JavaScript Developer. Professional

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

Длительность курса: 152 академических часа

1 JavaScript

1 Введение в курс Modern JavaScript Frameworks

Цели занятия:

после занятия вы сможете:
познакомиться с преподавателем и с программой курса, понимать как она построена и какие полезные навыки они получат;
объяснить основные возможности языка JavaScript;
применять техники языка, которые помогут при изучении фреймворков.

организационные вопросы по курсу;
типы данных;
переменные;
функции;
замыкания.

Домашние задания

1 Задача про список рекомендаций maxItemAssociation

Цель: Написать алгоритм для нахождения максимального списка рекомендаций.

Это задание является достаточно сложным, однако может быть интересно тем, кто хочет попробовать свои силы с реальными примерами из реальных собеседований.
Если вы хотите попробовать решить менее сложную задачу, внизу можно найти задание 2.
Решение любого из заданий достаточно для зачета этого ДЗ.

Задание 1.

Написать функцию maxItemAssociation(), получающую исторические данные покупок пользователей и возвращающую максимальный список рекомендаций.

Входные данные - массив исторических покупок пользователей [["a", "b"], ["a", "c"], ["d", "e"]]. То есть пользователь 1 купил "a" и "b". Пользователь 2 купил продукты "a", "c". Пользователь 3 купил продукты "d", "e".
Надо найти максимальную группу рекомендаций. Группа рекомендаций - это продукты, которые был куплены другими пользователями при условии, если они пересекаются с исходным списком.

Если количество рекомендаций в группах одинаковое - вернуть первую группу, из отсортированных в лексикографическом порядке.

Решение:
Группа рекомендаций 1 - ["a", "b", "c"]. Покупка "a" содержится в списке 2, поэтому весь список 2 может быть добавлен в рекомендации.
Группа рекомендаций 2 - ["d", "e"].

Ответ: ["a", "b", "c"].

Пример 2:

Входные данные: [
["q", "w", 'a'],
["a", "b"],
["a", "c"],
["q", "e"],
["q", "r"],
]

Ответ ["a", "b", "c", "e", "q", "r", "w"] - это максимальная по пересечениям группа. Можно видеть, что первый массив пересекается со всеми остальными, и потому результат является всем множеством значений.

Задание 2.

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**

Цели занятия:

после занятия вы сможете:
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.

наследование;
promise;
Async Patterns;
обзор ES6 Features.

---

3 **JavaScript - Работа с браузером**

Цели занятия:

после занятия вы сможете:
решать специфичные для браузерной разработки задачи на языке JavaScript;
работать с Chrome Dev Tools.

DOM API;
DevTools;
events.

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
```

4	**Введение в Node - Пакетный менеджер NPM и возможности package.json**	**Цели занятия:** после занятия вы сможете: 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. Node: About; пример Web сервера; структура; стандартные модули; примеры Callbacks; пакетный менеджер `npm`: возможности `package.json`; CLI.

5 **Test Driven Development**

Цели занятия:

после занятия вы сможете:
разбирать примеры.

обзор фреймворков и библиотек для тестирования;
техники тестирования;
behavior Driven Development.

Домашние задания

1 getPath - поиск уникального селектора

Цель: В ходе выполнения ДЗ студент напишет алгоритм и функцию `getPath()`, находящую уникальный css-селектор для элемента в документе.

Написать алгоритм и функцию `getPath()`, находящую уникальный css-селектор для элемента в документе.
Уникальный селектор может быть использован `document.querySelector()` и возвращать исходный элемент.

Так чтобы `document.querySelectorAll()`, вызванный с этим селектором, не должен находить никаких элементов, кроме исходного.

```javascript
$0 // HTMLElement
getPath($0) // => "body div.someclass ul li:first-child"
```

Использовать TDD, добавить юнит тесты для функции

2 Node.js

1 Основные концепции Node - Modules

Цели занятия:

после занятия вы сможете:
использовать require, exports и ES6 Imports для экспорта и импорта зависимостей.

modules;
pattern;
classic;
AMD;
commonJS;
ES Modules;
Native ES Modules - Gil Tayar.

---

2 Стандартная библиотека Node - EventLoop - Timers

Цели занятия:

после занятия вы сможете:
ориентироваться в понятии EventLoop и особенностях работы Timers;
использовать классы, объекты и функции модуля Streams;
работать с HTTP запросами в Node.

events;
event Loop;
timers.

Домашние задания

1. tree - вывод списка файлов и папок файловой системы

   Цель: В ходе выполнения ДЗ студент выведет список 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"
]
}
```

---

3 **Node Best Practices - Streams - Processes**

Цели занятия:

после занятия вы сможете:
работать с дочерними процессами в Node;
различать корректные и ошибочные техники при написании; серверного JavaScript кода.

streams: примеры и типы потоков, API, Pipe, Iterable Streams;

Домашние задания

1 Работа с потоками в NodeJS*

Цель: В ходе выполнения ДЗ студент поработает 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

---

4 **Web-сервер с Express**

Цели занятия:

- использовать стандартнтую библиотеку Node.js для веб сервера
- разбираться и понимать структуру и концепции Express - middlewares, templates, routing
- создавать приложения с использованием Express
- обрабатывать ошибки в Node.js приложениях

- стандартная библиотека Node.js
- модули HTTP(S), TCP
- Express: introduction, concepts, routing, middleware, template
- стэк MEAN
- обработка ошибок в Node.js
- Express error handlers

5	**Возможности MongoDB**	**Цели занятия:** - подключать и использовать различные дистрибутивы mongodb в node.js приложении - создавать и понимать концепции коллекций, документов и полей - писать CRUD-операции запросы к данным - понимать общие возможности фильтраций в mongodb (limit, skip, поддокументы) - Введение и обзор типов БД - NoSQL - Введение в MongoDB - обзор различных дистрибутивов mongodb - подключение и использование в node.js - коллекции, документы и поля - операции и запросы к данным
6	**Расширенные функции MongoDB - Aggregation Framework**	**Цели занятия:** - Понимать и использовать Aggregation Framework - map/reduce - Настраивать и администрировать MongoDB - концепция map-reduce - альтернатива pipeline - структура и синтаксис агрегационного фрейморка - синтаксис запросов $match, $group, $lookup - возможности настройки и администрирования Mongo - кластеры

7 **Построение Rest API с Express, Mongoose**

Цели занятия:

- добавлять общие стандарты создания API;
- объяснить REST, принципы построения API;
- Использовать ORM - Mongoose
- Реализовывать CRUD операции с Mongoose, Express и MongoDB
- проанализировать технологии аутентификации в веб приложениях.

- что такое REST?
- методы, коды ошибок;
- express, swagger;
- Обзор ORM - Mongoose, Sequelize, TypeORM
- Подключение и использование Mongoose
- express Middlewares для аутентификации и авторизации;
- сессии и куки, jwt;
- Обзор функций аутентификации, Passport.js.

---

8 **Основы GraphQL**

Цели занятия:

после занятия вы сможете:
работать с GraphQL и основными концепциями.

введение в GraphQL;
система типов;
операции, запросы, мутации;
обзор решений GraphQL.

Домашние задания

1 Домашняя работа для `Занятие "GraphQL Server"`

Цель: В этом ДЗ напишете схему GraphQL или `NodeJS Rest API`.

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.

---

9 **Возможности GraphQL в реальности**

Цели занятия:

после занятия вы сможете:
объяснить про специальные возможности GraphQL;
построить сложные схемы и запросы.
проанализировать сложности: нагрузка, безопасность, N+1.

кратко о GraphQL;
apollo Client (vue);
проблемы GraphQL - проблема N+1;
тестирование GraphQL.
кеширование.

---

10 **Сборка и деплой проекта, CI/CD**

Цели занятия:

понимать и использовать общие концепции CI/CD
определять требования к серверу
настраивать деплоймент приложений в облако

Основы CI
- Обзор CI/CD
- переменные окружения
- Gitlab CI/CD / Heroku

- MongoDB в облаке
- мониторинг / Prometheus
- запросы к логам
- запускаем Node.js в Docker
- Docker-compose на бою

Подводные камни запуска проекта на сервере
- Выстраивание подключения к БД
- Вопросы безопасности

- На что в первую очередь смотреть при запуске проекта

3 Современный Стэк JavaScript

1 Web Components

Цели занятия:

после занятия вы сможете:
ориентироваться в веб спецификациях Custom Elements и Shadow DOM;
создавать custom elements, используя встроенные браузерные возможности;
использовать HTML Template для шаблонизации компонент.
использовать особенности Lit-HTML для создания приложений;
объяснить принципы Polymer;

Custom Elements:
Standalone Elements;
Built-in Elements;
LifeCycle Hooks.
Shadow DOM:
DOM;
Slots;
Styles.
Web Standards:
HTML Template;
Script Type Module;
ES6 Templates Literals.
Lit-HTML:
Lit-Element;
pwa-starter-kit;

---

2 **Webpack**

Цели занятия:

после занятия вы сможете:
настраивать Webpack;
различать Parcel и Rollup;
понимать что такое и как использовать babel;
использовать eslint

modern JavaScript Frameworks;
системы сборки и тестирование

Домашние задания

1 Custom Elements Tree

Цель: В ходе выполнения ДЗ студент создаст приложение для показа дерева.

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. Можно также использовать для реализации Lit-Element, Lit-HTML или Polymer.

Пример структуры

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

---

3 **PWA**

Цели занятия:

после занятия вы сможете:
понимать и использовать подход PWA.

введение в функциональное программирование;
история ФП-Lisp;
почему ФП "можно"?
практика;
выводы.

---

4 **Service Workers**

Цели занятия:

после занятия вы сможете:
понимать и использовать Service Workers API

---

5 **Функциональное программирование в JavaScript**

Цели занятия:

после занятия вы сможете:
разбираться в основах функционального программирования и спецификах применения паттернов функционального программирования в JavaScript.

теория; чистые функции, функциональные методы. Side Effects
инструменты. от lodash до ramda;
отладка функционального кода.

---

6 **Введение в TypeScript**

Цели занятия:

после занятия вы сможете:
различать TypeScript и JavaScript, использовать преимущества статической типизации;
писать и понимать код на языке TypeScript, разрабатывать приложения в полноценном объектно-ориентированном стиле.

введение в TypeScript: некоторые примеры, инструменты, `tsconfig.json`.
типы: простые, сложные, классы;
другие особенности.

Домашние задания

1 Перевести проект на TypeScript

Цель: Перевести проект на JavaScript к TypeScript
- Добавить типы
- Добавить зависимости
- Исправить ошибки типизации

Можно использовать свой репозиторий или любой опенсерсовый проект, например
- https://github.com/korzio/djv/tree/ts

---

7 **Особенности TypeScript**

Цели занятия:

после занятия вы сможете:
проанализировать специальные особенности TypeScript;
разбираться в Namespaces, Generics, Decorators, Type Definitions, Metadata;
применять типы и декораторы.

подведем итоги дизайна и разработки в Node.
namespaces;
generics;
Advanced Types;
Decorators: Metadata

4 React

1 Основы React и JSX

Цели занятия:

после занятия вы сможете:
настроить себе окружение для работы с React и использовать его;
применять синтаксис JSX;
создавать простые приложения на React.

фронтент;
фреймворк;
AJAX;
APA;
компоненты;
транспиляция и диалект JS.

---

2 Компоненты React - Lifecycle, State & Props

Цели занятия:

после занятия вы сможете:
разрабатывать полноценные React-компоненты в различных стилях;
понимать жизненный цикл компонент;
использовать Lifecycle API для управления компонентами приложения
моделировать состояние через state / props.
корректно управлять данными React приложения.

Lifecycle hooks API - методы;
пример - "Загрузка данных";
пример - "Работа с DOM";
ошибки;
API state & props, setState, etc.;
плюсы и минусы использования;
state container;
практика.

3	**Паттерны проектирования React (Higher-Order Components)**	**Цели занятия:** после занятия вы сможете: свободно декомпозировать компоненты Делать переиспользуемые контейнеры children; переиспользовать state-логику и лайфсайкл- хуки через HOC; написать умные компоненты с конфигурируемым отображением через render pro. React — библиотека для интерактивных интерфейсов; JSX — тонкий синтаксис шаблонов.
4	**Обзор современных возможностей React**	**Цели занятия:** после занятия вы сможете: разбираться в техниках и использовать различные hooks. примеры использования хуков, useState, useContext, useHistory.
5	**Состояние приложения - Flux и Redux**	**Цели занятия:** после занятия вы сможете: отличать основные понятия однонаправленной архитектуры Flux; ориентироваться и использовать возможности redux - создавать actions, reducers, а также применять redux в связке в React. использовать Redux в React приложениях. практика с thunk middleware; обзор альтернативных техник Redux (redux-saga).

6 **Routing в React - Оптимизация приложения**

Цели занятия:

после занятия вы сможете:
создавать систему routing для React приложений,
использовать библиотеку react-router;
использовать специальные возможности библиотеки для оптимизации отрисовки.

библиотеки.

Домашние задания

1 Routing для приложения погоды

Цель: В ходе выполнения ДЗ студент реализует 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.

---

7 **Подготовка React Приложения к Production, Best Practices**

Цели занятия:

после занятия вы сможет:
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.

---

8 **Микросервисная архитектура и аспекты SSR**

Цели занятия:

после занятия вы сможете:
- понимать что такое микросервисная архитектура во FrontEnd
- различать подходы проектирования приложений
- понимать различия в реализации архитектур веб приложений

современные подходы построения веб приложений
микросервисная архитектура
аспекты server-Side Rendering с React;

5 Angular

1 Введение в Angular

Цели занятия:

после занятия вы сможет:
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.

angular;
технологии;
angularCLI;
основы Angular;
module;
component;
service;
pipe;
directive;
templates.

---

2 Компоненты и директивы

Цели занятия:

после занятия вы сможет:
декомпозировать макет страницы на компоненты;
различать директивы и компоненты во фреймворке Angular;
создавать простые директивы и компоненты.

декораторы;
компоненты;
директивы;
события жизненного цикла.

---

3 Observables - RxJS

Цели занятия:

после занятия вы сможет:
отличать основные понятия паттерна;
применять шаблон проектирования Observables, используя библиотеку RxJS.

observable, observer, subscriber, operator.

---

4 Сервисы и состояние

Цели занятия:

приложения

после занятия вы сможет:
создавать сервисы для получения, отправки и хранения данных для приложений Angular.
разбираться в особенностях шаблона проектирования Dependency Injection и его имплементации в Angular.

@Input, @Output и другие способы передавать данные в приложении Angular;
сервис для получения данных.
Observables in Angular;
Dependency Injection / Service Locator / Inversion of control;
Angular DI Specifics.

Домашние задания

1 Структура приложения для запоминания иностранных слов

Цель: В этом ДЗ вы создадите приложение 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.

Создать сервисы для работы с текстом
- Сервис перевода слова - должен запрашивать перевод через API (например, https://tech.yandex.com/translate/, https://mymemory.translated.net/)
- 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`.

---

5 **Создание и управление формами в Angular**

Цели занятия:

после занятия вы сможет:
создавать формы, используя техники Dynamic Forms, Reactive Forms;
описывать валидацию и другие функции для элементов форм.

Dependency Injectors Hierarchy;
Template-Driven Forms;
Reactive Forms.

---

6 **Routing, тестирование и сборка в Angular**

Цели занятия:

после занятия вы сможет:
создавать Routing систему для приложений, используя внутренние подходы Angular - такие как Router, router-outlet и другие;
применять хэндлеры навигации Guards.
писать и запускать тесты для приложений Angular;
настраивать сборку приложений Angular;
использовать возможность сборки Server-side Rendering.

Classic HTML;
AJAX;
SPA.
Routing
тесты в Angular.
Сборка приложения для Production

Домашние задания

1 Routing для приложения запоминания иностранных слов

Цель: В этом ДЗ вы добавите Routing 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

Добавить routing, ссылки на страницы и переходы между компонентами приложения.
Добавить и актуализировать тесты для компонент приложения, настроить universal рендеринг приложения.

6 Vue

1	**Введение в SvelteJS**	**Цели занятия:** после занятия вы сможете: объяснить основные преимущества и цели фреймворка; запускать проект на SvelteJS; использовать шаблоны и создавать компоненты. история проекта, сравнение с другими технологиями; пример Hello World; обзор инструментов, используемых с SvelteJS; конфигурация Rollup; обзор синтаксиса шаблонов; пример создания компонентов.
2	**Особенности разработки приложений с SvelteJS**	**Цели занятия:** после занятия вы сможете: создавать Routing для приложений; описавать стили для проекта под SvelteJS; тестировать и подготовить проект к production. особенности создания routing; управление данными; особенности описания стилей; тестирование; оптимизация сборки.
3	**Основы Vue**	**Цели занятия:** после занятия вы сможете: настроить себе окружение IDE, зависимости и библиотеки для создания проектов и работы с Vue; создавать простейшие приложения используя Vue. hello Vue; MVVM; templates; dev tools.

4 **Компоненты, шаблонизатор и формы**

Цели занятия:

после занятия вы сможете:
объяснить синтаксис шаблонизаторы;
создавать компоненты, описывать атрибуты элементов.

templates;
components.

---

5 **Routing и Vue3**

Цели занятия:

после занятия вы сможете:
описывать routing для Vue приложений;
основы Vue3

router
Vue3

Домашние задания

1 Структура & Routing для приложения "Устный счет"

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

6	**Advanced Vue - Vuex**	**Цели занятия:** после занятия вы сможете: применять анимацию в компонентах; создавать плагины; разбираться в тонкостях Change Detection. plugins & directives; vuex.

7 Проектная работа

1 Вводное занятие по проектной работе. Обзор пройденных фреймворков и технологий

Цели занятия:

выбрать и обсудить тему проектной работы;
спланировать работу над проектом;
ознакомиться с регламентом работы над проектом;
выделять характеристики проектов и окружения;
решать задачи выбора и сравнения фреймворков, понимать их преимущества и недостатки.

правила работы над проектом и специфика проведения итоговой защиты;
требования к результату проекта и итоговой документации.

Домашние задания

1 Проектная работа

Цель: В этом ДЗ вы:
- выберете тему проекта;
- закрепите тему в чат с преподавателем;
- защитите проект.

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 Защита проектных работ

Цели занятия:

защитить проект и получить рекомендации экспертов.

презентация проектов перед комиссией;
вопросы и комментарии по проектам.