

Описание курса



JavaScript. Уровень 1. Основы JavaScript

Тема	Часы
Модуль 1. Основы программирования <i>Операторы, переменные, типы данных...</i>	4
Модуль 2. Управляющие конструкции <i>if-else, for, while, switch...</i>	4
Модуль 3. Функции <i>Синтаксис функции, аргументы, области видимости...</i>	4
Модуль 4. Объектные типы <i>Свойства, методы, массивы...</i>	4
Модуль 5. Объектно-ориентированное программирование <i>Функция конструктор, прототипы, классы...</i>	4
Модуль 6. Дополнительная информация <i>Работа со строками, регулярные выражения, JSON...</i>	4

Программа курса предусматривает лабораторные работы по каждой теме, а также выполнение домашних заданий и контроль знаний

Модуль 6 *Дополнительная информация*

Басов Денис



В этом модуле мы рассмотрим:

- ✓ Регулярные выражения
- ✓ Работа со строками
- ✓ Работа с JSON
- ✓ Другие встроенные объекты



Обработка ошибок: `try catch finally`

```
// стандартная ошибка
addAlert("Добро пожаловать!");

// выводим текст ошибки сами
try {
    addAlert("Добро пожаловать!");
} catch (err) {
    throw "Ошибка";
}
```

Объект ошибки



```
try {  
  addAlert("Добро пожаловать!");  
} catch (err) {  
  // выводим ошибку в консоль  
  console.log(err.name);  
}
```



Блок `finally` выполняется в любом случае

```
try {  
  addAlert("Добро пожаловать!");  
} catch (err) {  
  console.log("Ошибка");  
} finally {  
  // выполнится вне зависимости от наличия ошибки  
  console.log("test");  
}
```



Регулярные выражения, RegExr

Регулярные выражения — формальный язык, используемый в компьютерных программах, работающих с текстом, для поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов.

Для поиска, замены используется строка-образец, состоящая из символов и метасимволов и задающая правило поиска.

Поиск по строке с помощью регулярных выражений



```
/**
 * Поиск: str.match
 */

// Получение всех совпадений по строке в виде массива
let str = "Беги, Форест, беги!";
console.log(str.match(/беги/gi));
// ['Беги', 'беги']

// Получение первого совпадения с доп информацией
console.log(str.match(/беги/i));
// ['Беги', index: 0, input: 'Беги, Форест, беги!', groups: undefined]
```


Замена в строке с помощью регулярных выражений



```
/**
 * Замена: str.replace
 */

let str = "Я изучаю JS, мне нравится JS";

// без флага g
console.log(str.replace(/JS/i, "Java"));
// Я изучаю Java, мне нравится JS

// с флагом g
console.log(str.replace(/JS/gi, "Java"));
// Я изучаю Java, мне нравится Java
```



Проверка строки на соответствие регулярному выражению

```
/**
 * Проверка: regexp.test
 */

let str = "Я люблю мороженое";
let regExp = /люблю/i;

console.log(regExp.test(str)); // true

regExp = /шоколад/i;
console.log(regExp.test(str)); // false
```



Основные управляющие символы

- ✓ \wedge Ограничитель начала строки
- ✓ $\$$ Ограничитель конца строки
- ✓ $[a-y]$ ОДИН символ из указанного диапазона
- ✓ $[a-y]^+$ Один или более символов из диапазона
- ✓ $[a-y]^*$ Ноль или более символов из диапазона
- ✓ $[a-y]\{n\}$ Символ повторяется n раз
- ✓ $[a-y]\{min, max\}$ Символ повторяется от min до max



Базовые механизмы работы со строками

```
// Поиск длины строки
let browserType = "mozilla";
browserType.length; // 7

// Получение определённого строкового символа
browserType[0]; // m

// получение последнего символа строки
browserType[browserType.length - 1]; // a
```



Методы для работы со строками.

Поиск в строке и копирование части строки

```
let browserType = "mozilla";  
  
// Поиск подстроки внутри строки  
browserType.indexOf("zilla"); // 2  
  
// копирование части строки  
browserType.slice(0, 3); // moz  
browserType.slice(2); // zilla
```

Методы для работы со строками.

Изменение регистра строки и замена части строки



```
// Изменение регистра
let userData = "My NaMe Is BuD";

console.log(userData.toLowerCase()); // my name is bud
console.log(userData.toUpperCase()); // MY NAME IS BUD

// Обновление части строки
userData.replace("BuD", "Bob"); // My NaMe Is Bob
```



Методы для работы со строками.

Разбивка строки на массив по разделителю

```
// Разбивка строки на массив
```

```
let str = "The quick brown fox jumps over the lazy dog.";
str.split(" "); // ['The', 'quick', 'brown', 'fox', 'jumps', 'over',
'the', 'lazy', 'dog.']
```

```
let userName = "SuperAdmin999";
console.log(userName.split("")); // ['S', 'u', 'p', 'e', 'r', 'A',
'd', 'm', 'i', 'n', '9', '9', '9']
```

```
let orderDate = "19.01.2024 12:54:12";
console.log(orderDate.split(" ")); // ['19.01.2024', '12:54:12']
```

Что такое JSON

JSON (англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript. Но при этом формат независим от JS и может использоваться в любом языке программирования.

В качестве значений могут быть:

- ✓ Объекты
- ✓ Массивы
- ✓ Числа
- ✓ Логические значения, null
- ✓ Строки



JSON структура данных



```
{  
  "firstName": "Иван",  
  "lastName": "Иванов",  
  "address": {  
    "streetAddress": "Московское ш., 101, кв.101",  
    "city": "Ленинград",  
    "postalCode": 101101  
  },  
  "phoneNumbers": [  
    "812 123-1234",  
    "916 123-4567"  
  ]  
}
```

Преобразование данных в формат JSON



```
let myCat = {  
  name: "Барсик",  
  color: "Серый",  
  age: 6,  
  avatar: "img/2.jpg",  
  owner: {  
    name: "Анна",  
    city: "Москва",  
    phones: [523, 653, 532],  
  },  
};  
console.log(JSON.stringify(myCat));
```

```
{"name": "Барсик" script.js:23  
  , "color": "Серый", "age": 6, "avatar": "img/2.jpg", "owner":  
  {"name": "Анна", "city": "Москва"  
  , "phones": [523, 653, 532]}}
```



Декодирование JSON-строки

```
// Кодируем в строку JSON
let myJSONCat = JSON.stringify(myCat);
console.log(myJSONCat);
// {"name":"Барсик","color":"Серый","age":6,"avatar":"img/2.jpg",
"owner":{"name":"Анна","city":"Москва","phones":[523,653,532]}}

// Раскодируем из строки JSON
console.log(JSON.parse(myJSONCat)); // {name: 'Барсик', color:
'Серый', age: 6, avatar: 'img/2.jpg', owner: {...}}
```



Получение данных через REST API

<https://jsonplaceholder.typicode.com/guide/>

```
// получение данных с помощью запроса на другой сайт
// https://jsonplaceholder.typicode.com/guide/
async function getApiData() {
  let response = await fetch(`https://jsonplaceholder.typicode.com/
  posts/1`); // делаем запрос, сохраняем ответ в переменной

  let data = await response.json(); // раскодируем
  console.log(data);
}

getApiData();
```



Получение данных через REST API

<https://restcountries.com/>

```
// получение данных с помощью запроса на другой сайт
// https://restcountries.com/
async function getApiData() {
  let response = await fetch(`https://restcountries.com/v3.1/name/
deutschland`); // делаем запрос, сохраняем ответ в переменной

  let data = await response.json(); // декодируем
  console.log(data);
}

getApiData();
```

Спасибо
за внимание!
Ваши вопросы...

