

JAVASCRIPT

Durée Totale du Module : 21H

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Gestion des erreurs

Try ... Catch

Aspect très important de la programmation d'autant plus quand on travaille dans une équipe, progressivement vos applications vont se complexifier et contenir beaucoup de code, d'objet, de fonctions etc..

Il y a des erreurs plus ou moins grave et de sources différentes

Les classiques :

- Erreur de Syntaxe, oubli d'un « ; », length ou lenght ?, etc...
- Erreur qui vient du serveur (pratique on peut accuser les développeurs BackEnd, l'incapacité à charger un fichier (404, etc...))
- Erreur qui vient du navigateur
- Erreur qui vient de l'utilisateur (envoyer une valeur non conforme dans un formulaire par exemple)

Dans la majorité des cas, une erreur va provoquer l'arrêt brutal d'un script et on risque donc d'avoir des codes et des pages non fonctionnelles.

Dans le pire des cas, une erreur peut être utilisée comme faille de sécurité par des utilisateurs malveillants qui vont l'utiliser pour dérober des informations ou pour tromper les autres visiteurs.

Javascript comporte nativement des fonctionnalités pour gérer les cas d'erreurs.

Dans tous les cas vous avez pu le constater, quand votre code comporte une erreur (vous avez un message d'erreur de base dans la console du navigateur), cela signifie que de base Javascript va créer, générer une erreur à partir de l'objet global Error (un objet de base dans javascript)

Par la suite nous allons pouvoir capturer l'objet d'Error renvoyé par Javascript et pouvoir indiquer ce que l'on souhaite faire dans le code quand cette erreur survient.

On va avoir à disposition un syntaxe de bloc try ... catch...

Dans le bloc try : on écrit le code à tester (qui peut potentiellement générer des erreurs)

Dans le bloc catch : on écrit le code pour gérer l'erreur (on fait un simple console.log ? On affiche un message à l'utilisateur ? On enregistre quelque chose en base de données ?

Nous allons voir le gestion d'erreurs au travers d'exemples.

Auteur :

Jean-François Pech

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

```
prenom;  
alert('Ce message ne s\'affichera pas');
```

Erreur générée par la fonction alerte

✖ Uncaught SyntaxError: Invalid or unexpected token (at app.js:4:7) app.js:4

Erreur générée par la variable prénom

✖ ▶ Uncaught ReferenceError: prenom is not defined app.js:3
 at app.js:3:1

Donc si on sait que ce code d'exemple peut nous créer des erreurs on peut l'écrire de cette manière :

```
try{  
  prenom  
  alert('Bonjour');  
}catch(err){  
  alert(`Erreur Détectée ALERTE STOPPEZ TOUT :  
  -----  
  Le Nom de l'erreur  
  ${err.name}  
  -----  
  Le Message de l'erreur :  
  ${err.message}  
  -----  
  L'emplacement de l'erreur:  
  ${err.stack}`);  
}  
alert(`Ce message s'affiche correctement`);
```

Comme on a un erreur dans le code du bloc Try seule les alertes du bloc catch puis ensuite la dernière alert sera affichée à l'utilisateur.

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☑ Jérôme CHRETIENNE
 ☑ Sophie POULAKOS
 ☑ Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.



127.0.0.1:5500 says

Erreur Détectée ALERTE STOPPEZ TOUT:

Le Nom de l'erreur
ReferenceError

Le Message de l'erreur :
prenom is not defined

L'emplacement de l'erreur:
ReferenceError: prenom is not defined
at http://127.0.0.1:5500/app.js:7:5

☐ Don't allow this page to display more dialogs

OK



127.0.0.1:5500 says

Ce message s'affiche correctement

☐ Don't allow this page to display more dialogs

OK

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Gestion des exceptions avec Throw

Dans certains cas, lorsque l'on écrit les différentes fonctions d'un programme, il se peut que le code soit juste mais cela ne correspond plus à la réalité.

Imaginez vous réalisez une application pour gérer des retrait et virement bancaires, vous avez 10 euros sur votre compte et vous voulez faire un retrait de 50 K€, techniquement si on se place du point de vue du code il nous faudra certainement une fonction qui fait une soustraction.

Donc du point de vue du code on peut faire $10 - 50000$, ça va fonctionner et votre solde sera de -49990 €.

Le code est juste d'un point de vue technique mais du point de vue du Banquier peut être pas.

Dans notre cas il faut bien prendre en compte les règles de gestion de l'application et du corps de métier, et donc ici il nous faudra mettre en place une exception dans le cas où le montant que voudrait retiré serait supérieur au solde de l'utilisateur.

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Ici ce programme a pour but de demander 2 données à l'utilisateur (la fonction prompt), pour ensuite diviser ces données.

On va donc mettre en place 2 exceptions (il pourrait y en avoir plus)

1 exception si l'utilisateur ne renseigne pas 2 nombres.

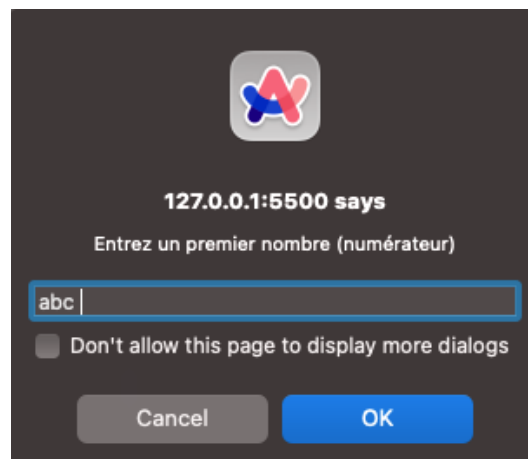
1 exception si l'utilisateur essaye de nous faire diviser par zéro

```
function division(){
  let x = prompt('Entrez un premier nombre (numérateur)');
  let y = prompt('Entrez un deuxième nombre (dénominateur)');

  if(isNaN(x) || isNaN(y) || x == '' || y == ''){
    throw new Error('Merci de rentrer deux nombres');
  }else if(y == 0){
    throw new Error('Division par 0 impossible')
  }else{
    alert(x / y);
  }
}

division();
```

1 ère exception :



✖ ▶ Uncaught Error: Merci de rentrer deux nombres
 at div (app.js:29:15)
 at app.js:37:1

app.js:29

Auteur :

Jean-François Pech

Relu, validé & visé par :

☑ Jérôme CHRETIENNE
 ☑ Sophie POULAKOS
 ☑ Mathieu PARIS

Date création :

03/03/2023


Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

2^e exception :



127.0.0.1:5500 says

Entrez un deuxième nombre (dénominateur)

☐ Don't allow this page to display more dialogs

Cancel OK

✖ ▶ Uncaught Error: Division par 0 impossible
at div (app.js:31:15)
at app.js:37:1

app.js:31

Auteur :

Jean-François Pech

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Finally

Au sein d'un bloc try catch, on peut (optionnel) utiliser l'instruction Finally, ce bloc nous permet de préciser du code qui sera exécuté dans tous les cas, qu'une erreur ou exception ait été générée ou pas.

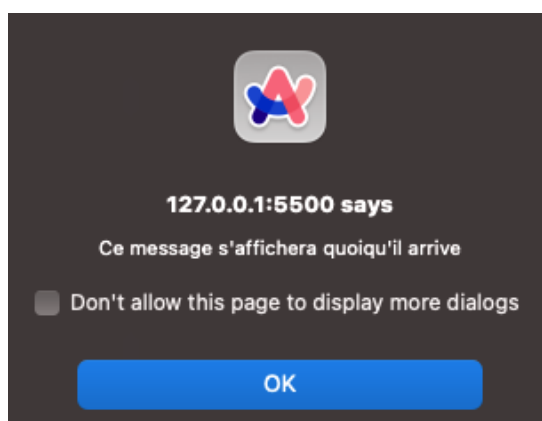
Reprenons notre fonction division : ici on ne va plus exécuter la fonction division directement dans notre programme on va **essayer** d'exécuter la fonction.

```
function division(){
    let x = prompt('Entrez un premier nombre (numérateur)');
    let y = prompt('Entrez un deuxième nombre (dénominateur)');

    if(isNaN(x) || isNaN(y) || x == '' || y == ''){
        throw new Error('Merci de rentrer deux nombres');
    }else if(y == 0){
        throw new Error('Division par 0 impossible');
    }else{
        alert(x / y);
    }
}

// division();

try{
    division();
}catch(err){
    alert(err.message);
}finally{
    alert(`Ce message s'affichera quoiqu'il arrive`);
}
```



Auteur :

Jean-François Pech

Relu, validé & visé par :

☑ Jérôme CHRETIENNE
☑ Sophie POULAKOS
☑ Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.