**Обробка виключних ситуацій**

У випадку, якщо у програмі виникає ситуація, коли подальше штатне виконання є неможливим (як правило – це критична помилка), то у програмі генерується виключна ситуація. Для обробки виключних ситуацій може бути використано блок try/catch

try {

    Блок коду, який може

згенерувати виключну ситуацію

}  
catch(err) {

     Блок коду – аналіз на виключну ситуацію

}

finally {

Блок коду, який виконується не залежно від того,  
чи виникла виключна ситуація, чи ні

}

Хоча при генерації виключних ситуацій може бути використано String, Number, Boolean або Object, як правило використовуються саме об’єкти. У JavaScript є ряд вбудованих об’єктів помилок.

|  |  |
| --- | --- |
| **Error Name** | **Description** |
| EvalError | An error has occurred in the eval() function |
| RangeError | A number "out of range" has occurred |
| ReferenceError | An illegal reference has occurred |
| SyntaxError | A syntax error has occurred |
| TypeError | A type error has occurred |
| URIError | An error in encodeURI() has occurred |

Об’єкт помилки має такі властивості

|  |  |
| --- | --- |
| **Property** | **Description** |
| name | Встановлення або зчитування імені виключної ситуації |
| message | Встановлення або зчитування повідомлення, що пов’язане з виключною ситуацією |
| stack | Встановлення або зчитування стеку викликів, в результаті яких було згенеровано виключну ситуацію |

try {

throw new Error('oops');

}

catch (ex) {

console.error(ex.message);

}

Є також можливість аналізувати тип помилки з використанням [instanceof](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/instanceof)

try {

foo.bar();

} catch (e) {

if (e instanceof EvalError) {

console.log(e.name + ': ' + e.message);

} else if (e instanceof RangeError) {

console.log(e.name + ': ' + e.message);

}

// ... и т.д.

}

**Генерування виключної ситуації**

Виключна ситуація може генеруватися і за допомогою оператора

throw об’єкт виключної ситуації.

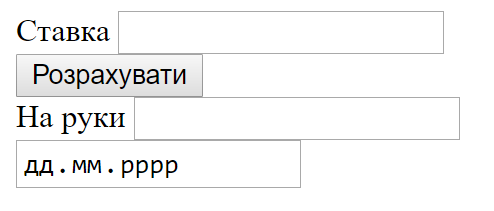
Загалом, об’єктом виключної ситуації може бути String, Number, Boolean або Object.

throw "Too big";    // throw a text  
throw 500;          // throw a number

Приклад.

<input id="demo" type="text">  
<button type="button" onclick="myFunction()">Test Input</button>  
<p id="p01"></p>  
  
<script>  
function myFunction() {  
    var message, x;  
    message = document.getElementById("p01");  
    message.innerHTML = "";  
    x = document.getElementById("demo").value;  
    try {   
        if(x == "") throw "empty";  
        if(isNaN(x)) throw "not a number";  
        x = Number(x);  
        if(x < 5) throw "too low";  
        if(x > 10) throw "too high";  
    }  
    catch(err) {  
        message.innerHTML = "Input is " + err;  
    }  
}

Приклад. З клавіатури вводиться ставка. Розрахувати розмір з/п (80%). Зробити перевірку введеного значення ставки (без використання HTML5).



<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title></title>

<script>

function calculateSalary()

{

var stavka = document.forms[0].elements[0].value;

try {

//---- Перевірка чи є числом

if (isNaN(stavka))

throw new Error("Ставка має бути числом");

//---- Перевірка чи є >0

stavka = parseFloat(stavka);

if (stavka < 0)

throw new Error("Ставка не може бути від'ємною");

//---- Перевірка чи є <90000

if (stavka > 90000)

throw new Error("Ставка не може бути >90000");

var salary = stavka \* 0.8;

document.forms[0].elements[2].value = salary;

}

catch (err)

{

alert(err.message);

}

alert("ok")

}

window.onload = function () {

btn.onclick = calculateSalary;

}

</script>

</head>

<body>

<form action="www.myserver.com" method="get">

Ставка <input type="text" name="Stavka" value="" /><br>

<input id="btn" type="button" value="Розрахувати" /><br>

На руки <input type="text" name="NaRuki" value="" /><br>

<input type="date" name="name" value="" />

</form>

</body>

</html>

Посилання:

<https://www.w3schools.com/js/js_errors.asp>

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch>

Створення власних виключних класів ситуацій

Можна також створювати типи власних виключних ситуацій. Для цього необхідно використати деякий із об’єктів помилок як предка, але при цьому самостійно задати властивості : name, message, stack.

При цьому можна використовувати як функціональний стиль, так і сучасний стиль

// Создаём новый объект, затем через прототип делаем его наследником конструктора Error.

function MyError(message) {

this.name = 'MyError';

this.message = message || 'Сообщение по умолчанию';

this.stack = (new Error()).stack;

}

MyError.prototype = Object.create(Error.prototype);

MyError.prototype.constructor = MyError;

try {

throw new MyError();

} catch (e) {

console.log(e.name); // 'MyError'

console.log(e.message); // 'Сообщение по умолчанию'

}

try {

throw new MyError('пользовательское сообщение');

} catch (e) {

console.log(e.name); // 'MyError'

console.log(e.message); // 'пользовательское сообщение'

}

**З використанням сучасного стилю**

//--------- Опис власного класу помилки ---------------

class MyError extends Error {

constructor(message="Моя виключна ситуація",row)

{

super();

this.row=row;

this.name ="MyError";

this.message = message;

this.stack = (new Error()).stack;

}

}

//---------- Використання власного класу помилки -----------

try {

throw new MyError(“ddfdf”,8);

} catch (e) {

console.log(e.row); // 'MyError'

console.log(e.message); // 'Моя виключна ситуація'

}

Приклад. Користувач задає стаж та заробітну плату. Визначити загальний об’єм виплачених грошей. Предбачити перевірку значень.

Проблема була у способі підключення (див. window.onload). Доробіть задачу додавши перевірку і на друге поле. Потім спробуйте описати клас «Захищене поле» (передаєте title, max, min, значення та правило введення (regexp).

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>My calculator</title>

<script>

//----------------------

class IsNotNumberError extends Error {

constructor() {

super();

this.message = "Має бути числом";

this.name = "IsNotNumberError";

this.stack = (new Error()).stack;

}

}

//-------------------------

class IsNegativeNumberError extends Error {

constructor() {

super();

this.message = "Не може бути від'ємним";

this.name = "IsNegativeNumberError";

this.stack = (new Error()).stack;

}

}

//----------------------

class IsTooHightNumberError extends Error {

constructor(currentNumber) {

super();

this.currentNumber = currentNumber;

this.message = "Не може бути більшим за 65";

this.name = "IsTooHightNumberError";

this.stack = (new Error()).stack;

}

}

class IsHHHHHHError extends IsTooHightNumberError {

constructor() {

super();

}

}

//----------------------

function validate() {

try {

var exp = document.forms[0].elements[0].value;

if (isNaN(exp))

throw new IsNotNumberError();

if (exp < 0)

throw new IsNegativeNumberError();

if (exp > 65)

throw new IsTooHightNumberError(exp);

} catch (e) {

if (e instanceof IsNotNumberError) {

console.log(e.message + " Пишіть тільки числа");

document.forms[0].elements[0].value = 0;

}

else if (e instanceof IsNegativeNumberError) {

console.log(e.message + " Не може бути від'ємним.");

document.forms[0].elements[0].value = 0;

}

else if (e instanceof IsHHHHHHError) {

console.log(e.message + " Не може бути більшим за 65.");

document.forms[0].elements[0].value = 65;

}

else if (e instanceof IsTooHightNumberError) {

console.log(e.message + " Не може бути більшим за 65.");

document.forms[0].elements[0].value = 65;

}

else

console.log(e.message);

return false;

}

return true;

}

window.onload = function () {

document.forms[0].onsubmit = validate;

}

</script>

</head>

<body>

<form action="" method="get">

Стаж: <input type="text" name="exp" value="" /> <br>

З.п: <input type="text" name="salary" value="" /> <br>

<input type="submit" value="Надіслати" />

</form>

</body>

</html>

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title></title>

<script>

class MyError extends Error {

constructor(message="Моя виключна ситуація")

{

super();

this.name ="MyError";

this.message = message;

this.stack = (new Error()).stack;

}

}

class MyErrorSal extends MyError{

constructor(message = "Моя виключна ситуація",value) {

super(message);

this.value = value;

}

}

try {

throw new MyError();

} catch (e) {

console.log(e.name); // 'MyError'

console.log(e.message); // 'Моя виключна ситуація'

}

function validate() {

var v1=document.forms[0].elements[0].value;

var v2 = document.forms[0].elements[1].value;

alert(v1);

alert(v2);

try {

if (isNaN(v1)) {

throw new MyError("age i wrong");

}

if (isNaN(v2)) {

throw new MyErrorSal("salary i wrong", v2);

}

}

catch (e)

{

if (e instanceof MyErrorSal)

alert(e.message + "- " + e.value);

else

alert(e.message );

return false;

}

alert("ok")

return true;

}

</script>

</head>

<body>

<form action="" method="get" onsubmit="validate()">

Age <input type="text" name="name" value="" /><br>

Salary <input type="text" name="name" value="" />

<input type="submit" value="Submit" />

</form>

<script>

document.forms[0].onsubmit = validate;

</script>

</body>

</html>

==========================