STEP IT ACADEMY

Занятие №13

Обработка ошибок. try, catch

Почему важно обрабатывать ошибки?

Почему это важно?

Неважно, насколько мы хороши в программировании, иногда наши скрипты содержат ошибки. Они могут возникать из-за наших промахов, неожиданного ввода пользователя, неправильного ответа сервера и по тысяче других причин.

Обычно скрипт в случае ошибки «падает» (сразу же останавливается), с выводом ошибки в консоль.

Но есть одна конструкция, которая позволяет ловить ошибки и вместо падения сайта делать что-то другое.

Синтаксис try...catch

Конструкция это состоит из двух основных блоков.

try и catch

```
1  try {
2
3    // код...
4
5  } catch (err) {
6
7    // обработка ошибки
8
9  }
```

Как работает?

Сначала выполняется код внутри блока try

Если в нём нет ошибок, то блок catch игнорируется: выполнение доходит до конца try и потом далее, полностью пропуская catch

Если же в нём возникает ошибка, то выполнение try прерывается, и поток управления переходит в начало catch(err) Переменная err (можно использовать любое имя) содержит объект ошибки с подробной информацией о произошедшем.

Пример без ошибок

```
1  try {
2
3    alert('Начало блока try');  // (1) <--
4
5    // ...код без ошибок
6
7    alert('Конец блока try');   // (2) <--
8
9  } catch(err) {
10
11   alert('Catch игнорируется, так как нет ошибок'); // (3)
12
13 }
```

Пример с ошибкой

```
try {

  alert('Начало блока try');  // (1) <--

  lalala; // ошибка, переменная не определена!

  alert('Конец блока try (никогда не выполнится)');  // (2)

} catch(err) {

  alert(`Возникла ошибка!`); // (3) <--

}
```

Объект ошибки

Когда возникает ошибка, JavaScript генерирует объект, содержащий её детали. Затем этот объект передаётся как аргумент в блок catch.

Для всех встроенных ошибок этот объект имеет два основных свойства:

name - Имя ошибки. Например, для неопределённой переменной это "ReferenceError“

message - Текстовое сообщение о деталях ошибки.

Пример 1:

```
let json = '{"name":"John", "age": 30}'; // данные с сервера

let user = JSON.parse(json); // преобразовали текстовое представление в JS-объект

// теперь user - объект со свойствами из строки
alert( user.name ); // John
alert( user.age );  // 30
```

Пример 2:

```
let json = "{ некорректный JSON }";

try {

  let user = JSON.parse(json); // <-- тут возникает ошибка...
  alert( user.name ); // не сработает

} catch (e) {
  // ...выполнение прыгает сюда
  alert( "Извините, в данных ошибка, мы попробуем получить их ещё раз." );
  alert( e.name );
  alert( e.message );
}
```

Пример 3:

```
let json = '{ "age": 30 }'; // данные неполны

try {

  let user = JSON.parse(json); // <-- выполнится без ошибок
  alert( user.name ); // нет свойства name!

} catch (e) {
  alert( "не выполнится" );
}
```

Задания

1. Создайте функцию на JavaScript, которая будет делить два числа, но при этом использовать блок try-catch для обработки возможной ошибки деления на ноль. Если при делении произошла ошибка, функция должна выдавать сообщение "Ошибка: деление на ноль!". Если ошибки не произошло, функция должна выводить результат деления.
2. Создайте функцию на JavaScript, которая будет принимать массив чисел и возвращать их сумму. Однако, если в массиве есть хотя бы один элемент, который не является числом, функция должна выдавать сообщение "Ошибка: в массиве есть неверные данные!" с использованием блока try-catch.
3. Создайте функцию на JavaScript, которая будет принимать строку и возвращать её длину. Однако, если входная строка не является строкой, функция должна выдавать сообщение "Ошибка: введенное значение не является строкой!" с использованием блока try-catch.

Давайте подведем итоги!

Чему мы научились?

Что мы использовали?