Обработка ошибок

Ошибки

Грамотная стратегия обработки ошибок должна информировать пользователей о том, что происходит, не отпугивая их, но для реализации такой процедуры нужно уметь применять разные способы обработки javaScript-ошибок по мере их возникновения.

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.

try … catch

В третьей редакции ECMA-262 для обработки исключений в JavaScript была представлена инструкция try-catch

```
try {
    // код, который может привести к ошибке
} catch (error) {
    // действия при возникновении ошибки
}
```

try … catch

Любой код, который может привести к ошибке, следует помещать в раздел try, а код обработки ошибки - в раздел catch

Если в разделе try происходит ошибка, выполнение кода немедленно прекращается и возобновляется с раздела catch, в который передается объект со сведениями об ошибке.

try … catch

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

- name (Имя ошибки. Например, для неопределённой переменной это "ReferenceError".)
- message (Текстовое сообщение о деталях ошибки.)

try … catch

```
try {
    global.someNotExistentFunction();
}
catch (error) {
    console.log(error.message);
}
```

try … catch - finally

Необязательное предложение finally инструкции try-catch выполняется после нее в любом случае независимо от того, возникла ли ошибка. Ничто в разделе try или catch, даже инструкция return, не может предотвратить выполнение кода в разделе finally.

```
function testFinally(){
    try {
        return 2;
    } catch (error){
        return 1;
    } finally {
        return 0;
    }
}
```

try … catch

Если ошибка возникает в инструкции try-catch, браузер не уведомляет о ней, потому что считает, что она обрабатывается должным образом. Это идеально подходит для веб-приложений, ориентированных на пользователей без технической подготовки, которым не имеет смысла сообщать об ошибках. С помощью try-catch можно реализовать собственный механизм обработки ошибок конкретных типов.

IT SHATLE

Задача 1

Создать блок обработки ошибок - при вызове несуществующей функции показывать пользователю ошибку на экране “что-то пошло не так”

Типы ошибок

JavaScript-ошибки делятся на несколько категорий, каждой из которых соответствует особый объект, генерируемый при ошибке. В ECMA-262 определены семь типов ошибок:

- Error;
- EvalError - ошибка, возникающая в глобальной функции;
- RangeError - ошибка, возникающую при выходе числовой переменной или параметра за пределы допустимого диапазона;
- ReferenceError - ошибка, возникающая при разыменовывании недопустимой ссылки;
- SyntaxError - ошибка, возникающая при разборе исходного кода в функции `eval()`;
- TypeError - ошибка, возникающая при недопустимом типе для переменной или параметра;
- URIError - ошибку, возникающую при передаче в функции `encodeURI()` или `decodeURI()` недопустимых параметров.

Типы ошибок

Тип Error - это базовый тип, от которого наследуются все остальные типы ошибок. Таким образом, все они имеют набор общих свойств

Оператор «throw»

Инструкцию try-catch дополняет оператор throw, с помощью которого в любое время можно сгенерировать собственную ошибку. Он применяется со значением, но не налагает никаких ограничений на тип значения.

```
throw 12345;

throw "Hello world!";

throw true;

throw { name: "JavaScript"};
```

Оператор «throw»

Оператор throw немедленно останавливает выполнение кода, которое возобновляется только в том случае, если инструкция try-catch перехватывает сгенерированное им значение.

Ошибки браузера можно имитировать, используя один из встроенных типов. Кон структор каждого типа ошибки принимает как единственный аргумент сообщение об ошибке, например:

```
throw new Error("Something bad happened . " ) ;
```

Задача 2

Создайте метод find_index(arr, value), который принимает список(массив значений) и значение. Если значение не найдено в списке нужно сгенерировать ошибку.

Вывод сообщений на консоль

- error(сообщение) - выводит на консоль сообщение об ошибке;
- info(сообщение) - выводит на консоль информационное сообщение;
- log(сообщение) - выводит на консоль сообщение общего характера;
- warn(сообщение) - выводит на консоль предупреждение.

Задача 3

Функция для оценки строки: вы должны проверить любой код HTML (т. Е. какие-либо теги HTML), если какой-либо код найден, вы должны вернуть false, если входные данные не являются строкой, вы должны выбросить ошибку TypeError. Если длина строки превышает 255 символов или содержит 0 символов, вы должны бросить RangeError и, наконец, если введенная строка пуста, выведите ReferenceError.

используйте `regex = /(<([^>]+)>)/ig` и метод `test()`

IT SHATLE

Задача 4

Создать поле ввода и кнопку, на нажатие кнопки должна вызываться функция сложения введенных значений и должен появляться результат вывода на экране. Если поля ввода пустое - кнопка неактивна. Если пользователь вводит не числовое значение - генерировать ошибку - TypeError