**Error Constructing and Handling**

1. **Runtime Errors**

When we execute code and a line of code throws an error, that error is referred to as a *runtime error*. In JavaScript, there are built-in error objects that have a name and message property which tell us what went wrong. Examples of built-in runtime errors include:

* ReferenceError: when a variable or function cannot be found.
* TypeError: when a value is not a valid type

1. **Constructing an Error**

JavaScript errors are objects that have a name and message property. What if we need an error message isn’t covered by the built-in errors (e.g., the string entered is too short)

🡪 We use the Error() function to create our own error object.

- The Error function takes an argument of a string which becomes the value of the error’s message property.

- Errors can also be created with the new keyword. This is similar to calling the Error() function only, but these are **different from throwing an error.** *Throwing an error will stop the program.*

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

1. **Throw keyword**

Using the throw keyword will cause the program to stop running.

throw Error('Something wrong happened');  
// Error: Something wrong happened  
console.log('This will never run');

1. **try…catch Statement**

Up to this point, thrown errors have caused our program to stop running. But, we have the ability to anticipate and *handle* these errors by writing code to address the error and allow our program to continue running.

In JavaScript, we use **try...catch statement** to anticipate and handle errors.

Generally speaking, in a try...catch statement, we evaluate code in the try block and if the code throws an error, the code inside the catch block will handle the error for us.

const someVar = 'Cannot be reassigned';  
try {  
  someVar = 'Still going to try';  
} catch(e) {  
  console.log(e);  
}  
// Prints: TypeError: Assignment to constant variable.