Разработчик JavaScript

Полный курс по JavaScript для web-разработчиков, которые хотят вывести свои навыки программирования на новый профессиональный уровень

Длительность курса: 160 академических часов

1 JavaScript

1	**Введение в курс Modern JavaScript Frameworks**	Участники смогут: - Познакомиться с преподавателем и с программой курса, понимать как она построена и какие полезные навыки они получат - Вспомнить основные возможности языка JavaScript - Применять техники языка, которые помогут при изучении фреймворков Домашние задания 1 Написать функцию суммирования значений Написать функцию sum, которая может быть исполнена любое количество раз с не `undefined` аргументом. Если она исполнена без аргументов, то возвращает значение суммы всех переданных до этого значений. sum(1)(2)(3)....(n)() === 1 + 2 + 3 + ... + n
2	**Возможности современного JavaScript**	участники смогут: - Решать специфичные для браузерной разработки задачи на языке JavaScript - Освоить и вспомнить теорию, которая будет базисом для последующих уроков - Попрактиковаться с технологиями AJAX, WebSocket, Promise

Домашние задания

1 promiseReduce - работа с асинхронными функциями

Цель: Написать функцию promiseReduce(asyncFunctions, reduce, initialValue) asyncFunctions - массив асинхронных функций, возвращающих промис reduce(memo, value) - функция, которая будет вызвана для каждого успешно завершившегося промиса. initialValue - стартовое значение для функции reduce promiseReduce последовательно вызывает переданные асинхронные функции и выполняет reduce функцию сразу при получении результата до вызова следующей асинхронной функции. Функция promiseReduce должна возвращать промис с конечным результатом.

Пример использования

```javascript
var fn1 = () => {
console.log('fn1')
return Promise.resolve(1)
}

var fn2 = () => new Promise(resolve => {
console.log('fn2')
setTimeout(() => resolve(2), 1000)
})

function promiseReduce(asyncFunctions, reduce, initialValue) {
/*
* Реализация
*/
}

promiseReduce(
[fn1, fn2],
function (memo, value) {
console.log('reduce')
return memo * value
},
1
)
.then(console.log)
```

Вывод в консоль

```
fn1
reduce
fn2
```

reduce
2
```

---

3 **Введение в Node - Пакетный менеджер NPM и возможности package.json**

Участники смогут:
- Запускать приложения на платформе Node
- Писать и запускать тесты для серверного JavaScript
- Работать с пакетным менеджером NPM
- Управлять зависимостями и автоматизировать задачи с помощью package.json

Домашние задания

1 Реализовать скрипт request для тестирования веб сервера

Создать локальный веб сервер `server`, отвечающий на запросы каждые 100ms

Создать скрипт `request`, принимающий на вход
- количество запросов `N`
- тип запросов - параллельный или последовательный

Скрипт `request` должен отправлять `N` последовательных или параллельных `HTTP` запросов к локальному серверу `server`

---

4 **Test Driven Development с JavaScript**

Обзор фреймворков и библиотек для тестирования
Техники тестирования
Behavior Driven Development
Разбор примеров

---

5 **Основные концепции Node - Modules**

Участники смогут:
- Использовать require, exports и ES6 Imports для экспорта и импорта зависимостей

---
```

6 **Стандартная библиотека Node - EventLoop - Timers**

Участники смогут:
- Ориентироваться в понятии EventLoop и особенностях работы Timers
- Использовать классы, объекты и функции модуля Streams
- Работать с HTTP запросами в Node

Домашние задания

1 tree - вывод списка файлов и папок файловой системы

Напишите `NodeJS` скрипт `tree` для вывода списка файлов и папок файловой системы.
Результатом работы должен быть объект с массивами `{ files, folders }`.
Вызовы файловой системы должны быть асинхронными.
Скрипт принимает входной параметр - путь до папки.
Добавить возможность выполнять этот скрипт через команду `npm run tree -- path`

Пример

```
foo/
├── bar/
│ ├── bar1.txt
│ ├── bar2.txt
│ └── baz/
├── f1.txt
└── f2.txt
```

При вызове с путем `foo/` скрипт должен вернуть структуру:

```json
{
"files": [
"foo/f1.txt",
"foo/f2.txt",
"foo/bar/bar1.txt",
"foo/bar/bar2.txt"
],
"dirs": [
"foo",
"foo/bar",
"foo/bar/baz"
]
}
```

7 **Node Best Practices - Streams - Errors - Processes**

- Работать с дочерними процессами в Node
- Различать корректные и ошибочные техники при написании серверного JavaScript кода
- Node Best Practices
- Streams
- Node Summary

Домашние задания

1 Работа с потоками в NodeJS*

* - задача со звездочкой. Сдается при желании.

Необходимо отсортировать большой файл со случайными целыми числами, размером 100 МБ, в условиях ограниченной оперативной памяти - 50 МБ. Решение должно быть построено с использованием потоков.

Для решения задачи можно использовать алгоритм “Сортировка слиянием”.
Процесс можно разделить на 3 этапа.

Этап 0
Любым удобным вам способом создаем исходный файл с числами размером 100 МБ.

Этап 1
Исходный файл с числами необходимо разбить на несколько файлов поменьше, предварительно отсортировав их независимо друг от друга.

Этап 2
Необходимо создать механизм чтения чисел сразу из нескольких файлов (потоков).
Читать данные из потоков следует по принципу pause/resume.

Этап 3
Необходимо создать цикл, который будет работать с данными сразу из всех потоков.
Такой цикл будет прерван только тогда, когда будут полностью прочитаны все файлы.
В цикле следует искать наименьшее значение и записывать его в итоговый файл.
1 итерация = 1 число

Для проверки решения, скрипт необходимо запустить командой
$ node --max-old-space-size=50 script.js

8	**Web-сервер на JavaScript - Стек MEAN - Express - MongoDB**	Участники смогут: - Создавать простые приложения с использованием библиотеки Express, а также базы данных MongoDB
9	**Построение Rest API**	Участники смогут: - Понимать и добавлять общие стандарты создания API

10 **GraphQL Server**

Домашние задания

1 Домашняя работа для `Занятие "GraphQL Server"`

На выбор одна из следующих задач:

Часть 1.
Написать схему GraphQL для примера веб-приложения e-commerce shop:
до 3 балла - какие сущности (минимум 3, можно больше), какие у них поля, какие обязательные какие нет
до 4 баллов - какие запросы/мутации понадобятся (минимум 4, можно больше)

Часть 2.
до 5 баллов - развернуть локально graphQL + nodejs или воспользоваться одним из веб демо (graphqlbin), перенести полностью или частично написанную в Части 1 схему.
Результатом работы будет ссылка на онлайн демо или репозиторий.

// ИЛИ

Написать `NodeJS Rest API` приложение для сохранения `RSS` рассылок.
В приложении должно быть следующие точки доступа
- Создание рассылки по `URL`. При успешном добавлении приложение будет запрашивать `RSS` рассылку, парсить `XML` и сохранять документы в базу данных.
- Показ списка всех добавленных `URL` рассылок.
- Показ всех сохраненных из `RSS` документов.

Приложение должно содержать тесты для всех точек доступа.

---

11 **WebAssembly**

- Понимать и использовать технологию WebAssembly
- Понимать архитектуру JavaScript движка на примере V8

---

12 **JavaScript - Работа с браузером**

- Решать специфичные для браузерной разработки задачи на языке JavaScript
- Работать с Chrome Dev Tools

Домашние задания

1 getPath - поиск уникального селектора

Написать алгоритм и функцию `getPath()`, находящую уникальный css-селектор для элемента в документе. Уникальный селектор может быть использован `document.querySelector()` и возвращать исходный элемент. `document.querySelectorAll()`, вызванный с этим селектором, не должен находить никаких элементов, кроме исходного.

```javascript
$0 // HTMLElement
getPath($0) // => "..."
```

2 UI Libraries

1 **Обзор Web Components - HTML Template - Polymer - Lit-HTML**

Участники смогут
- Создавать custom elements, используя встроенные браузерные возможности
- Подключать и использовать Polymer для создания приложений
- Ориентироваться в веб спецификациях, на базе которых работает Polymer

---

2 **Custom Elements**

Подключать и использовать веб-компоненты в приложениях
Создавать Polymer компоненты, решающие функциональные и композиционные задачи

---

3 **Shadow DOM**

Домашние задания

1 Custom Elements Tree

С помощью Custom Elements создать приложение для показа дерева с помощью компонентов my-tree и my-leaf. Компоненты должны получать данные о структуре поддерева от родительского элемента. Используйте Shadow DOM при отрисовке компонент.

Пример структуры

```
{
"id": 1,
"items": [{
"id": 2,
"items": [{ "id": 3 }]
}]
}
```

---

4 **Основы React и JSX**

Участники смогут:
- Настроить себе окружение для работы с React и использовать его
- Понимать и применять синтаксис JSX
- Создавать простые приложения на React

---

5 **Компоненты React - Lifecycle React-компонент - state и props**

Участники смогут:
- Разрабатывать полноценные React-компоненты в различных стилях.
- Корректно использовать state и props.

Домашние задания

1 Создать структуру приложения погоды

Приложение для самостоятельной работы в блоке React - веб-приложение погоды.
На странице приложения должна быть возможность добавлять города в список избранных.
По каждому городу показывается информация о температуре, ветре, другие параметры.

Создать структуру приложения, создать компоненты контейнеры.

---

6 **Higher-Order Components**

---

7 **Состояние приложения - Flux и Redux**

Участники смогут:
- Отличать основные понятия однонаправленной архитектуры Flux.
- Ориентироваться и использовать возможности redux - создавать actions, reducers, а также применять redux в связке в React

---

8 **Routing в React - Оптимизация приложения**

- Создавать систему routing для React приложений, использовать библиотеку react-router
- Использовать специальные возможности библиотеки для оптимизации отрисовки

Домашние задания

1 Routing для приложения погоды

Реализовать компонент фильтра и поиска городов.
Данные по городам сохранять в браузерном хранилище.
Добавить страницу погоды по конкретному городу.
При переходе на нее должен меняться url, показываться информация на несколько дней вперед.

9	**Подготовка React Приложения к Production, Best Practices**	- Эффективно разрабатывать приложения на React, учитывая последние тенденции в разработке front-end - Использовать Advanced React - Применять на практике Best-Practices разработки на React - Сборщики - Webpack, Parcel - Аспекты Server-Side Rendering
10	**Основы Vue**	- Настроить себе окружение IDE, зависимости и библиотеки для создания проектов и работы с Vue - Создавать простейшие приложения используя Vue
11	**Компоненты, шаблонизатор и формы**	Участники смогут - Понимать синтаксис шаблонизаторы - Создавать компоненты, описывать атрибуты элементов Домашние задания 1Структура приложения "Устный счет" В разделе Vue одна большая самостоятельная работа - SPA (Single Page Application) игра "Устный счет". Игра состоит из двух экранов - на первом экране пользователь выбирает настройки, которые будут использовать в игре - типы вычислений, сложность, время раунда. На этой же странице показывается статистика тренировок. Вторая страница - сама игра. Пользователь должен решить максимальное количество задач на заданное время. Мокапы - https://app.moqups.com/korzio@gmail.com/bTYyBLCtpU/edit/page/ad64222d5 --- Подготовить общую структуру приложения - компоненты контейнеры для страниц приложения. Сделать первую страницу приложения - форму настроек.
12	**Routing и модели данных**	Участники смогут: - Описывать routing для Vue приложений - Создавать формы, связывая шаблоны с моделями

13 **Advanced Vue - Vuex**

Участники смогут:
- Применять анимацию в компонентах
- Создавать плагины
- Разбираться в тонкостях Change Detection

Домашние задания

1 Routing для приложения "Устный счет"

Реализовать второй экран - игру "калькулятор". Настроить переходы по страницам приложения.

---

14 **Специфика построений приложений с Vue, Best Practices**

работа с анимацией во Vue
Unit тестирование
Server-Side Rendering с Nuxt
Custom Directives во Vue

Домашние задания

1 Добавление фич в open source проект

На каждый поток мы выбираем репозиторий с каким то проектом или плагином и развиваем его силами студентов курса.

3 Angular

1	**Введение в Angular**	Участники смогут: - Настроить себе окружение IDE, а также скачать зависимости и библиотеки, коммандные утилиты для TypeScript и создания проектов для работы с Angular - Различать TypeScript и JavaScript - Писать и понимать код на языке TypeScript
2	**TypeScript**	- Различать TypeScript и JavaScript, использовать преимущества статической типизации - Писать и понимать код на языке TypeScript, разрабатывать приложения в полноценном объектно-ориентированном стиле

3 **Компоненты и директивы**

Участники смогут:
- Декомпозировать макет страницы на компоненты
- Различать директивы и компоненты во фреймворке Angular
- Создавать простые директивы и компоненты

Домашние задания

1 Структура приложения для запоминания иностранных слов

Приложение для запоминания иностранных слов.
В этом приложении пользователь сможет добавлять слова для изучения, проходить тесты для запоминания слов.
Это Single Page Application состоит из 3 страниц:
- Последние добавленные слова (Recently Added)
- Упражнениями (Go)
- Настройки (Settings)
На главном экране, на странице Recently Added пользователь видит список последних добавленных слов, может добавить новое слово в словарь.

На странице упражнений пользователь занимается тестированием своих знаний. Ему показывается слово на одном языке, и он должен написать его перевод на другой язык. Если перевод правильный, слово засчитывается, иначе показываем ошибку. Мы начнем с двух языков - русского и английского, будем расширять возможности приложения по мере написания программы.

На странице настроек пользователь выбирает языки, количество слов в упражнении, отводимое на упражнение время.

Навигация по страницам происходит с помощью ссылок в верхней части страниц, каждой странице соответствует отдельный url.

Декомпозировать приложение для запоминания иностранных слов.
Создать структуру и компоненты контейнеры приложения.

---

4 **Сервисы**

Участники смогут создавать сервисы для получения, отправки и хранения данных для приложений Angular

---

5 **Observables - RxJS**

Участники смогут:
- Отличать основные понятия паттерна
- Observable, Observer, Subscriber, Operator
- Применять шаблон проектирования Observables, используя библиотеку RxJS

Домашние задания

1 Создать сервисы для работы с текстом

Создать сервисы для работы с текстом
- Сервис перевода слова - должен запрашивать перевод через API (например, https://tech.yandex.com/translate/)
- Сервис хранения словаря - небольшая обертка для управления словарем с помощью `localStorage`
- Сервис добавления слов - должен разбивать текст на отдельные слова, запрашивать их перевод и сохранять в словарь для приложения.

Сервисы должны общаться с помощью библиотеки `RxJS`.

---

6 **Состояние приложения**

Участники смогут разбираться в особенностях шаблона проектирования Dependency Injection и его имплементации в Angular

---

7 **Создание и управление формами в Angular**

Участники смогут:
- Создавать формы, используя техники Dynamic Forms, Reactive Forms
- Описывать валидацию и другие функции для элементов форм

Домашние задания

1 User Interface для приложения запоминания иностранных слов

Реализовать `UI` приложения
- Создать компоненты для добавления текста/слов в словарь
- Разработать компоненты и формы для тренировки запоминания слов
- Добавить экран настройку приложения, сохранять состояние

8	**Routing**	Участники смогут: - Создавать Routing систему для приложений, используя внутренние подходы Angular - такие как Router, router-outlet и другие - Понимать и применять хэндлеры навигации Guards
9	**Тестирование в Angular. Сборка приложения для Production**	Участники смогут: - Писать и запускать тесты для приложений Angular - Настраивать сборку приложений Angular - Использовать возможность сборки Server-side Rendering Домашние задания 1 Routing для приложения запоминания иностранных слов Добавить routing, ссылки на страницы и переходы между компонентами приложения. Добавить и актуализировать тесты для компонент приложения, настроить universal рендеринг приложения.

4 Проектная работа

1	**Вводное занятие по проектной работе. Обзор пройденных фреймворков и технологий.**	Участники смогут: - Определиться и обсудить тему проектной работы - Выделять характеристики проектов и окружения - Решать задачи выбора и сравнения фреймворков, понимать их преимущества и недостатки Домашние задания 1 Проектная работа Заключительный месяц курса посвящен проектной работе. Это то, чем интересно заниматься студенту на базе знаний, полученных на курсе. При этом не обязательно закончить его за месяц. В процессе написания по проекту можно получить консультации преподавателей. Проект должен стать примером кода, который можно показывать потенциальным работодателям.
2	**Консультация по проектным работам**	Обсудим цели и проектные работы
3	**Защита проектных работ**	- Подведение итогов модуля и курса в целом - Презентация выполненных проектов