two"] = "deux"; // Ajout d'une paire nom/valeur  
ang2fr; // Renvoie Object { one: "un", two: "deux" }  
ang2fr.one; // Renvoie "un"  
ang2fr.content = {}; // Le contenu est vidé
```

4. *Listes et dictionnaires*

a) Liste de dictionnaire

Il est possible d'inclure des dictionnaires dans une liste :

```
let pointsXY = [ // un tableau à deux dimensions
  {x:1, y:0}, // l'élément est un objet
  {x:10, y:10}
];
pointsXY[1] ;// Retourne Object { x: 10, y: 10 }
```

b) Des dictionnaires de tableaux

```
let vecteurs = {
  vect1:[ [1,2] , [3,4] ],
  vect2:[ [3,4] , [5,4]]
};
```

```
vecteurs["vect1"] // Renvoie Array [ [1,2] , [3,4] ]
vecteurs["vect"] = [ [11,12], [21,22]]; // Modifie le deuxième
```

5. Des calculs

```
// Calculs
```

```
3 + 2 ;// retourne 5: addition
```

```
3 - 2 ;// retourne 1: soustraction
```

```
3 * 2 ;// retourne 6: multiplication
```

```
3 / 2 ;// retourne 1.5: division
```

```
"3" + "2" ;// retourne Attention le résultat est "32"
```

```
// Les opérateurs d'incrémentation et de décrémentation
```

```
let compteur = 0; // Défini un compteur
```

```
compteur++; // Incrémente le compteur
```

```
compteur--; // Décrémente le compteur
```

```
compteur += 2; // ajoute 2 à la variable : idem compteur = compteur + 2;
```

```
compteur *= 3; // Multiplie par 3: idem compteur = compteur * 3;
```

```
compteur // retourne 6
```

QCM Tableaux ...

○ Si je veux accéder au troisième item d'un tableau, quel indice dois-je utiliser ?

☐ 4

☐ 3

☐ 2

○ Quelle est la déclaration syntaxiquement correcte ?

☐ `var a = { maj: 'A': min: 'a' };`

☐ `var b = { maj: 'B'; min: 'b' };`

☐ `var c = { maj: 'C', min: 'c' };`

6. Conditions booléennes

```
// Condition booléenne
let x = 1, y = 2;
x === y ;// retourne false faux: égalité stricte
x !== y ;// retourne true vrai: inégalité
x < y ;// retourne true vrai
x <= y ;// retourne true vrai
x > y ;// retourne false faux
x >= y ;// retourne false faux
"un" === "deux" ;// retourne false faux
"un" > "deux" ;// retourne false faux car "un" est alphabétiquement plus grand que "de"
```

☛ L'égalité faible (==) effectuera une conversion des deux éléments à comparer avant d'effectuer la comparaison
L'égalité stricte (===) effectuera la même comparaison mais sans conversion préalable (elle renverra toujours false si les types des deux valeurs comparées sont différents)

```
let num = 0, str="0";
num == str; // Retourne true Vrai !!!
num === str; // Retourne false Faux ouf
```

7. Définition et utilisation de fonctions

```
// Les fonctions  
function carre(x){ // La définition d'une fonction se fait par fonction  
    return x*x; // on retourne le carré de x : x*x  
}
```

L'appel se fera tout simplement comme ceci :

```
carre(4); // Appelle la fonction carre avec comme paramètre 4. Cette fonction retourne 16
```

QCM Fonction

○ À quoi servent les fonctions ?

- ☐ À rien
- ☐ À se passer des boucles
- ☐ À n'écrire qu'une seule fois un même code pour ensuite l'appeler où on le souhaite
- ☐ À exécuter un code provenant d'un autre site Web

o La déclaration de ma fonction est-elle correcte ?

```
function MyFunction(arg1 arg2) {  
    // Mon code.  
}
```

- ☐ Oui
- ☐ Non, il manque un point-virgule dans la déclaration des arguments
- ☐ Non, il manque une virgule dans la déclaration des arguments

o Quelle est la différence entre une variable globale et une variable locale ?

- ☐ La locale est accessible partout dans le code tandis que la globale est limitée à la fonction où elle est déclarée
- ☐ La globale est accessible partout dans le code tandis que la locale est limitée à la fonction où elle est déclarée
- ☐ Les globales et locales sont identiques tant qu'elles ne sont pas déclarées dans une boucle ou une condition

○ Le code suivant fonctionne-t-il ?

```
function test() {  
  if (true) { var a = "hello"; }  
  alert(a);  
}
```

- ☐ Non, erreur de syntaxe
- ☐ Non, `alert(a)` n'affiche rien
- ☐ Oui, mais c'est vraiment moche
- ☐ Oui, et c'est une bonne idée

8. *If alors sinon :*

```
// Conditionnel
function abs(x){
    if (x<0){
        return -x;
    }
    else {
        return x;
    }
}
```

Puis les appels pourront être :

```
abs(3); // Retourne 3
abs(-3); // Retourne 3
```

☼ Vous avez remarqué l'indentation (tabulation) et les crochets {} qui entourent les instructions. Les tabulations ne sont pas obligatoires mais vivement conseillées par une meilleure lisibilité. Par contre les crochets le sont !!!

6. Conditions booléennes

```
// Condition booléenne
let x = 1, y = 2;
x === y ;// retourne false faux: égalité stricte
x !== y ;// retourne true vrai: inégalité
x < y ;// retourne true vrai
x <= y ;// retourne true vrai
x > y ;// retourne false faux
x >= y ;// retourne false faux
"un" === "deux" ;// retourne false faux
"un" > "deux" ;// retourne false faux car "un" est alphabétiquement plus grand que "de"
```

☛ L'égalité faible (==) effectuera une conversion des deux éléments à comparer avant d'effectuer la comparaison
L'égalité stricte (===) effectuera la même comparaison mais sans conversion préalable (elle renverra toujours false si les types des deux valeurs comparées sont différents)

```
let num = 0, str="0";
num == str; // Retourne true Vrai !!!
num === str; // Retourne false Faux ouf
```

QCM conditionnels

o Que va être le résultat de cette condition ? `var result = 8 % 2 > 0 || !(3 % 2 < 1);`

☐ true

☐ false

☐ Ce code renvoie une erreur

☐ 42

```
// Condition initiale :  
var result = 8 % 2 > 0 || !(3 % 2 < 1);  
  
// Condition décomposée en trois parties :  
var result1 = 8 % 2 > 0,  
result2 = !(3 % 2 < 1),  
result3 = result1 || result2;
```

La condition ainsi décomposée est déjà plus facile à cerner :

- Pour `result1` on fait le calcul $8 \% 2 = 0$ et on obtient ainsi la comparaison $0 > 0$ qui renvoie donc `false`.
- Pour `result2` on fait le calcul $3 \% 2 = 1$ et on obtient ainsi la comparaison $1 > 1$ qui renvoie donc `false`. En revanche là on constate que la condition est entourée de parenthèses et est précédée de l'opérateur NON, il nous faut donc inverser le résultat de la condition, ainsi, `false` devient `true` !
- Et pour terminer, dans `result3`, nous utilisons l'opérateur logique OU qui renvoie `true` si l'une des valeurs soumises vaut `true` elle aussi. Ainsi, `false || true = true` !

Au final, notre condition renvoie `true` !

○ Dans quel ordre doit-on voir apparaître ces structures ?

☐ if > else > else if

☐ else if > if > else

☐ if > else if > else

☐ else > else if > if

○ Que va-t-il se passer si je clique sur le bouton « OK » dans la fenêtre de confirmation ?

```
if (!confirm('OK ?')) {  
    alert("C'est OK !");  
}
```

- ☐ Le message « C'est OK ! » va s'afficher.
- ☐ La page d'accueil de mon navigateur va s'afficher.
- ☐ Rien

9. *Une boucle avec pour*

Une variable de boucle, ici nommée `i` est déclarée puis utilisée avec `for` :

```
// Boucle pour
let somme = 0,i;
for (i=0;i<=3;i++) {
    somme = somme + i;
}
somme; // Renvoie 6 car 0+1+2+3 = 6
```

10. Boucle tant que

Ce code effectue le même calcul que précédemment :

```
// Boucle tant que
```

```
i = 0;
```

```
somme = 0;
```

```
while (i<=3){
```

```
    somme += i;
```

```
    i++;
```

```
}
```

```
somme; // Renvoie 6 car 0+1+2+3 = 6
```

11. For avec un tableau

```
// pour avec un tableau
let prix = [10,100,1000];

function somme(tableau) { // Calcule la somme des éléments de tableau
  let total = 0; // initialisation à zéro.
  // Boucle sur tableau et récupère chaque élément dans unPrix
  for(let unPrix of tableau) {
    total += unPrix; // Ajoute l'élément au total
  } // Fin de la boucle

  somme(prix) // Renvoie 10+100+1000 = 1110
```

CQM Boucles

○ Quelle est la valeur d'`output` dans l'instruction suivante : `var output = count++;` ?

- ☐ `count`, incrémentée de 1
- ☐ `count`, sans incrémentation
- ☐ Juste l'incrément

Si l'opérateur `++` se trouve après la variable, la valeur de la variable est retournée, et l'incrément se fait après. Ici, `output` contient donc la valeur de `count`, avant l'incrément.

○ Quelle est l'utilité de **break** dans une boucle ?

☐ Il permet de faire une pause

☐ Il permet d'arrêter l'itération en cours et de passer à la suivante

☐ Il permet d'arrêter l'itération en cours et de quitter la boucle

o Quelle est la particularité d'une boucle `do while` ?

☐ Aucune, c'est la forme longue de la boucle `while`

☐ Les instructions sont exécutées au moins une fois

☐ La condition n'est exécutée qu'au début de chaque itération

○ Quelle est la syntaxe exacte d'une boucle **for** ?

☐ `for (initialisation; condition; incrémentation) { }`

☐ `for (condition; initialisation; conclusion) { }`

☐ `for (initialisation; incrémentation; condition) { }`

☐ `for (initialisation; condition) { }`

○ Dans une boucle **for**, à quel moment le bloc d'*incrément*ation est-il exécuté ?

☐ Au début de chaque itération

☐ Pendant chaque itération

☐ À la fin de chaque itération