Обработка исключений

№ урока:	18	**Курс:**	JavaScript Базовый

Средства обучения:	Visual Studio Code Web Browser

Обзор, цель и назначение урока

Изучить понятие исключений. Научиться использовать операторы try, catch и finally для обработки ошибок этапа выполнения.

Изучив материал данного занятия, учащийся сможет:

- Понимать, что такое исключение.
- Понимать разницу между ошибкой этапа выполнения и синтаксической ошибкой.
- Работать с оператором try catch.
- Использовать оператор finally.
- Использовать оператор throw для генерации пользовательских исключений.
- Создавать свои типы исключений.

Содержание урока

1. Что такое исключение
2. Конструкция try catch
3. Блок finally
4. Использование throw
5. Создание пользовательских исключений

Резюме

- Исключение - ситуация, возникающая из-за состояния внешних данных, устройств или других систем, при которой дальнейшее выполнение приложения является бессмысленным.

- Например, попытка десериализовать объект из JSON строки, которая содержит ошибку приведет к исключению. Так как функция JSON.parse должна вернуть объект, но не может этого сделать из-за неправильных входных данных, она выбрасывает исключение (генерирует ошибку), которая останавливает дальнейшее выполнение сценария. Если parse не может вернуть данные, нет смысла в выполнении дальнейшего кода, только если разработчик не предусмотрит альтернативный алгоритм и обработает исключительную ситуацию.

- Если исключение генерируется в коде, находящемся в блоке try, то исключение прекращает выполнение всех последующих инструкций в блоке try и переносит выполнение в блок catch. При этом информация о возникшей ошибке передается в качестве параметра в блок catch. После блока catch сценарий продолжает выполнятся. Если исключение возникает за пределами блока catch – прекращается выполнение всего сценария.

CyberBionic systematics

CyberBionic Systematics ®
19 Evgenia Sverstiuka Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Объект исключения содержит три основных свойства: name – название исключения, message – описание возникшей ошибки, и stack – список вызовов, которые привели к исключению.

- Основные конструкторы ошибок:

  Error – ошибка во время выполнения
  EvalError – ошибка во время выполнения функции eval
  RangeError – значение выходит за определенный диапазон
  ReferenceError – обращение к несуществующей переменной
  SyntaxError – синтаксическая ошибка при исполнении кода в функции eval
  TypeError – недопустимый тип для переменной или параметра
  URIError – недопустимые параметры для функции encodeURI и decodeURI

- **Finally** - блок кода, который гарантировано выполняется. Finally может следовать сразу же за блоком try или за блоком catch. Если при выполнении происходит исключение в блоке try, срабатывает блок catch и после этого выполняется блок finally. Если в блоке try не произошло исключение, блок catch игнорируется, но finally все равно выполняется. Если в блоке catch произойдет еще одно исключение или до блока finally выполнится оператор return - блок finally все равно выполнится.

- Finally используют для тех случаев, когда нужно гарантировать выполнение действия при любых обстоятельствах. Например, когда это операция, связанная с освобождением ресурсов, которые были выделены во время выполнения блока try.

- **Throw** оператор, использующийся для создания исключения. Если этот оператор выполняется в коде, созданное исключение по стеку вызова передается до тех пор, пока не попадет в блок catch или пока не закончатся адреса в стеке вызовов. Данный оператор есть смысл использовать для библиотечных функций, которые используются в разных частях программы и должны уведомлять разработчика о неправильном использовании.

- Для создания пользовательского типа исключения, нужно определить новый класс и расширить один из системных классов исключений. По необходимости определить дополнительные свойства и проинициализировать свойства, полученные по наследству от класса родителя.

- После ключевого слова throw может использоваться любой объект или простое значение, который будет служить в качестве исключения. Но браузер и инструменты для работы с исключениями могут дать больше информации и будут удобными в использовании, если объект ошибки будет производным от встроенного класса ошибки.

Закрепление материала

- Что такое исключение? Приведите пример, когда в приложении возникает исключение.
- Как работает блок try/catch?
- Назовите основные свойства объекта исключения.
- Что такое finally? Опишите принцип работы данного блока.
- Как создать пользовательский тип исключения?

CyberBionic systematics

CyberBionic Systematics ®
19 Evgenia Sverstiuka Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Самостоятельная деятельность учащегося

Выполните задания в директории Exercises\Tasks\018 Try Catch. Текст задач, находится в комментариях в тегах script.

Рекомендуемые ресурсы

try/catch
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Statements/try...catch

Error и другие типы ошибок
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Error

CyberBionic Systematics ®
19 Evgenia Sverstiuka Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com