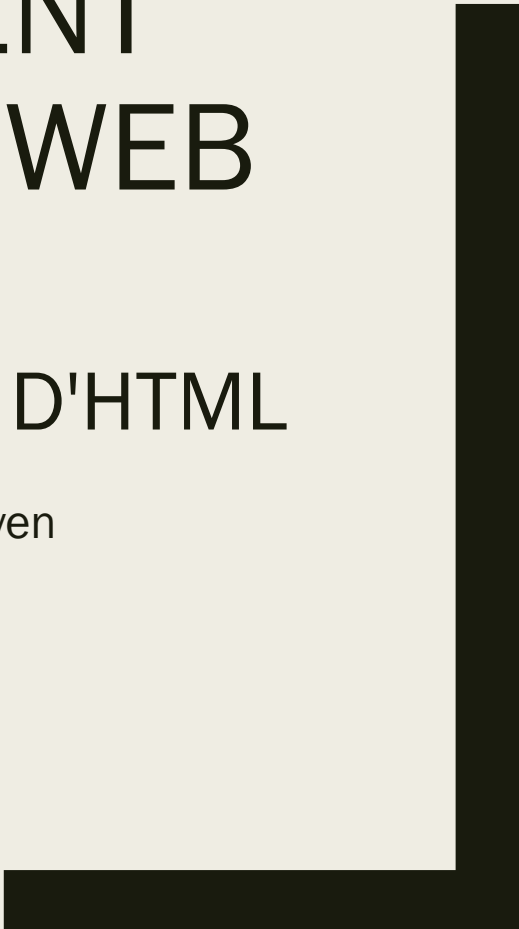




DÉVELOPPEMENT D'APPLICATIONS WEB

COURS 5: INTRODUCTION À JAVASCRIPT

Enseignante: NGUYEN Thi Minh Tuyen



Plan du cours

1. Qu'est-ce que JavaScript?
2. Syntaxe de base
 - a. *Variables*
 - b. *Types de données de valeur*
 - c. *Mathématiques de base en JavaScript*
 - d. *Gérer du texte – chaîne de caractères*
 - e. *Tableaux*
 - f. *Blocs*

Plan du cours

1. Qu'est-ce que JavaScript?
2. Syntaxe de base
 - a. *Variables*
 - b. *Types de données de valeur*
 - c. *Mathématiques de base en JavaScript*
 - d. *Gérer du texte – chaîne de caractères*
 - e. *Tableaux*
 - f. *Blocs*

Qu'est-ce que JavaScript? [1]

- Abrégé en « JS »
- Un langage de script côté client: il est traité sur la machine de l'utilisateur (et pas sur le serveur).
- Dépend des capacités du navigateur (il peut même être totalement indisponible).
- Un langage de programmation dynamique - il n'est pas nécessaire de le compiler dans un programme exécutable.
- faiblement typé (loosely typed) – pas besoin de définir des types de variable comme nous le faisons pour d'autres langages de programmation.

Qu'est-ce que JavaScript? [2]

- Créé en 1995 par Brendan Eich (cofondateur de Mozilla)
- JavaScript n'a rien à voir avec Java
- Littéralement nommé ainsi pour des raisons de marketing
- La première version a été écrite en 10 jours
- Plusieurs décisions linguistiques fondamentales ont été prises à cause de la politique de l'entreprise et non de raisons techniques

Une définition générale

- JavaScript est l'une des 3 langues que tous les développeurs web doivent apprendre:
 1. **HTML**: un langage de balisage utilisé pour structurer et donner sens au contenu web
 2. **CSS**: un langage de règles de style utilisé pour appliquer un style au contenu HTML.
 3. **JavaScript**: programmer le comportement des pages web



Tâches JavaScript

JavaScript ajoute une couche comportementale (interactivité) à une page Web. Quelques exemples incluent:

- Vérifier les soumissions de formulaire et fournir des messages de feedback et des modifications de l'interface utilisateur
- Injecter du contenu dans les documents actuels à la volée
- Afficher et cacher de contenu lorsque l'utilisateur clique sur un lien ou un en-tête
- Remplir un terme dans une boîte de recherche
- Tester des fonctionnalités du navigateur
- etc.

Ajout de scripts à une page [1]

Script interne

- Incluez l'élément `script` juste avant la balise fermante `</head>`:

```
<script>
```

... JavaScript code goes here

```
</script>
```

Script externe

- Utilisez l'attribut **src** dans l'élément `script` pour désigner un fichier `.js` externe et autonome:

```
<script src="my_script.js"></script>
```


Ajout de scripts à une page [2]

Inline JavaScript handlers

- Nous pouvons mettre des morceaux de JavaScript directement dans le HTML.

Placement de script

- L'élément de `script` peut aller n'importe où dans le document, mais les endroits les plus courants sont les suivants:

En tête du document

Pour quand vous voulez que le script fasse quelque chose avant que le corps ne soit complètement chargé (ex: Modernizr):

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <script
      src="script.js"></script>
  </head>
```

...

Juste avant la balise `</ body>`

Préféré lorsque le navigateur doit analyser le document et sa structure DOM avant d'exécuter le script:

```
...
<body>
  <!--contents of page-->
  <script
    src="script.js"></script>
</body>
</html>
```

Premier exemple

```
<p>Player: Chris</p>
```

HTML

```
p {  
  font-family: 'helvetica neue', helvetica, sans-serif;  
  letter-spacing: 1px;  
  text-transform: uppercase; text-align: center;  
  border: 2px solid rgba(0,0,200,0.6);  
  background: rgba(0,0,200,0.3);  
  color: rgba(0,0,200,0.6);  
  box-shadow: 1px 1px 2px rgba(0,0,200,0.4);  
  border-radius: 10px;  
  padding: 3px 10px;  
  display: inline-block;  
  cursor:pointer;  
}
```

CSS

```
const para = document.querySelector('p');  
para.addEventListener('click', updateName);  
function updateName() {  
  let name = prompt('Enter a new name');  
  para.textContent = 'Player 1: ' + name;  
}
```

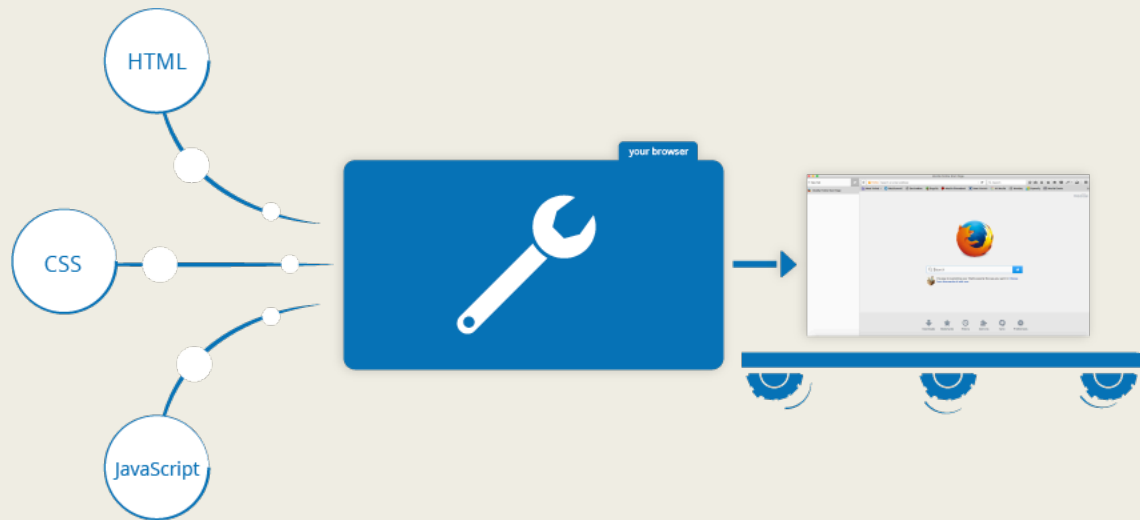
Javascript

Que peut-il vraiment faire ?

Le cœur de JavaScript est constitué de fonctionnalités communes de programmation permettant de :

- stocker des valeurs utiles dans des variables.
- faire des opérations sur des chaînes de caractères.
- exécuter du code en réponse à certains évènements se produisant sur une page web.
- etc.

Que fait JavaScript sur votre page ?



Stratégies de chargement de script [1]

- Le chargement des scripts au bon moment pose un certain nombre de problèmes.
 - Un problème courant est que tout le code HTML d'une page est chargé dans l'ordre dans lequel il apparaît.
 - Si vous utilisez JavaScript pour manipuler des éléments sur la page, votre code ne fonctionnera pas si le JavaScript est chargé et analysé avant le code HTML auquel vous tentez de faire quelque chose.
- Solution: Deux façons de contourner le problème du script de blocage: **async** et **defer**.

Stratégies de chargement de script [2]

- Les scripts asynchrones téléchargeront le script sans bloquer le rendu de la page et l'exécuteront dès que le téléchargement du script sera terminé.
- Vous n'obtenez aucune garantie que les scripts s'exécutent dans un ordre spécifique, mais seulement qu'ils n'empêcheront pas le reste de la page de s'afficher.
- Il est préférable d'utiliser **async** lorsque les scripts de la page s'exécutent indépendamment les uns des autres et ne dépendent d'aucun autre script de la page.
- Exemple:

```
<script async src="js/vendor/jquery.js"></script>  
<script async src="js/script2.js"></script>  
<script async src="js/script3.js"></script>
```

Stratégies de chargement de script [3]

- defer exécute les scripts dans l'ordre dans lequel ils apparaissent dans la page et les exécute dès que le script et le contenu sont téléchargés.

```
<script defer src="js/vendor/jquery.js"></script>  
<script defer src="js/script2.js"></script>  
<script defer src="js/script3.js"></script>
```

- Pour résumé :
 - Si vos scripts n'ont pas besoin d'attendre l'analyse et peuvent s'exécuter indépendamment sans dépendances, utilisez `async`.
 - Si vos scripts doivent attendre l'analyse et dépendent d'autres scripts, chargez-les à l'aide de `defer` et placez leurs éléments `<script>` correspondants dans l'ordre dans lequel vous souhaitez que le navigateur les exécute.

Plan du cours

1. Qu'est-ce que JavaScript?
2. Syntaxe de base
 - a. *Variables*
 - b. *Types des variables*
 - c. *Mathématiques de base en JavaScript*
 - d. *Gérer du texte – chaîne de caractères*
 - e. *Tableaux*
 - f. *Blocs*

Plan du cours

1. Qu'est-ce que JavaScript?
2. Syntaxe de base
 - a. *Variables*
 - b. *Types de données de valeur*
 - c. *Mathématiques de base en JavaScript*
 - d. *Tableaux*
 - e. *Blocs*

Commentaires

Les commentaires sont très utiles, et vous devriez les utiliser fréquemment, surtout pour des applications de grande taille.

Deux types :

- Un commentaire sur une ligne s'écrit après un double slash, par exemple :

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

- Un commentaire sur plusieurs lignes s'écrit entre deux balises /* et */ :

```
/*
```

```
    Ceci est un commentaire  
    sur deux lignes
```

```
*/
```

Syntaxe de base de JavaScript

- JavaScript est sensible à la casse.
- Les espaces sont ignorés (sauf s'ils sont mis entre guillemets dans une chaîne de texte).
- Un script est composé d'une série d'instructions et de commandes qui indiquent au navigateur quoi faire.

Variables [1]

- Une variable est un conteneur pour une valeur.
- Une variable est composée d'un **nom** et d'une **valeur**.
- Une variable est utilisée afin de pouvoir faire référence à la valeur par son nom ultérieurement dans le script.
- La valeur peut être un nombre, une chaîne de texte, un élément du DOM ou une fonction, etc.
- Les variables sont définies à l'aide du mot clé **var**:

```
var foo = 5;
```

Variables [2]

Règles pour nommer une variable:

- Commencer par une lettre ou un sous-tiret _.
- Ne peut pas contenir d'espaces de caractères. Utilisez des sous-tiret _ ou “CamelCase” à la place.
- Ne doit pas contenir de caractères spéciaux (!., / \ + * =).
- Devrait décrire les informations qu’il contient.

Variables [3]

- Déclaration d'une variable

```
var myName;
```

```
var myAge;
```

- Initialisation d'une variable

```
myName = 'Chris';
```

```
myAge = 37;
```

- Il est possible de déclarer et initialiser une variable en même temps

```
var myName = 'Chris';
```

- Mise à jour d'une variable

```
myName = 'Bob';
```

```
myAge = 40;
```

Plan du cours

1. Qu'est-ce que JavaScript?
2. **Syntaxe de base**
 - a. *Variables*
 - b. ***Types de données de valeur***
 - c. *Mathématiques de base en JavaScript*
 - d. *Tableaux*
 - e. *Blocs*

Types de données de valeur [1]

Les valeurs attribuées aux variables relèvent de quelques types de données:

- **Undefined** La variable est déclarée en lui donnant un nom, mais pas de valeur:

```
var foo;
```

```
alert(foo); // Will open a dialog containing "undefined"
```

- **null** Attribue à la variable aucune valeur inhérente:

```
var foo = null;
```

```
alert(foo); // Will open a dialog containing "null"
```

- **Nombres** Lorsque vous attribuez un nombre , JavaScript le traite comme un nombre (vous n'avez pas besoin de lui dire que c'est un nombre):

```
var foo = 5;
```

```
alert(foo + foo); // This will alert "10"
```

Types de données de valeur [2]

- **Chaînes de caractère** Si la valeur est entourée de guillemets simples ou doubles, elle est traitée comme une chaîne de texte:

```
var foo = "five";  
alert(foo); // Will alert "five"  
alert(foo + foo); // Will alert "fivefive"
```

- **Booléens** Attribue une valeur `true` ou `false`, utilisée pour la logique de script:

```
var foo = true; // The variable "foo" is now true
```

- **Tableaux** Groupe de plusieurs valeurs (appelées membres) attribuées à une seule variable.
 - Les valeurs dans les tableaux sont indexées (à partir de 0).
 - Vous pouvez faire référence à des valeurs de tableau par leurs numéros d'index.

Types de données de valeur [3]

```
var foo = [5, "five", "5"];  
alert( foo[0] ); // Alerts "5"  
alert( foo[1] ); // Alerts "five"  
alert( foo[2] ); // Also alerts "5"
```

- **Objets** C'est une structure de code qui modélise un objet du réel.

```
var dog = { name : 'Spot', breed : 'Dalmatian' };
```

Pour récupérer une information stockée dans un objet, vous pouvez utiliser la syntaxe suivante :

```
dog.name
```

Types de données de valeur [4]

Typage faible

- JavaScript est un « langage faiblement typé »: vous n'êtes pas obligé de préciser quel est le type de donnée que doit contenir une variable (par ex. nombres, chaînes, tableaux, etc).
- Par exemple, si vous déclarez une variable et si vous y placez une valeur entre guillemets, le navigateur la traitera comme étant une chaîne

```
var myString = 'Hello';
```

```
var myNumber = '500'; // oops, c'est toujours une chaîne  
typeof(myNumber);
```

```
myNumber = 500; // mieux – maintenant c'est un nombre  
typeof(myNumber);
```

Exemple 2

```
<button>Press-moi</button>
```

```
var button = document.querySelector('button');  
  
    button.onclick = function() {  
        var name = prompt('Quel est votre nom?');  
        alert('Salut ' + name + ', sympa de vous voir!');  
    }
```

Plan du cours

1. Qu'est-ce que JavaScript?
2. Syntaxe de base
 - a. *Variables*
 - b. *Types de données de valeur*
 - c. ***Mathématiques de base en JavaScript***
 - d. *Gérer du texte – chaîne de caractères*
 - e. *Tableaux*
 - f. *Blocs*

Types de nombres

Entier (Integer)

- C'est un nombre sans partie fractionnaire, comme son nom l'indique, par exemple 10, 400 ou -5

Nombre à virgule flottante (float)

- Il a un point de séparation entre la partie entière et la partie fractionnaire (là où en France nous mettons une virgule), par exemple 12.5 et 56.7786543

Doubles (pour double précision)

- Ce sont des nombres à virgule flottante de précision supérieure aux précédents (on les dit plus précis en raison du plus grand nombre de décimales possibles).

Opérateurs arithmétiques

Opérateur	Nom	But	Exemple
+	Addition	Ajoute deux nombres.	$6 + 9$

Opérateurs de comparaison

Opérateur	Nom	But	Exemple
===	Égalité stricte	Teste si les valeurs de droite et de gauche sont identiques (même type de données)	5 === 2 + 4

Exemple

- Lorsque nous comparons deux valeurs, JavaScript évalue l'instruction et renvoie une valeur booléenne (`true` / `false`):

```
alert( 5 == 5 ); // This will alert "true"  
alert( 5 != 6 ); // This will alert "true"  
alert( 5 < 1 ); // This will alert "false"
```

- REMARQUE: Égal à (`==`) n'est pas identique à (`===`). Les valeurs identiques doivent partager un type de données:

```
alert( "5" == 5 ); /* This will alert "true". They're  
both "5".*/  
alert( "5" === 5 ); /* This will alert "false". They're  
both "5", but they're not the same data type.*/  
alert( "5" !== 5 ); /* This will alert "true", since  
they're not the same data type.*/
```

Plan du cours

1. Qu'est-ce que JavaScript?
2. **Syntaxe de base**
 - a. *Variables*
 - b. *Types de données de valeur*
 - c. *Mathématiques de base en JavaScript*
 - d. **Gérer du texte – chaîne de caractères**
 - e. *Tableaux*
 - f. *Blocs*

Chaînes de caractères [1]

- Créer une chaîne de texte

```
var string = 'La révolution ne sera pas télévisée.';
```

- Guillemets simples vs guillemets doubles: En JavaScript, vous pouvez envelopper vos chaînes entre des guillemets simples ou doubles. Les deux expressions suivantes sont correctes :

```
var sgl = 'Guillemet simple.';
```

```
var dbl = "Guillemets doubles.";
```

- Échappement de caractères dans une chaîne

```
var bigmouth = 'Je n\'ai pas eu droit à prendre place...';
```

Chaînes de caractères [2]

- Concaténation de chaînes

```
var one = 'Hello, ';  
var two = 'comment allez-vous?';  
var joined = one + two;
```

- Concaténation dans un contexte

```
var button = document.querySelector('button');  
  
button.onclick = function() {  
    var name = prompt('Quel est votre nom?');  
    alert('Hello ' + name + ', sympa de vous voir!');  
}
```

Nombres vs chaînes

- Que se passe-t-il quand vous faites l'additionner (ou concaténer) une chaîne et un nombre ?

```
'VP ' + 17;
```

→ "VP 17"

```
var myvar = '17' + '18'
```

```
typeof(myvar)
```

→ "string"

- Conversion: Chaînes de caractères → nombre

```
var myString = '123';
```

```
var myNum = Number(myString);
```

```
typeof myNum;
```

→ "number"

Nombres vs chaînes

- Conversion: nombre → Chaînes de caractères

```
var myNum = 123;
```

```
var myString = myNum.toString();
```

```
typeof myString;
```

```
→ "string"
```


Plan du cours

1. Qu'est-ce que JavaScript?
2. **Syntaxe de base**
 - a. *Variables*
 - b. *Types de données de valeur*
 - c. *Mathématiques de base en JavaScript*
 - d. *Gérer du texte – chaîne de caractères*
 - e. **Tableaux**
 - f. *Blocs*

Tableau [1]

- Décrit comme un « objet du type liste »
- Simplement: c'est un objet contenant un rangement de plusieurs valeurs en liste.
- Création d'un tableau

```
var shopping = ['pain', 'lait', 'fromage',  
                'houmous', 'nouilles'];
```

```
shopping;
```

```
→ ["pain", "lait", "fromage", "houmous", "nouilles"]
```

- Accès aux éléments de tableau et modification ses valeurs:

```
shopping[0];
```

```
→ "pain"
```

Tableau [2]

```
shopping[0] = 'crème de sésame';
```

```
shopping;
```

```
→["crème de sésame", "lait", "fromage", "houmous",  
  "nouilles"]
```

■ La taille d'un tableau:

```
shopping.length;
```

Conversions entre chaînes et tableaux [1]

Utiliser `split` et `join`

```
var myData =  
  'Manchester,London,Liverpool,Birmingham,Leeds,Carlisle';  
  
var myArray = myData.split(',');  
myArray;  
→ ["Manchester", "London", "Liverpool",  
  "Birmingham", "Leeds", "Carlisle"]  
  
var myNewString = myArray.join(',');  
myNewString;  
→ "Manchester,London,Liverpool,Birmingham,Leeds,Carlisle"
```

Conversions entre chaînes et tableaux [2]

```
var dogNames =  
["Rocket", "Flash", "Bella", "Slugger"];  
dogNames.toString();  
→ "Rocket,Flash,Bella,Slugger"
```

Ajout et suppression d'éléments de tableau

■ push et pop

```
var myArray = ['Manchester', 'London', 'Liverpool',  
'Birmingham', 'Leeds', 'Carlisle'];
```

```
myArray.push('Cardiff');
```

```
myArray;
```

```
myArray.push('Bradford', 'Brighton');
```

```
myArray;
```

```
myArray.pop();
```

■ unshift() et shift()

Déboguer du code JavaScript

Types d'erreurs en commun

- Erreurs de syntaxe
- Erreurs logiques

Réparer les erreurs de syntaxe

- Utiliser la console de votre navigateur

Plan du cours

1. Qu'est-ce que JavaScript?
2. Syntax de base
 - a. *Variables*
 - b. *Types de données de valeur*
 - c. *Mathématiques de base en JavaScript*
 - d. *Gérer du texte – chaîne de caractères*
 - e. *Tableaux*
 - f. **Blocs**

Principaux blocs en JS

- Conditions
- Boucles
- Fonctions
- Événements

Déclarations `if/else` [1]

- Une déclaration `if/else` teste les conditions en posant une question vraie / fausse.
- Si la condition entre parenthèses est remplie, exécutez les commandes entre accolades (`{}`):

```
if(true) {  
    // Do something.  
}
```

- Exemple:

```
if( 1 != 2 ) {  
    alert("These values are not equal.");  
    /* It is true that 1 is never equal to 2, so we  
       should see this alert.*/  
}
```

Déclarations `if/else` [1]

- Si vous voulez faire quelque chose si le test est vrai et quelque chose d'autre s'il est faux, incluez une instruction `else` après l'énoncé `if`:

```
var test = "testing";  
if( test == "testing" ) {  
    alert( "You haven't changed anything." );  
} else {  
    alert( "You've changed something!" );  
}
```

Si vous modifiez la valeur de la variable `test` en n'excluant que le mot «testing», l'alerte "You've changed something!"

Instruction switch

```
switch (expression) {  
    case choix1:  
        //exécuter ce code  
        break;  
  
    case choix2:  
        //exécuter ce code à la place  
        break;  
    // incorporez autant de case que vous le  
    souhaitez  
  
    default:  
        //sinon, exécutez juste ce code  
}  

```

Opérateur ternaire

`(condition) ? exécuter ce code : exécuter celui-ci à la place`

■ Exemple

```
var formuleDePolitesse = ( anniversaire ) ? 'Bon anniversaire Mme Smith  
— nous vous souhaitons une belle journée !' : 'Bonjour Mme Smith.';
```

Boucles [1]

- Les boucles vous permettent de faire quelque chose pour chaque variable d'un tableau sans écrire une instruction pour chacune.
- Une façon d'écrire une boucle est d'utiliser une instruction for:

```
for (condition initiale; condition de sortie; expression finale) {  
    // code à exécuter  
}
```

Boucles [2]

- Exemple:

```
for(var i = 0; i <= 2; i++) {  
    alert(i);  
}
```

- Quitter une boucle avec `break`
- Passer des itérations avec `continue`

while et do ... while

initializer

```
while (exit-condition) {
```

```
    // code to run
```

```
    final-expression
```

```
}
```

initializer

```
do {
```

```
    // code to run
```

```
    final-expression
```

```
} while (exit-condition)
```

Gérer les exceptions

- Pour lever des exceptions: utilise l'instruction `throw` et
- Pour gérer (intercepter) des exceptions : utilise des instructions `try...catch`.
- L'instruction `throw`
 - *est utilisée afin de signaler une exception.*
 - *Lorsqu'on signale une exception, on définit une expression qui contient la valeur à renvoyer pour l'exception.*

`throw` expression;

- Exemple:

```
throw "Erreur2"; //type String
throw 42; //type Number
throw true; //type Boolean
throw {toString: function () { return "je suis un
objet !"; } };
```

Instruction `try...catch`

- Permet de définir un bloc d'instructions qu'on essaye (`try` en anglais) d'exécuter, ainsi qu'une ou plusieurs instructions à utiliser en cas d'erreur lorsqu'une exception se produit.
- Si une exception est signalée, l'instruction `try...catch` permettra de l' « attraper » (`catch` en anglais) et de définir ce qui se passe dans ce cas.

```

function getNomMois(numMois) {
    numMois = numMois - 1;
    // On décale de 1 car les indices du tableaux commencent à 0
    var mois = ["Janvier", "Février", "Mars", "Avril", "Mai",
"Juin", "Juillet",
"Août", "Septembre", "Octobre", "Novembre", "Décembre"];
    if (mois[numMois] != null) {
        return mois[numMois];
    } else {
        throw "NuméroMoisInvalide";
        // Ici on utilise l'instruction throw
    }
}

try { // les instructions à essayer si tout se passe bien
    nomMois = getNomMois(maVarMois);
    // La fonction peut renvoyer une exception
} catch (e) {
    nomMois = "inconnu";
    gestionErreurLog(e); // on gère l'erreur avec une fonction
}

```

Syntaxe

```
try {  
    instructions_try  
}  
[catch (exception_var_1 if condition_1) {  
    // non-standard  
    instructions_catch_1  
}]  
...  
[catch (exception_var_2) {  
    instructions_catch_2  
}]  
[finally {  
    instructions_finally  
}]
```

Bloc `finally`

- Contient les instructions à exécuter après les blocs `try` et `catch` mais avant l'instruction suivant le `try...catch`.
- Est exécuté dans tous les cas, qu'il y ait une exception de produite ou non. Si une exception est signalée et qu'il n'y a pas de bloc `catch` pour la gérer, les instructions du bloc `finally` seront tout de même exécutées.
- Peut être utilisé afin de finir proprement l'exécution malgré une exception.
 - *Par exemple: devoir libérer une ressource, ou fermer un flux, etc.*

Exemple

```
ouvrirFichier();  
try {  
    écrireFichier(données);  
    // Une erreur peut se produire  
} catch(e) {  
    gérerException(e);  
    // On gère le cas où on a une exception  
} finally {  
    fermerFichier();  
    // On n'oublie jamais de fermer le flux.  
}
```

Références

- <https://developer.mozilla.org/fr/docs/Apprendre/JavaScript>
- <https://www.w3schools.com/js/default.asp>
- Learning Web design

Exercice 1

1. Créez un projet contenant trois fichiers: html, css et javascript.
2. Demandez utilisateur à entrer son nom et prénom
3. Vérifier si la chaîne entrée n'est pas vide, affichez un message, par exemple: » Bonjour» si oui. Sinon, levez une exception et la gérez.

Exercice 2

1. Créez un projet contenant trois fichiers: html, css et javascript.
2. Votre page permet à utilisateur d'entrer deux nombres, sélectionne une opération (addition, soustraction, multiplication, division) puis affichez le résultat.

Exercice 3

1. Écrivez un Javascript qui permet de retourner la valeur maximale de trois entiers entrés par l'utilisateur.
2. Écrivez un Javascript qui permet de retourner la valeur maximale de n entiers entrés par l'utilisateur, séparé par un virgule
3. Écrivez un Javascript qui permet de convertir un entier en heures et minutes.
4. Écrivez un Javascript qui permet de retourner la première et la dernière lettre d'une chaîne de caractère entrée par l'utilisateur.

Question ?