Telerik Academy

JavaScript Exceptions

Fixing some of the bad JavaScript practices

Telerik Software Academy
Learning & Development Team
http://academy.telerik.com

Telerik Academy

Table of Contents

Telerik Academy

JavaScript Exceptions

Telerik Academy

JavaScript Exceptions

- Exceptions are special objects that hold information about errors
- Exceptions are the correct way to handle errors in a programming language
  - Almost every object or function can throw an exception
- Most common exceptions in JavaScript are:
  - ReferenceError
  - TypeError
  - SyntaxError

Telerik Academy

Exception Handling in JavaScript

Telerik Academy

Exception Handling

- Handling exceptions means:
  1. Catch an exception
  2. Resolve the error
  3. Continue the execution of the application
- Exception handling provides a way to catch the exception without breaking the workflow of an application
  - Catch the error, solve it and then continue the execution of the application

Telerik Academy

Exception Handling (2)

- Exception handling is done using a try-catch block

```
try {
  // code that can throw an exception
} catch (ex) {
  // if the above code throws an exception this code is
  // executed and ex holds the info about the exception
}
```

- Try runs an error-prone code that can throw an exception
  - If an exception is thrown the execution of the code inside the try stops and the code in the catch block is executed

Telerik Academy

Exception Handling (3)

- Every try-catch block can contain only one try and one catch block
  - If you are looking to catch a specific exception, check the exception type inside the catch block
- The exception object holds information about the exception
  - Its type
  - An exception message

Telerik Academy

Exception Handling: Example

- Create a function that throws an exception based on an argument

```
try {
  throwException("TypeError");
} catch (ex) {
  console.log("---------------");
  console.log("Exception object: " + ex);
  console.log("Type: " + ex.name);
  console.log("Message: " + ex.message);
  console.log("---------------");
}
```

Telerik Academy

Exception Handling

Live Demo

Telerik Academy

Handling Multiple Exception

React differently based on the exception type

Telerik Academy

Handling Multiple Exception

- try-catch blocks contain a single catch
  - How can our code handle multiple exceptions?
  - Make an if-else on the exception type
- If looking for a TypeError, SyntaxError or ReferenceError
  - Check the type of the exception object

```
try {
  throwException("reference");
} catch (ex) {
 if (ex instanceof TypeError) { … }
 else if (ex instanceof ReferenceError) { … }
 else if (ex instanceof SyntaxError) { … }
}
```

Telerik Academy

Handling Multiple Exception

Live Demo

Telerik Academy

Creating and Throwing Exceptions

Creating Exception

- Exceptions can be created using an exception constructor
  - Different constructor for each exception type

```
var typeException = new TypeError([message]);
var rangeException = new RangeError([message]);
```

  - The constructor takes an optional message
    - If skipped, the message is an empty string
- To throw an exception use the reserved word `throw` with an exception object

```
var typeEx = new TypeError("Not correct use of an object");
throw typeEx;
```

Telerik Academy

Throwing Exceptions

Live Demo

Telerik Academy

Custom Exceptions

Telerik Academy

Custom Exception

- Custom exceptions are made by just inheriting the Error type

```
function AgeError(message, minAge, maxAge) {
  this.message = message;
  this.minAge = minAge;
  this.maxAge = maxAge;
}
AgeError.inherit(Error);
```

  - Our custom exception is ready

- A drawback to this approach is that there is no benefit from inheriting Error

Telerik Academy

Custom Exceptions

Live Demo

Telerik Academy

More about Custom Exceptions

- Inheriting `Error` is the OOP way
  - Yet what we gain from this inheritance?
    - Performance is slower
    - Constructors are hardly to reuse
- Remember JavaScript is loosely-typed?
  - The only thing a catch wants is an object
    - It does not care what kind of object

Telerik Academy

More about Custom Exceptions (2)

- To use an object as an exception, it only needs a message property
  - It is not obligatory, but this is a good practice
  - The following are correct exception usages

```
throw { message: "sample exception“ };
throw { message: "Age is out of range",
        minAge: 0, maxAge: 135
};
throw document.createElement("div"); // don't do this!
```

- JS does not care whether the thrown object is of type Error
  - It only cares if the thrown thing is an object
    - Does not work with primitives

Telerik Academy

Throwing Objects

Live Demo

Telerik Academy

JavaScript Exceptions

http://academy.telerik.com