

# Error Handling in JS

++Debounce,  
++JS Libraries - JQuery, Bootstrap



**IT TALENTS**  
Training Camp

# What is an error in programming?

Things don't go always well in our programs.

In particular, there are situations where we may want **to stop the program or inform the user if something bad happens.**

For example:

- ▶ the program tried to open a non-existent file.
- ▶ the network connection is broken.
- ▶ the user entered invalid input.

In all these cases we as programmers, create **errors**, or we let the programming engine create some for us.

After creating the error we can inform the user with a message.

# What are the errors in JS?

- ▶ The Error in Javascript is a simple object that contains a few characteristics of the error. For example:
  - ▶ What is the type of the error
  - ▶ Where the error occurred(file, line, character, etc...)
  - ▶ What is the error message

# Types of Errors

- ▶ Error
  - ▶ EvalError
  - ▶ InternalError
  - ▶ RangeError
  - ▶ ReferenceError
  - ▶ SyntaxError
  - ▶ TypeError
  - ▶ URIError
- 
- ▶ Reference: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Error](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Error)

# How to create an Error?

- ▶ To create a new error in JavaScript we call the appropriate **constructor function**. For example, to create a new, generic error we can do:  

```
const err = new Error("Error occurred!");
```
- ▶ Using the same approach you can create any other type of error using their constructors: `new TypeError()`, `new SyntaxError()`, etc..
- ▶ Once created, the error object presents three properties:
  - ▶ **message**: a string with the error message.
  - ▶ **name**: the error's type.
  - ▶ **stack**: a stack trace of functions execution.

# What happens when an error is thrown?

- ▶ Error objects are **thrown** when runtime errors occur.
- ▶ When an error is thrown the execution of the current function **will stop (No statements after the error are going to be executed!)**. The control will be passed to the first **catch** statement.
- ▶ If no catch block exists among caller functions, **the program will terminate**.
- ▶ Exceptions are like an elevator going up: once you throw one, it bubbles up in the program stack, unless it is caught somewhere.

# How to handle errors in JS?

Now lets make sure these exceptions don't crash our apps...

- ▶ Handling errors in JS is achieved by these statements: **try**, **catch**, **finally**.

```
try {  
    // a function that potentially throws an error  
    someFunction();  
} catch (err) {  
    // this code handles exceptions  
    console.log(err.message);  
} finally {  
    // this code will always be executed  
    console.log('finally');  
}
```

# Handling errors

- ▶ The **try** block is mandatory and wraps a block of code that potentially can throw an error.
- ▶ It is followed by a **catch** block, which wraps JavaScript code that handles the error.
- ▶ The catch clause **stops the error from propagating** through the call stack and **allows the application flow to continue**. The error itself is passed as an argument to the catch clause.
- ▶ The **finally** code block is executed after the try and catch clauses, regardless of any exceptions.



# How to throw an exception?

- ▶ JavaScript allows developers to trigger exceptions via the **throw** statement.

```
throw new Error('Slatkata is under 18 years old');
```

- ▶ Each of the built-in error objects takes an optional “message” parameter that gives a human-readable description of the error.
- ▶ You can also throw any type of object as an Exception, such as: Numbers, Strings, Arrays, etc. However, throwing an Error objects is always the preferable approach.



# Error Handling in Promises

- ▶ In the realm of Promise, **catch** is the construct for handling errors.

```
getUsers()  
  .then(result => result.json())  
  .catch(error => console.error(error.message))  
  .finally(() => console.log('AJAX completed'));
```

- ▶ In addition to **catch** and **then** we have also **finally**, similar to the finally in try/catch.

# Working with Fetch

- ▶ Fetch promises **only reject with a `TypeError` when a network error occurs**. Since 4xx and 5xx responses aren't network errors, there's nothing to catch.
- ▶ However, the Response object contains a flag called: **`ok`**. This flag indicates whether an HTTP response's status code is in the successful range or not.

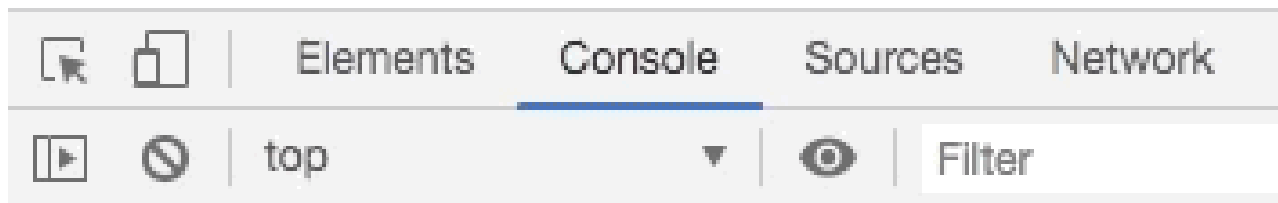
```
fetch("http://httpstat.us/500")
  .then(res => {
    if (!res.ok) {
      throw new Error(res.statusText);
    }
    return res;
  })
  .then(res => console.log(res))
  .catch(err => console.error(err));
```

# Write once - use anywhere

```
function genericFetch(url, options = {}) {  
    return fetch(url, options)  
        .then(res => {  
            if (!res.ok) {  
                throw new Error(res.statusText);  
            }  
            return res;  
        });  
}
```

# Event Handlers -> The Problem

- ▶ Certain DOM events like scrolling, resizing are firing waaay too much times than we want to. Now imagine a scenario where you have a scroll handler which does a bunch of work (like heavy calculations or DOM manipulations). You are definitely going to see performance issues!



# Debounce and Throttle

- ▶ **Debounce** and **throttle** are two similar (but different!) techniques to control how many times we allow a function to be executed over time.
- ▶ The Debounce technique allow us to “group” multiple sequential calls in a single one.

Debounced:

Regular:

- ▶ Example: <https://redd.one/blog/debounce-vs-throttle>

# Demo



# JS Libraries

Write less.. Achieve more..



**IT TALENTS**  
Training Camp



# What is a JS library/package?

- ▶ A library is a JavaScript file that contains a bunch of functions, and those functions accomplish some useful task for your webpage.
- ▶ How do we know what functions we can use?
  - ▶ **Read the library documentation!**
- ▶ Open sourced projects available in Github. Everyone can see/contribute to these projects.
- ▶ Community -> The people who are using this library. Choosing popular libraries gives us bigger chance to find help or answers to our questions.

# How to install a package?

- ▶ Using the npm (node package manager)
  - ▶ Run **npm init** in your project
  - ▶ Then install any packages using the **npm install <package\_name>**
  - ▶ As you already know the **node\_modules** folder should NOT be committed in Github
- ▶ Using other package managers like: yarn, bower, etc
- ▶ Download the code of the library
- ▶ Using CDN: <https://cdnjs.com/>



# Useful JS Libraries

- ▶ Lodash -> JavaScript utility library delivering modularity, performance & extras.
- ▶ Moment -> Display dates and times in JavaScript.
- ▶ D3.js -> Data manipulation in Javascript. Charts and more
- ▶ Anime.js -> Create animations easily
- ▶ Bootstrap -> UI framework with reusable components!

# Bootstrap

- ▶ Components like: Accordions, Modals, Carousel, Pagination are taking so much time to built, right? Well that won't be the case anymore..
- ▶ Bootstrap is a library that saves us so many time in building these components.
- ▶ It contains a lot of JS, HTML and CSS code which we can reuse easily.
- ▶ Installation. In order to use Bootstrap you only have to include:
  - ▶ CSS: `<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-BmbxuPwQa2lc/FVzBcNJ7UAyJxM6wuqlj61tLrc4wSX0szH/Ev+nYRRuWlolflfl" crossorigin="anonymous">`
  - ▶ JS: `<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/js/bootstrap.bundle.min.js" integrity="sha384-b5kHyXgcpbZJO/tY9Ul7kGkf1S0CWuKcCD38l8YkeH8z8QjE0GmW1gYU5S9FOnJ0" crossorigin="anonymous"></script>`
- ▶ <https://getbootstrap.com/docs/5.0/getting-started/introduction/>



# Start using it...

Click to toggle popover

**Popover title**  
And here's some amazing content. It's very engaging. Right?

Accordion Item #1

**This is the first item's accordion body.** It is hidden by default, until the collapse plugin adds the appropriate classes that we use to style each element. These classes control the overall appearance, as well as the showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our default variables. It's also worth noting that just about any HTML can go within the `.accordion-body`, though the transition does limit overflow.



Accordion Item #2

Accordion Item #3

< First slide >

**First slide label**  
Some representative placeholder content for the first slide.


— — —

 **Bootstrap** 11 mins ago 

Hello, world! This is a toast message.

**Modal title**

Modal body text goes here.



Close Save changes

Tooltip on bottom

Tooltip on bottom

Primary

Secondary

Success

Danger

Warning

Info

Light

Dark

[Link](#)