

Обробка виключних ситуацій

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

Загальна форма

```
try {
```

Блок коду, який може згенерувати
виключну ситуацію

```
}
```

```
catch(err) {
```

Блок коду – аналіз на виключну
ситуацію

```
}
```

```
finally {
```

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

```
}
```

Приклад

```
try {
```

```
const val = parseInt(ed.value) / dollarRate  
alert(val)
```

Ця команда буде виконуватись
тільки тоді, якщо попередній
рядок виконається конектно

```
} catch (err) {
```

```
document.write('Помилка операції: ' + err.message)
```

```
} finally {
```

```
document.write('Дякуємо за користування!')
```

```
}
```

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
<code>EvalError</code>	An error has occurred in the <code>eval()</code> function
<code>RangeError</code>	A number "out of range" has occurred
<code>ReferenceError</code>	An illegal reference has occurred
<code>SyntaxError</code>	A syntax error has occurred
<code>TypeError</code>	A type error has occurred
<code>URIError</code>	An error in <code>encodeURIComponent()</code> 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

Приклад використання властивостей

```
try {
  const val = parseInt(ed.value) / dollarRate
  alert(val)
} catch (err) {
  console.log(err.message)
  console.log(err.name)
  console.log(err.stack)
} finally {
  document.write('Дякуємо за користування сервісом!')
```

Приклад виведення

```
ed is not defined
ReferenceError
ReferenceError: ed is not defined
at EditableDiv.html:15:30
```

Є також можливість аналізувати тип помилки з використанням *instanceof*

Загальна форма	Приклад
<pre>try { Блок коду, який може згенерувати виключну ситуацію } catch (err) { if (err instanceof тип_виключної_ситуації_1) дії_у_випадку_виключної_ситуації_1 else if (err instanceof тип_виключної_ситуації_2) дії_у_випадку_виключної_ситуації_2 } Блок коду, який виконується незалежно від того, чи виникла виключна ситуація, чи ні }</pre>	<pre>try { const val = parseInt(ed1[1].value) / dollarRate alert(val) } catch (err) { if (err instanceof ReferenceError) console.log('Елемента вводу не знайдено') else if (err instanceof TypeError) console.log('Елемента з таким номером немає') else console.log('Якась інша помилка') } finally { document.write('Дякуємо за користування сервісом!') }</pre>

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).

Ставка

На руки

```
<form>
Ставка
  <input type="text" name="Stavka" value=""/><br>
  <button id="btn"> Розрахувати </ button > <br>
На руки
  <input type="text" name="NaRuki" value="" /><br>
</form>
```

```
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;
}
```

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`.

Стаж: <input type="text"/>	<code><form></code>
З.п: <input type="text"/>	Стаж: <code><input type="text" name="exp" value="" />
</code>
<input type="button" value="Надіслати"/>	З.п: <code><input type="text" name="salary" value="" />
</code>
	<code><input type="submit" value="Надіслати" /></code>
	<code></form></code>

Опис власних класів помилок

```
//----у випадку значення не є числом----
class IsNotNumberError extends Error {
  constructor() {
    super()
    this.message = 'Має бути числом'
    this.name = 'IsNotNumberError'
  }
}

//---у випадку значення є від'ємним---
class IsNegativeNumberError extends Error {
  constructor() {
    super()
    this.message = "Не може бути від'ємним"
    this.name = 'IsNegativeNumberError'
  }
}

//---у випадку число є дуже великим---
class IsTooHightNumberError extends Error {
  constructor(currentNumber) {
    super()
    this.currentNumber = currentNumber
    this.message = "Не може бути більшим за 65"
    this.name = 'IsTooHightNumberError'
  }
}
```

Використання класів помилок

```
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 (err) {
    if (err instanceof IsNotNumberError) {
      console.log(err.message + ' Пишіть тільки числа')
      document.forms[0].elements[0].value = 0
    } else if (err instanceof IsNegativeNumberError) {
      console.log(err.message + " Не може бути від'ємним.")
      document.forms[0].elements[0].value = 0
    } else if (err instanceof IsTooHightNumberError) {
      console.log(err.message + " Не може бути більшим за 65.")
      document.forms[0].elements[0].value = 65
    } else console.log(err.message)
    return false
  }
  return true
}

window.onload = function () {
  document.forms[0].onsubmit = validate
}
```


Задача. Тир. Дано масив з 5 елементів, у якому випадково розміщено 1. Дати користувачу за 3 спроби знайти цю одиницю. Для аналізу результатів використати виключні ситуації.

