vivasoft

Learnathon 2.0 - JS Class 8

•••

Error Handling and Others

Kazi Tajnur Islam | Software Engineer (L-2) | Vivasoft Limited

vivasoft

Different Types of Errors in JavaScript

7 types of native errors in JavaScript you should know:

1. Syntax Error
2. Reference Error
3. Type Error
4. Range Error
5. URI Error
6. Internal Error
7. Evaluation Error

vivasoft

Try-Catch-Finally

The code in the *try* block is executed first, and if it throws an exception, the code in the *catch* block will be executed. The code in the *finally* block will always be executed before control flow exits the entire construct.

```
try {
  tryStatements
} catch (exceptionVar) {
  catchStatements
} finally {
  finallyStatements
}
```

vivasoft

Try-Catch-Finally

Unlike other constructs such as *if* or *for*, the *try*, *catch*, and *finally* blocks must be blocks, instead of single statements.

```
try doSomething(); // SyntaxError
catch (e) console.log(e);
```

vivasoft

Custom Error Throw

You can create custom errors by extending the built-in *Error* object or any other error object like *TypeError*, *SyntaxError*, etc. Creating custom errors can be helpful for your applications to provide more specific information about the nature of an error and handle it accordingly.

```
class CustomError extends Error {
  constructor(message) {
    super(message);
    this.name = "CustomError";
  }
}
```

vivasoft

Garbage Collection

Garbage collection is a crucial part of memory management in JavaScript. It helps automatically clean up memory by identifying and reclaiming objects that are no longer reachable or in use by the application. JavaScript engines handle garbage collection automatically in the background, without explicit developer intervention. Developers do not need to worry about freeing memory manually, as long as they manage references properly.

vivasoft

Garbage Collection

To work effectively with the garbage collector:

- Minimize global variables and avoid creating unnecessary references.
- Use variables within limited scopes and ensure they go out of scope when they are no longer needed.
- Remove event listeners and other references to objects when they are no longer required.
- Be aware of potential memory leaks in closures and circular references.

vivasoft

Memory Leak

A memory leak in JavaScript occurs when an application unintentionally retains references to objects or data in memory that are no longer needed, preventing the garbage collector from freeing up that memory. Over time, these retained references can lead to increased memory consumption and degrade the performance of your application.

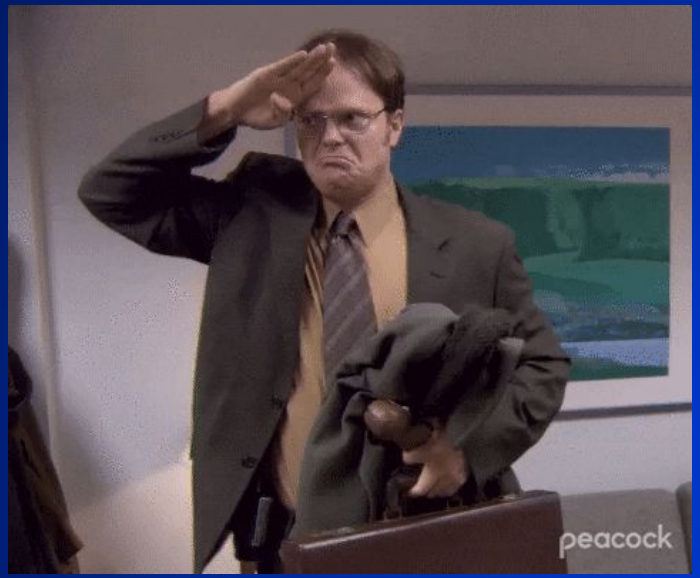
```
const data = [];

function addData() {
  for (let i = 0; i < 10000; i++) {
    data.push({ value: i });
  }
}
```

vivasoft

Q&A time

Learnathon - 2.0

vivasoft



Thank You! Goodbye Everyone!

Learnathon - 2.0