Errors and Error Handling

ReferenceError

A *ReferenceError* is a type of error thrown when a variable is used that does not exist.
To prevent this error, all variables should be properly declared beforehand.

```
// Example of a ReferenceError in JavaScript
let firstName = "John";

// Here, we get a ReferenceError because
lastName has not been declared
console.log(firstName + lastName);
```

MDN JavaScript error documentation

The MDN JavaScript error documentation contains information about JavaScript error types, properties, and Methods. The document shows how to prevent and create these errors. This document is most helpful when developers come across an error they are not familiar with.

SyntaxError

A *SyntaxError* is a type of error that is thrown when there is a typo in the code, creating invalid code - code which cannot be interpreted by the compiler.
Some common causes of a *SyntaxError* are:

- Missing opening or closing brackets, braces, or parentheses
- Missing or invalid semicolons
- Misspelling of variable names or functions

```
# Example of a SyntaxError in Python

# A colon is missing after the closing
parenthesis
def sum(a, b)
  return a + b
```

TypeError

A *TypeError* is a type of error thrown when an attempt is made to perform an operation on a value of the incorrect type.
One example of a *TypeError* is using a string method on a numerical value.

```
# Example of a TypeError in Python
number = 1
string = "one"

# Here, we try to concatenate the number and
string which will yield a TypeError
print(number + string)
```

Javascript error stack trace

An error stack trace tells a developer that it has detected an error within the code. Along with, which line to find the error, what type of error has occurred and a description of the error.

Javascript documentation

Many times we can track down bugs, but still, be confused about how to solve it. During these situations, we can look at documentation. For JavaScript, the *MDN JavaScript web docs* is a powerful resource. If we are still confused after looking at this we can go to StackOverflow - a question and answer forum where programmers post issues and other programmers discuss and vote for solutions.

Runtime Error in JavaScript

A JavaScript runtime error is an error that occurs within code while its being executed. Some runtime errors are built-in objects in JavaScript with a name and message property. Any code after a thrown runtime error will not be evaluated.

The `throw` Keyword in JavaScript

The JavaScript `throw` keyword is placed before an `Error()` function call or object in order to construct and raise an error. Once an error has been thrown, the program will stop running and any following code will not be executed.

```
// The program will raise and output this
Error object with message 'Something went
wrong'
throw Error('Something went wrong');

//The program will stop running after an
error has been raised, and any following
code will not be executed.
console.log('This will not be printed');
```

Javascript Error Function

The JavaScript *Error()* function creates an error object with a custom message. This function takes a string argument which becomes the value of the error's `message` property. An error created with this function will not stop a program from running unless the `throw` keyword is used to raise the error.

```
console.log(Error('Your password is too
weak.')); //Error: Your password is too
weak.
```

javascript try catch

A JavaScript `try` … `catch` statement can anticipate and handle thrown errors (both built-in errors as well as those constructed with `Error()`) while allowing a program to continue running. Code that may throw an error(s) when executed is written within the `try` block, and actions for handling these errors are written within the `catch` block.

```
// A try...catch statement that throws a constructed Error()
try {
  throw Error('This constructed error will be caught');
} catch (e) {
  console.log(e); // Prints the thrown Error object
}

// A try...catch statement that throws a built-in error
const fixedString = 'Cannot be reassigned';
try {
  fixedString = 'A new string'; // A TypeError will be thrown
} catch (e) {
  console.log('An error occurred!'); // Prints 'An error occurred!'
}

console.log('Prints after error'); // Program continues running after the error is handled and prints 'Prints after error'
```