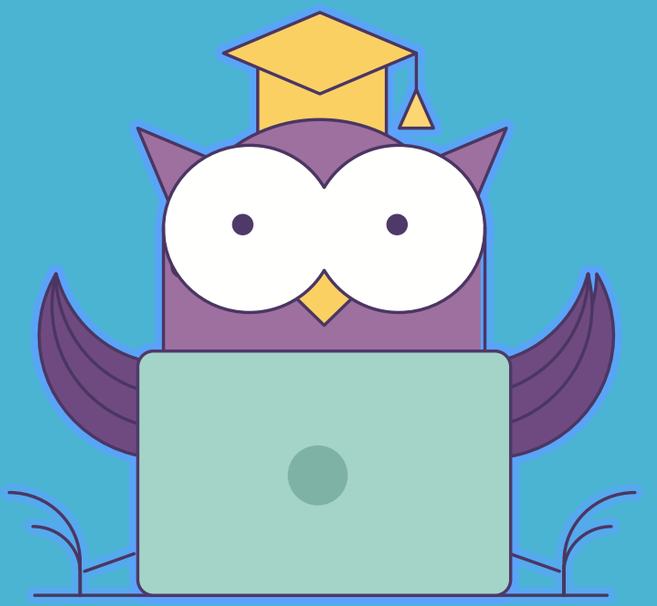
ОНЛАЙН-ОБРАЗОВАНИЕ

Введение в курс

Александр Коржиков

Как меня слышно и видно?



> Напишите в чат

+ если все хорошо

– если есть проблемы со звуком или с видео

Цели занятия

- Познакомиться с преподавателем и с программой курса, понимать как она построена
- Вспомнить и применять основные техники языка **JavaScript**, которые помогут при изучении фреймворков

- Расскажите о себе 😀 , можно в слаке
  - Кто Вы?
  - Какой Ваш опыт в программировании?
  - Какие **JavaScript** фреймворки Вы используете?
  - Какие **JavaScript** библиотеки Вы используете?
  - Что Вы ждете от курса?

Для отправки сообщений в Zoom используйте

All panelists and attendees

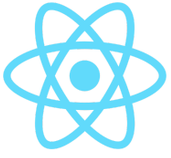
Ведущие и участники


JS


node
JS


npm


express




A


TS

Цели курса

- Понимать для чего нужны веб фреймворки
- Уметь применять паттерны проектирования (Virtual DOM, Dependency Injection, Observables, Event Loop)
- Знать подходы при проектировании, организации, контроля состояния и тестирования приложений
- Выбирать подходящие инструменты для проекта
- Создавать веб-приложения с использованием современных технологий

- Тема “Modern JavaScript Frameworks”
- 38 занятия / 1.5 часа / понедельник / четверг
- Вебинары, материалы, теория и практика
- Домашние задания
- Выпускной проект
- Общение в slack

- **Александр Коржиков**

JavaScript @ Alpari, Comindware, Tinkoff, Backbase, ING
korzio@gmail.com https://github.com/korzio

- **Юрий Дворжецкий**

Java, JavaScript @ Luxoft
Teaching @ 1500+ hours, 600+ developers

- **Владимир Клепов**

FullStack @ Яндекс, Российские Суперкомпьютеры
v.klepov@gmail.com https://thoughtspile.github.io

- **Михаил Кузнецов**

FullStack @ ING https://github.com/shershen08

Программа курса

Q&A

- Типы данных
- Переменные
- Функции
- Замыкания

ECMAScript 6 (2015)

- http://es6-features.org/

  (кратко)
- http://exploringjs.com/es6/

  (подробно)
- https://developer.mozilla.org/

  (справочник)
- http://javascript.ru/

  (Илья Кантор)

JavaScript динамически типизированный язык

Примитивные типы данных

- number
- string
- boolean
- null
- undefined
- *Symbol*

- number **1, 2, 3.5, 0x8? , 0b11?**
- string '1' , "2", `oh`, my`fn`
- boolean **true ll false**
- null
- undefined // === undefined ?
- *Symbol - Вы знаете, для чего они нужны и как их использовать?*

```
var n = 4
n.a = 5
n.a // ?
```

Все другие типы данных в **JavaScript** - объекты

Objects

```
typeof null === // 1 ?
typeof {} === // 2 ?
typeof function === // 3 ?
typeof typeof null === // 4 ?
```

Может ли это выражение быть инстинным?

```
a == 1 && a == 2 && a == 3
=== true // ?
```

- **valueOf()** возвращает простое представление объекта
- **toString()** возвращает строковое представление объекта

```
a == 1 && a == 2 && a == 3
=== true // ?

var a = (function() {
  var i = 1
  return {
    valueOf: function() {
      return i++
    }
  }
})()
```

Области видимости

- **var** доступны во всем теле функции
- **let** и **const** определены в блоке **{...}**

```
if (true) {
  var b = 1
}

console.log(b) // 1

if (!false) {
  const c = 2
}

// console.log(c)
// ReferenceError: c is not defined
```

```
var x = "global"
function f() {
  var x = "local"
  console.log(x) // ?
}
f()
console.log(x) // ?
```

- **var** обрабытывается до исполнения кода **<=> hoisting**
- **let** и **const** определены в блоке **{...}**

```
console.log(b) // undefined

// console.log(c)
// ReferenceError: c is not defined
var b = 1
const c = 2
```

Определение и присвоение переменных из объектов

```
function h ({ name, val }) {
  console.log(name, val)
}

var {
 op: a, lhs: { op: b } = {}
} = {} // A
```

Вопрос

Какая может быть структура у объекта $\mathbf{A}$, чтобы $\mathbf{a}$ и $\mathbf{b}$ были определены?

- Будет ли это работать?
- Что здесь происходит?

```
var a = { a }
let b = { b }
const c = { c }
```

```
var a = { a }
// Ok!

let b = { b }
// Uncaught ReferenceError:
// b is not defined
const c = { c }
// Uncaught ReferenceError:
// c is not defined
```

- **var** (hoisting + shorthand)
- **let**
- **const**

- Правильное именование переменных
- Декларация переменных до их использования

```
"use strict";
```

- Не публиковать локальные переменные
- **const** --> **let** -?-> **var**

- Этот код будет работать?
- Какие из примеров являются
  - **function expression**,
  - **function declaration**,
  - **function constructor**?
- Что такое IIFE?

```
function a() {} // 1

new Function('a, b', 'return a')
// 2

const a = function () {} // 3
const a = () => {} // 4
```

Отличия

- Контекст исполнения **this**
- Arguments
- Constructor

Что будет выведено в консоль?

```
var obj = {
  a: function (){
    console.log(this.prop)
  },
  prop: 1
}

obj.a.prop = 2
obj.a() // ?
var fn = obj.a
fn() // ?
```

Область видимости определяется функцией

```
function outer() {
  var outerVar

  var func = function() {
    var innerVar = innerVar + outerVar
  }

  return func
}
```

- Что будет в консоли?
- Как это исправить?

```
for (var i = 0; i < 10; i++) {
  setTimeout(function() {
    console.log(i)
  }, 1000)
}
```

let

```
for (let i = 0; i < 10; i++) {
  setTimeout(function() {
    console.log(i)
  }, 1000)
}
```

bind

```
for (var i = 0; i < 10; i++) {
  setTimeout(function() {
    console.log(this)
  }.bind(i), 1000)
}
```

```
for (var i = 0; i < 10; i++) {
  setTimeout(function() {
    console.log(i)
  }.bind(null, i), 1000)
}
```

Какой контекст будет у функции, если к ней применить **bind()** дважды?

```
var fn = function() {
  console.log(this)
}

var a = fn.bind(1)
a() // a?

var b = a.bind(2)
b() // b?
```

Почему?

Реализовать функцию **reduce** - **polyfill** для **Array.prototype.reduce**

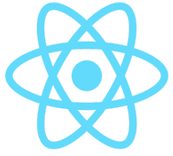
```
// example
[[0, 1], [2, 3], [4, 5]].reduce((memo, currentValue) => {
  return memo.concat(currentValue)
}, [])

reduce(
 [1, 2, 3, 4], // arguments
 (a, b) => a + b, // action
 0 // initial value
) // 10
```

Q&A

- Познакомились с преподавателем и с программой курса
- Вспомнили теорию - типы данных и переменные, функции и замыкания
- Попрактиковали различные **JavaScript** техники


JS

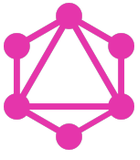

node
JS




npm


TS


express



- https://github.com/
- single repo lastname-name(-otus) - **korzhikov-alex-otus**
- **master** branch
- MR from **block-lesson-number** to **master** (**javascript-1** -> **master**, **react-4** -> **master**)
- exclude **node_modules**, **bower_components**, **editor**, scaffolding
- https://github.com/korzio/modern_javascript_frameworks
- Добавить ссылку на MR в чат с преподавателем внутри ЛК Отуса

Написать функцию sum, которая может быть исполнена множество раз.

Если она исполнена без аргументов, то возвращает значение суммы всех переданных до этого значений.

```
sum(1)(2)(3)....(n)() === 1 + 2 + 3 + ... + n
```

Пожалуйста, пройдите **опрос**

в личном кабинете

- Все ли темы были понятны? (да - нет)
- Легкий материал или нет? (1 просто - 10 сложно)