ЧОУ ВО «Тольяттинская академия управления»

Кафедра прикладной информатики и высшей математики

Б1.В.06

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (СРЕДСТВА)

Учебная дисциплина	*Фреймворки front-end разработчика*
По направлению подготовки	*09.03.03*
	«Прикладная информатика»
Профиль (программа бакалавриата)	«Дизайн и разработка цифровых программных продуктов»
Форма обучения	*Очная*

Оценочные материалы (средства) дисциплины рассмотрены (актуализированы) и утверждены на заседании кафедры прикладной информатики и высшей математики

Протокол заседания № 10 от «25» мая 2026 г.

Заведующий кафедрой Стрекалова Наталья Борисовна

2025

Оглавление

1. ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ И ЭТАПЫ ИХ ФОРМИРОВАНИЯ В ПРОЦЕССЕ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ

Оценочные материалы (средства) сформированы в соответствии с ФГОС ВО - бакалавриат по направлению подготовки 09.03.03 «Прикладная информатика», утвержденный приказом Министерства образования и науки Российской Федерации от 19.09.2017 № 922 (с изменениями и дополнениями от 26.11.2020, 08.02.2021). В соответствии с матрицей компетенций основной профессиональной образовательной программы «Дизайн и разработка цифровых программных продуктов» (уровень бакалавриата) в процессе обучения по дисциплине «Фреймворки front-end разработчика» происходит формирование закрепленных за дисциплиной компетенций обучающихся. Оценка сформированности компетенций на каждом этапе обучения происходит через оценку планируемых результатов обучения по дисциплине (знаний, умений, навыков).

Результаты освоения образовательной программы (компетенции)	Индикаторы достижения компетенций	Планируемые результаты обучения по дисциплине	Этапы формирования компетенции (семестры и темы)	Оценочные средства
ПК-2. Способен проектировать цифровой программный продукт	ПК-2.3. Проектирует пользовательский интерфейс с учетом тенденций развития цифровых технологий	**Знать:** - основы прототипирования и разработки дизайна интерфейса web-приложений; - этапы и инструменты проектирования web-приложений и их компонентов; **Уметь:** - продумать и визуализировать логику web-приложения, его пользовательский интерфейс и отдельные компоненты; - создавать графические решения, максимально соответствующие функциональности web-приложения **Владеть:** - инструментами разработки пользовательских интерфейсов web-приложений.	Тема 1. Основы Vue.js Тема 3. Основные принципы и архитектура Тема 6. Знакомство с экосистемой React. Построение приложений с Redux.js. Тема 7. React для SPA: react-router и продвинутые API	- устный опрос по темам 1, 3, 6-7 -практические работы по теме 1, 3, 6; - защита практических работ по теме 7
ПК-3. Способен разрабатывать цифровой программный продукт	ПК-3.1. Осуществляет разработку программного кода и компонентов цифрового программного продукта с помощью специализированных цифровых платформ для индивидуальной и совместной разработки	**Знать:** - основы языка JavaScript; - инструменты разработки Frontend приложений; - основы разработки web-приложений на Vue.js, Angular, React; **Уметь:** - разрабатывать web-приложения с помощью фреймворков и библиотек JavaScript (Vue.js, Angular, React); **Владеть:** - навыками построения приложений с Redux.js; - навыками применения компонентного подхода при разработке web-приложений; - навыками разработки одностраничных приложений (SPA).	Тема 1. Основы Vue.js Тема 2. Углубление во Vue.js и разработку компонентов Тема 3. Основные принципы и архитектура Тема 4. Компоненты и шаблоны Тема 5. Реактивность. Маршрутизация. Формы Тема 6. Знакомство с экосистемой	- устный опрос по темам 1, 3, 6-7 -практические работы по темам 1-6; - защита практических работ по теме 7

			React. Построение приложений с Redux.js. Тема 7. React для SPA: react-router и продвинутые API	
	ПК-3.2. Обеспечивает интеграцию программных модулей и компонентов цифрового программного продукта на основе облачных и отраслевых ИТ-решений	**Знать:** - принципы и назначение фреймворков и библиотек (Vue.js, Angular, React); - приемы написания минималистического кода с гибкой архитектурой. **Уметь:** - обеспечивать интеграцию программных модулей и компонентов web-приложений. **Владеть:** - методами и инструментальными средствами интеграции программных модулей и компонентов web-приложений.	Тема 1. Основы Vue.js Тема 7. React для SPA: react-router и продвинутые API	- устный опрос по темам 1, 7 - практические работы по теме 1; - защита практических работ по теме 7
ПК-5. Способен обеспечивать качество работы цифрового программного продукта	ПК-5.3. Осуществляет работы по интеграционному тестированию цифрового программного продукта с внешним окружением	**Знать:** - современные инструменты и практики тестирования JavaScript-кода; - процесс интеграции программного обеспечения и интеграцию разработки API **Уметь:** - проводить интеграционное тестирование кода с помощью различных инструментов. **Владеть:** - навыками создания среды для запуска тестов.	Тема 4. Компоненты и шаблоны Тема 7. React для SPA: react-router и продвинутые API	- устный опрос по теме 7; -практические работы по теме 4; - защита практических работ по теме 7

2. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (СРЕДСТВА) ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ И ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ, ОБЕСПЕЧИВАЮЩИХ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ В ПРОЦЕССЕ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ

Устные опросы

Примерные вопросы для проведения устного опроса по теме «Основы Vue.js»

1. Введение в JavaScript: основы, диалог с пользователем и вычисления, массивы, обработка событий.
2. Инструменты разработки Frontend приложений.
3. Синтаксис шаблонов.
4. Реактивность вычисляемые свойства, отслеживание.
5. Работа с DOM событиями, атрибутами, формами.
6. Хуки жизненного цикла.
7. Компонентный подход.

8. Создание компонентов.
9. Параметры и события компонентов.
10. Однофайловые компоненты (SFC) и инкапсуляция стилей.
11. Создание и разработка приложений с @vue/cli.
12. Одностраничные приложения (SPA).
13. Маршрутизация на клиенте, vue-router.
14. Unit-тестирование

Примерные вопросы для проведения устного опроса
по теме «Основные принципы и архитектура Angular»

1. Назначение фреймворка Angular.
2. Модульная архитектура Angular.
3. Атрибутивные и структурные директивы Angular.
4. Структура приложения и практические рекомендации по её формированию. Синтаксис привязки данных и шаблоны.
5. Формы и валидация данных (реактивные формы и валидаторы).
6. Проекция контента. Стратегия обнаружения изменений onPush.

Примерные вопросы для проведения устного опроса
по теме «Знакомство с экосистемой React. Построение приложений с Redux.js»

1. Создание простых компонентов.
2. Декларативный подход.
3. Отличие React.js от других популярных фреймворков.
4. Знакомство с экосистемой React.
5. Использование сторонних компонентов и работа с формами.
6. Особенности Redux.js: функциональный подход, Redux dev tools.
7. Синхронный поток с Redux.
8. React-redux для связи компонентов с логикой.
9. Мемоизированные селекторы reselect.
10. Side-effects в Redux: создание и использование Middlewares.
11. Получение данных от сервера. Асинхронные экшены с redux-thunk.

Примерные вопросы для проведения устного опроса
по теме «React для SPA: react-router и продвинутые API»

1. Разработка single-page applications.
2. Использование react-router и продвинутые элементы API React.JS.
3. Настройка вложенных роутов.
4. Выбор и настройка history для приложения.
5. Объединение react-router и Redux.
6. Обработка серверных ошибок и декларативное управление роутером. Использование props.children для композиции компонентов.
7. Анимации в React, CSSTransitionGroup.

8. GraphQL + Relay/Apollo. MobX vs Redux.
9. Разные подходы к сайд-эффектам: redux-thunk, redux-loop, redux-saga, redux-observable.

Критерии оценивания устного опроса:

- оценка «*отлично*» выставляется обучающемуся, если он ответил развернуто на один из заданных вопросов, активно дополнял ответы других обучающихся или задавал им дополнительные вопросы;
- оценка «*хорошо*» выставляется обучающемуся, если он ответил развернуто на один из заданных вопросов или ответил кратко на ряд вопросов;
- оценка «*удовлетворительно*» выставляется обучающемуся, если он кратко ответил на один из задаваемых вопросов или просто дополнял ответы других обучающихся, задавал им дополнительные вопросы;
- оценка «*неудовлетворительно*» выставляется обучающемуся, если он не смог ответить правильно на заданный вопрос, либо не отвечал на вопросы и не участвовал в их обсуждении.

Практические работы

Практическое задание по теме «Основы Vue.js»

Практическое задание:

1. Создать приложение с помощью vue.js
2. Измените в коде в data значение свойства message с "Hello, FructCode!" на "Hello, VueJS!" Остальной код оставить без изменений.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title> VueJS</title>
</head>
<body>

  <div id="app">
    <input type="text" v-model="message">
    <h1>{{ message }}</h1>
  </div>

  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
  <script>

    var app = new Vue({
      el: '#app',
      data: {
        message: 'Hello, FructCode!'
      }
    })

  </script>
```

```
</body>
</html>
```

3. Создайте функционал для очистки value из input и из данных message при клике на кнопку Clean Message с помощью метода cleanMessage. Точку с запятой(;), в теле метода cleanMessage не ставьте. Остальной код оставить без изменений.

Критерии оценивания:

- оценка «*отлично*» выставляется обучающемуся, если задание было выполнено полном объеме, текст программы хорошо структурирован, в тексте программы присутствуют комментарии, предложен корректный (логично выстроенный) программный код, обеспечивающий заданный функционал (очищение сообщения);
- оценка «*хорошо*» выставляется обучающемуся, если задание было выполнено в полном объеме, но с недочетами, в тексте программы могут отсутствовать комментарии, в исходном коде есть изменения;
- оценка «*удовлетворительно*» выставляется обучающемуся, если задание было выполнено частично, функционал реализован, но работает с ошибками;
- оценка «*неудовлетворительно*» выставляется обучающемуся, если программа фактически не работает или программный код отсутствует.

Практическое задание по теме «Углубление во Vue.js и разработку компонентов»

Задание:

Во Vue-приложении все данные, методы, вычисляемые свойства размещаются в корневом экземпляре Vue. В дальнейшем это приведет к сложностям при поддержке кода, поэтому в практическом задании требуется разбить код на модульные части, с которыми процесс разработки будет более простым и гибким.

Для этого:

1. Взять существующий код и перенести его в новый компонент. Для этого в файле main.js регистрируется компонент:

```
Vue.component('product', {})
```

Первый аргумент — это выбранное нами имя компонента. Второй — это объект с опциями, как при создании экземпляра Vue.

В экземпляре Vue используется свойство `el` для организации его привязки к элементу DOM. В случае с компонентом используется свойство `template`, которое определяет HTML-код компонента.

2. Описать шаблон компонента в объекте с опциями:

```
Vue.component('product', {
 template: `
   <div class="product">
... // Здесь будет весь HTML-код, который раньше был в элементе с классом product
   </div>
```

```
  `
})
```

Если шаблон должен включать в себя множество элементов, например — набор элементов, заключённых в с классом product, эти элементы должны быть помещены во внешний элемент-контейнер. В результате в шаблоне будет лишь один корневой элемент.

3. Добавить в компонент данные, методы, вычисляемые свойства, которые раньше были в корневом экземпляре Vue, когда в шаблоне находится HTML-код, который раньше был в файле index.html.
4. Разместить компонент product в коде файла index.html.
5. Добавить в код index.html 2 компонента product.

Критерии оценивания:

- оценка «*отлично*» выставляется обучающемуся, если задание было выполнено полном объеме, текст программы хорошо структурирован, в тексте программы присутствуют комментарии, компонент практически полностью совпадает со структурой экземпляра Vue, сущность data выступает в роли функции, компоненты содержат различные данные;
- оценка «*хорошо*» выставляется обучающемуся, если задание было выполнено почти в полном объеме и компонент частично совпадает со структурой экземпляра Vue, программа работает с незначительными ошибками (в логике или в эффективности кода), в тексте программы могут отсутствовать комментарии;
- оценка «*удовлетворительно*» выставляется обучающемуся, если задание было выполнено частично, присутствует программный код, в компоненте присутствуют данные, методы, вычисляемые свойства;
- оценка «*неудовлетворительно*» выставляется обучающемуся, если компонент совсем не совпадает со структурой экземпляра Vue, сущность data выступает как свойство.

Практическое задание по теме «Основные принципы и архитектура» раздела 3. Angular

Задание:

1. Реализовать динамические формы с помощью стандартных средств Angular.

Критерии оценивания:

- оценка «*отлично*» выставляется обучающемуся, если задание было выполнено, динамические формы созданы, предложен корректный (логично выстроенный) программный код;
- оценка «*хорошо*» выставляется обучающемуся, если задание выполнено частично, предложен корректный (логично выстроенный) программный код;
- оценка «*удовлетворительно*» выставляется обучающемуся, если обучающийся использует стандартные средства Angular, но формы реализованы не полностью или работоспособны частично;

- оценка «*неудовлетворительно*» выставляется обучающемуся, если задание не было выполнено совсем.

Практическое задание по теме «Компоненты и шаблоны» раздела 3. Angular

Задание:

1. Создать компонент с использованием Angular CLI.

Критерии оценивания:

- оценка «*отлично*» выставляется обучающемуся, если компонент создан и эффективно используется;
- оценка «*хорошо*» выставляется обучающемуся, если задание выполнено частично, новый компонент со своими собственными файлами (HTML, CSS и TypeScript) регистрирован в модуле приложения;
- оценка «*удовлетворительно*» выставляется обучающемуся, если обучающийся использует стандартные средства Angular, но компонент не создан;
- оценка «*неудовлетворительно*» выставляется обучающемуся, если задание не было выполнено совсем.

Практическое задание по теме «Реактивность. Маршрутизация. Формы» раздела 3. Angular

Задание:

1. Использовать для работы с формами директиву NgForm.

Критерии оценивания:

- оценка «*отлично*» выставляется обучающемуся, если создан объект FormGroup и привязан к форме, что позволяет отслеживать состояние формы, управлять ее валидацией;
- оценка «*хорошо*» выставляется обучающемуся, если задание выполнено частично, определена переменная #myForm, которая инициализирована значением "ngForm";
- оценка «*удовлетворительно*» выставляется обучающемуся, если обучающийся использует стандартные средства Angular, но форма не создана;
- оценка «*неудовлетворительно*» выставляется обучающемуся, если задание не было выполнено совсем.

Практическое задание по теме «Знакомство с экосистемой React. Построение приложение с Redux.js»

Задание:

2. Реализовать динамические формы с помощью React.

Критерии оценивания:

- оценка «*отлично*» выставляется обучающемуся, если задание было выполнено, динамические формы созданы, предложен корректный (логично выстроенный) программный код;
- оценка «*хорошо*» выставляется обучающемуся, если задание выполнено частично, предложен корректный (логично выстроенный) программный код;
- оценка «*удовлетворительно*» выставляется обучающемуся, если обучающийся использует React, но формы реализованы не полностью или работоспособны частично;
- оценка «*неудовлетворительно*» выставляется обучающемуся, если задание не было выполнено совсем.

Практическое задание по теме
«React для SPA: react-router и продвинутые API»

Задание:

3. Продемонстрировать различные подходы к сайд-эффектам: redux-thunk, redux-loop, redux-saga, redux-observable.

Критерии оценивания:

- оценка «*отлично*» выставляется обучающемуся, если задание было выполнено, различные подходы к сайд-эффектам продемонстрированы в программном коде;
- оценка «*хорошо*» выставляется обучающемуся, если задание выполнено частично, предложен корректный (логично выстроенный) программный код с частично выполненными сайд-эффектами;
- оценка «*удовлетворительно*» выставляется обучающемуся, если обучающийся использует React, но сайд-эффекты продемонстрированы частично;
- оценка «*неудовлетворительно*» выставляется обучающемуся, если задание не было выполнено совсем.

3. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (СРЕДСТВА) ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ

ОЦЕНКА ЗНАНИЙ ОБУЧАЮЩИХСЯ НА ЭКЗАМЕНЕ

Список вопросов для подготовки к экзамену

Компетенция ПК-2. Способен проектировать цифровой программный продукт

Индикатор ПК-2.3. Проектирует пользовательский интерфейс с учетом тенденций развития цифровых технологий

Обучающийся знает: основы прототипирования и разработки дизайна интерфейса web-приложений; этапы и инструменты проектирования web-приложений и их компонентов

1. Инструменты разработки Frontend приложений.
2. Работа с DOM событиями, атрибутами, формами.
3. Компонентный подход.
4. Одностраничные приложения (SPA).
5. Назначение фреймворка Angular.

6. Формы и валидация данных (реактивные формы и валидаторы).
7. Особенности Redux.js: функциональный подход, Redux dev tools.
8. Компоненты-обёртки. Компоненты-формы и однонаправленный поток данных.
9. Реактивность вне компонентов и реализация реактивности.
10. Структура приложения и практические рекомендации по её формированию в Angular.
11. Проекция контента в Angular.
12. Стратегия обнаружения изменений onPush в Angular.
13. Стили и шаблоны компонента в Angular.
14. Жизненный цикл компонента в Angular.

Компетенция ПК-3. Способен разрабатывать цифровой программный продукт

Индикатор ПК-3.1. Осуществляет разработку программного кода и компонентов цифрового программного продукта с помощью специализированных цифровых платформ для индивидуальной и совместной разработки

Обучающийся знает: основы языка JavaScript; инструменты разработки Frontend приложений; основы разработки web-приложений на Vue.js, Angular, React;

1. JavaScpipt: назначение и основные инструменты для создания современных web-приложений
2. Инструменты разработки Frontend приложений.
3. Маршрутизация на клиенте, vue-router.
4. Назначение фреймворка Angular.
5. Жизненный цикл компонента в Angular.
6. Создание простых компонентов React.
7. Отличие React.js от других популярных фреймворков.
8. Знакомство с экосистемой Использование сторонных сторонних компонентов и работа с формами.

Компетенция ПК-3. Способен разрабатывать цифровой программный продукт

Индикатор ПК-3.2. Обеспечивает интеграцию программных модулей и компонентов цифрового программного продукта на основе облачных и отраслевых ИТ-решений

Обучающийся знает: принципы и назначение фреймворков и библиотек (Vue.js, Angular, React); приемы написания минималистического кода с гибкой архитектурой.

1. Создание и разработка приложений с @vue/cli.
2. Ограничения Vue.js, его зона ответственности и работа с другими библиотеками.
3. Синтаксис привязки данных и шаблоны в Angular.
4. Формы и валидация данных (реактивные формы и валидаторы) в Angular.

5. Объект Observable и библиотека RxJS. Обработка ошибок.
6. Особенности Redux.js: функциональный подход, Redux dev tools.
7. Объединение react-router и Redux.
8. Обработка серверных ошибок и декларативное управление роутером.
9. Анимации в React, CSSTransitionGroup.
10. Разные подходы к сайд-эффектам: redux-thunk, redux-loop, redux-saga, redux-observable.
11. Основные этапы создания web-приложения
12. Инструменты разработки Frontend приложений.
13. Среды разработки приложений.
14. Основы рендеринга, Virtual DOM, render-функции, JSX.
15. HttpClient и отправка запросов.
16. Создание простых компонентов React.
17. Разные подходы к сайд-эффектам: redux-thunk, redux-loop, redux-saga, redux-observable.

Компетенция ПК-5. Способен обеспечивать качество работы цифрового программного продукта

Индикатор ПК-5.3. Осуществляет работы по интеграционному тестированию цифрового программного продукта с внешним окружением

Обучающийся знает: современные инструменты и практики тестирования JavaScript-кода; процесс интеграции программного обеспечения и интеграцию разработки API

1. Unit-тестирование Vue.js приложения с Jest и vue-test-utils
2. Синтаксис привязки данных и шаблоны в Angular.
3. Проекция контента в Angular.
4. Взаимодействие между модулями. ngClass и ngStyle. Создание атрибутивных директив.
5. Взаимодействие с пользователем, HostListener и HostBinding. Получение параметров в директивах. Структурные директивы ngIf, ngFor, ngSwitch.
6. Отправка данных в запросе. POST-запросы. Определение маршрутов. Создание ссылок. Параметры маршрута. Параметры строки запроса.
7. Модуль FormsModule и директива NgModel. Получение и изменение модели. Состояние модели и валидация. Директива NgForm. Reactive Forms
8. Синхронный поток с Redux. React-redux для связи компонентов с логикой. Использование react-router и продвинутые элементы API React.JS.
9. Объединение react-router и Redux.
10. Обработка серверных ошибок и декларативное управление роутером.

ОЦЕНКА УМЕНИЙ И НАВЫКОВ ОБУЧАЮЩИХСЯ НА ЭКЗАМЕНЕ

Компетенция ПК-2. Способен проектировать цифровой программный продукт

Индикатор ПК-2.3. Проектирует пользовательский интерфейс с учетом тенденций развития цифровых технологий

Обучающийся умеет: продумать и визуализировать логику web-приложения, его пользовательский интерфейс и отдельные компоненты; создавать графические решения, максимально соответствующие функциональности web-приложения

Обучающийся владеет: инструментами разработки пользовательских интерфейсов web-приложений

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам выполнения практического задания по созданию web-приложения по заданным критериям.

Компетенция ПК-3. Способен разрабатывать цифровой программный продукт

Индикатор ПК-3.1. Осуществляет разработку программного кода и компонентов цифрового программного продукта с помощью специализированных цифровых платформ для индивидуальной и совместной разработки

Обучающийся умеет: разрабатывать web-приложения с помощью фреймворков и библиотек JavaScript (Vue.js, Angular, React);

Обучающийся владеет: навыками построения приложений с Redux.js; навыками применения компонентного подхода при разработке web-приложений; навыками разработки одностраничных приложений (SPA).

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам выполнения практического задания по созданию web-приложения по заданным критериям.

Компетенция ПК-3. Способен разрабатывать цифровой программный продукт

Индикатор ПК-3.2. Обеспечивает интеграцию программных модулей и компонентов цифрового программного продукта на основе облачных и отраслевых ИТ-решений

Обучающийся умеет: обеспечивать интеграцию программных модулей и компонентов web-приложений.

Обучающийся владеет: методами и инструментальными средствами интеграции программных модулей и компонентов web-приложений

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам выполнения практического задания по созданию web-приложения по заданным критериям.

Компетенция ПК-5. Способен обеспечивать качество работы цифрового программного продукта

Индикатор ПК-5.3. Осуществляет работы по интеграционному тестированию цифрового программного продукта с внешним окружением

Обучающийся умеет: проводить интеграционное тестирование кода с помощью различных инструментов.

Обучающийся владеет: навыками создания среды для запуска тестов

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам выполнения практического задания по тестированию web-приложения по заданным критериям.

4. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ, ОПРЕДЕЛЯЮЩИЕ ПРОЦЕДУРЫ ОЦЕНОВАНИЯ ЗНАНИЙ, УМЕНИЙ И НАВЫКОВ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ

ПРОЦЕДУРА ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

Для контроля усвоения обучающимися данной дисциплины и достижения запланированных результатов обучения учебным планом предусмотрен экзамен, проводимый в форме устного ответа на теоретический вопрос. При выставлении итоговой оценки по дисциплине учитываются результаты текущего контроля, отражающего сформированность практических умений и навыков обучающегося. Во время обучения обучающиеся участвуют в разных мероприятиях текущего контроля: участвуют в устных теоретических опросах, выполняют практические задания. Для допуска к экзамену обучающемуся необходимо выполнить все практические задания.

КРИТЕРИИ ОЦЕНИВАНИЯ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИЙ

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
Компетенция ПК-2. Способен проектировать цифровой программный продукт Индикатор ПК-2.3. Проектирует пользовательский интерфейс с учетом тенденций развития цифровых технологий					
Знать: основы прототипирования и разработки дизайна интерфейса web-приложений; этапы и инструменты проектирования web-приложений и их компонентов	Не знает	Фрагментарные знания основ прототипирования и разработки дизайна интерфейса web-приложений; этапы и инструменты проектирования web-приложений и их компонентов	Общие, но не структурированные знания основ прототипирования и разработки дизайна интерфейса web-приложений; этапы и инструменты проектирования web-приложений	Сформированные, но содержащие отдельные пробелы знания основ прототипирования и разработки дизайна интерфейса web-приложений; этапы и инструменты	Сформированные систематизированные прочные знания основ прототипирования и разработки дизайна интерфейса web-приложений; этапы и инструменты проектирования

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
			и их компонентов	проектирования web-приложений и их компонентов	web-приложений и их компонентов
Уметь: продумать и визуализировать логику web-приложения, его пользовательский интерфейс и отдельные компоненты; - создавать графические решения, максимально соответствующие функциональности web-приложения	Не умеет	Частично освоенные умения продумать и визуализировать логику web-приложения, его пользовательский интерфейс и отдельные компоненты; - создавать графические решения, максимально соответствующие функциональности web-приложения	В целом освоенные, но не подкрепляемые знаниями и самостоятельностью выполнения умения продумать и визуализировать логику web-приложения, его пользовательский интерфейс и отдельные компоненты; - создавать графические решения, максимально соответствующие функциональности web-приложения	В целом сформированные и самостоятельно выполняемые, но не всегда интеллектуально обоснованные умения продумать и визуализировать логику web-приложения, его пользовательский интерфейс и отдельные компоненты; - создавать графические решения, максимально соответствующие функциональности web-приложения	Успешно сформированные, самостоятельно и осознанно выполняемые умения продумать и визуализировать логику web-приложения, его пользовательский интерфейс и отдельные компоненты; - создавать графические решения, максимально соответствующие функциональности web-приложения
Владеть: - инструментами разработки пользовательских интерфейсов web-приложений	Не владеет	Фрагментарно сформированные навыки использования инструментовиразработки пользовательских интерфейсов web-приложений	В целом сформированные, но выполняемые на элементарном уровне навыки использования инструментовиразработки пользовательских интерфейсов web-приложений	Сформированные, но не устойчивые в сложных ситуациях навыки использования инструментовиразработки пользовательских интерфейсов web-приложений	Успешно сформированные, устойчивые, технологически грамотно выполняемые навыки использования инструментовиразработки пользовательских интерфейсов web-приложений
Компетенция ПК-3. Способен разрабатывать цифровой программный продукт Индикатор ПК-3.1. Осуществляет разработку программного кода и компонентов цифрового программного продукта с помощью специализированных цифровых платформ для индивидуальной и совместной разработки					
Знать: основы языка JavaScript; инструменты разработки Frontend	Не знает	Фрагментарные знания основ языка JavaScript; инструментов разработки Frontend	Общие, но не структурированные знания основ языка JavaScript; инструментов	Сформированные, но содержащие отдельные пробелы знания основ языка	Сформированные систематизированные прочные знания основ языка

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
приложений; основы разработки web-приложений на Vue.js, Angular, React		приложений; основ разработки web-приложений на Vue.js, Angular, React	разработки Frontend приложений; основ разработки web-приложений на Vue.js, Angular, React	JavaScript; инструментов разработки Frontend приложений; основ разработки web-приложений на Vue.js, Angular, React	JavaScript; инструментов разработки Frontend приложений; основ разработки web-приложений на Vue.js, Angular, React
Уметь: разрабатывать web-приложения с помощью фреймворков и библиотек JavaScript (Vue.js, Angular, React)	Не умеет	Частично освоенные умения разрабатывать web-приложения с помощью фреймворков и библиотек JavaScript (Vue.js, Angular, React)	В целом освоенные, но не подкрепляемые знаниями и самостоятельностью выполнения умения разрабатывать web-приложения с помощью фреймворков и библиотек JavaScript (Vue.js, Angular, React)	В целом сформированные и самостоятельно выполняемые, но не всегда интеллектуально обоснованные умения представлять разрабатывать web-приложения с помощью фреймворков и библиотек JavaScript (Vue.js, Angular, React)	Успешно сформированные, самостоятельно и осознанно выполняемые умения представлять разрабатывать web-приложения с помощью фреймворков и библиотек JavaScript (Vue.js, Angular, React)
Владеть: навыками построения приложений с Redux.js; навыками применения компонентного подхода при разработке web-приложений; навыками разработки одностраничных приложений (SPA).	Не владеет	Фрагментарно сформированные навыки построения приложений с Redux.js; применения компонентного подхода при разработке web-приложений; разработки одностраничных приложений (SPA).	В целом сформированные, но выполняемые на элементарном уровне навыки построения приложений с Redux.js; применения компонентного подхода при разработке web-приложений; разработки одностраничных приложений (SPA).	Сформированные, но не устойчивые в сложных ситуациях навыки построения приложений с Redux.js; применения компонентного подхода при разработке web-приложений; разработки одностраничных приложений (SPA).	Успешно сформированные, устойчивые, технологически грамотно выполняемые навыки построения приложений с Redux.js; применения компонентного подхода при разработке web-приложений; разработки одностраничных приложений (SPA).
Компетенция ПК-3. Способен разрабатывать цифровой программный продукт Индикатор ПК-3.2. Обеспечивает интеграцию программных модулей и компонентов цифрового программного продукта на основе облачных и отраслевых ИТ-решений					
Знать: принципы и назначение фреймворков и библиотек (Vue.js, Angular,	Не знает	Фрагментарные знания принципов и назначения фреймворков и библиотек	Общие, но не структурированные знания принципов и назначения фреймворков и	Сформированные, но содержащие отдельные пробелы знания принципов и	Сформированные систематизированные прочные знания принципов и

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
React); приемы написания минималистического кода с гибкой архитектурой		(Vue.js, Angular, React); приемов написания минималистического кода с гибкой архитектурой	библиотек (Vue.js, Angular, React); приемов написания минималистического кода с гибкой архитектурой	назначения фреймворков и библиотек (Vue.js, Angular, React); приемов написания минималистического кода с гибкой архитектурой	назначения фреймворков и библиотек (Vue.js, Angular, React); приемов написания минималистического кода с гибкой архитектурой
Уметь: - обеспечивать интеграцию программных модулей и компонентов web-приложений	Не умеет	Частично освоенные умения обеспечивать интеграцию программных модулей и компонентов web-приложений	В целом освоенные, но не подкрепляемые знаниями и самостоятельностью выполнения умения обеспечивать интеграцию программных модулей и компонентов web-приложений	В целом сформированные и самостоятельно выполняемые, но не всегда интеллектуально обоснованные умения обеспечивать интеграцию программных модулей и компонентов web-приложений	Успешно сформированные, самостоятельно и осознанно выполняемые умения обеспечивать интеграцию программных модулей и компонентов web-приложений
Владеть: - инструментальными средствами интеграции программных модулей и компонентов web-приложений	Не владеет	Фрагментарно сформированные навыки применения инструментальных средств интеграции программных модулей и компонентов web-приложений	В целом сформированные, но выполняемые на элементарном уровне навыки применения инструментальных средств интеграции программных модулей и компонентов web-приложений	Сформированные, но не устойчивые в сложных ситуациях навыки применения инструментальных средств интеграции программных модулей и компонентов web-приложений	Успешно сформированные, устойчивые, технологически грамотно выполняемые навыки применения инструментальных средств интеграции программных модулей и компонентов web-приложений
Компетенция ПК-5. Способен обеспечивать качество работы цифрового программного продукта Индикатор ПК-5.3. Осуществляет работы по интеграционному тестированию цифрового программного продукта с внешним окружением					
Знать: - современные инструменты и практики тестирования JavaScript-кода; - процесс интеграции программного	Не знает	- Фрагментарные знания современных инструментов и практики тестирования JavaScript-кода; процесса интеграции программного	Общие, но не структурированные знания современных инструментов и практики тестирования JavaScript-кода; процесса интеграции программного	Сформированные, но содержащие отдельные пробелы знания современных инструментов и практики тестирования JavaScript-кода; процесса	Сформированные систематизированные прочные знания современных инструментов и практики тестирования JavaScript-кода; процесса

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
обеспечения и интеграцию разработки API		обеспечения и интеграцию разработки API	обеспечения и интеграцию разработки API	интеграции программного обеспечения и интеграцию разработки API	интеграции программного обеспечения и интеграцию разработки API
Уметь: - проводить интеграционное тестирование кода с помощью различных инструментов	Не умеет	Частично освоенные умения проводить интеграционное тестирование кода с помощью различных инструментов	В целом освоенные, но не подкрепляемые знаниями и самостоятельностью выполнения умения проводить интеграционное тестирование кода с помощью различных инструментов	В целом сформированные и самостоятельно выполняемые, но не всегда интеллектуально обоснованные умения проводить интеграционное тестирование кода с помощью различных инструментов	Успешно сформированные, самостоятельно и осознанно выполняемые умения проводить интеграционное тестирование кода с помощью различных инструментов
Владеть: - навыками создания среды для запуска тестов	Не владеет	Фрагментарно сформированные навыки создания среды для запуска тестов	В целом сформированные, но выполняемые на элементарном уровне навыки создания среды для запуска тестов	Сформированные, но не устойчивые в сложных ситуациях навыки создания среды для запуска тестов	Успешно сформированные, устойчивые, технологически грамотно выполняемые навыки создания среды для запуска тестов

КРИТЕРИИ ОЦЕНКИ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ

Критерии оценивания для экзамена:

При выставлении итоговой оценки за дисциплину необходимо учитывать результаты текущего контроля и критерии оценивания сформированности компетенций.

Оценка «**отлично**» выставляется обучающемуся, если:

- при ответе на теоретический вопрос были продемонстрированы глубокие, систематизированные и прочные знания (в соответствии с критериями сформированности компетенций), умения логично и обоснованно излагать материал, быстро ориентироваться в дополнительных вопросах и давать полные ответы;
- практические задания выполнены безошибочно и в полном соответствии с требованиями на оценку «отлично», обучающийся в работе продемонстрировал устойчивые навыки работы и осознанно исполняемые действия для разработки web-приложений, способность вносить коррективы и обосновывать свои действия;
- во время сдачи практического задания обучающийся логически стройно, доходчиво и грамотно продемонстрировал весь функционал разработанного web-приложения, полно и доступно отвечал на дополнительные вопросы;

- обучающийся участвовал во всех видах текущего контроля и получал по ним максимальные баллы (допустимо отдельные отметки «хорошо»).

Все вышеперечисленное говорит о сформированности запланированных компетенций на высоком уровне.

Оценка «**хорошо**» выставляется обучающемуся, если:

- при ответе на теоретический вопрос было продемонстрировано хорошее, но недостаточно полное изложение материала, с отдельными пробелами в знаниях (возможно, незначительными неточностями в употреблении терминов), логичное изложение материала, но недостаточно быстрая ориентация в нем при ответе на дополнительные вопросы;
- практические задания выполнены на оценку «отлично» или «хорошо» (с незначительными замечаниями и/или ошибками), обучающийся продемонстрировал хорошие навыки работы и осознанно исполняемые действия по разработке web-приложений;
- во время сдачи практического задания обучающийся достаточно подробно продемонстрировал функционал web-приложения и особенности его разработки, доступно отвечал на дополнительные вопросы;
- обучающийся участвовал во всех видах текущего контроля и получал по ним в основном отметки «хорошо».

Все вышеперечисленное говорит о сформированности запланированных компетенций на хорошем уровне.

Оценка «**удовлетворительно**» выставляется обучающемуся, если:

- при ответе на теоретический вопрос было продемонстрировано поверхностное владение материалом и фрагментарные знания, отсутствие стройного и логично выстроенного ответа, затруднение в употреблении терминов и приведении примеров, плохая ориентация в материале при ответе на дополнительные вопросы;
- практические задания выполнены на оценку «хорошо» или «удовлетворительно» (с ошибками и замечаниями), с отклонениями от требований, обучающийся затруднялся в объяснении применяемых средств и инструментов для разработки web-приложений;
- во время сдачи практического задания обучающийся невнятно представлял функционал web-приложений, не смог ответить на все дополнительные вопросы;
- обучающийся участвовал не во всех мероприятиях текущего контроля и получал по ним в основном удовлетворительные отметки.

Все вышеперечисленное говорит о сформированности запланированных компетенций на низком уровне.

Оценка «**неудовлетворительно**» выставляется обучающемуся, если:

- при ответе на теоретический вопрос было продемонстрировано незнание значительного объема материала, «фрагментарность имеющихся знаний, отсутствие стройного и логично выстроенного ответа, затруднение в употреблении терминов и приведении примеров, неумение выделить главное, сделать выводы и обобщения, неспособность отвечать на дополнительные вопросы;
- практические задания выполнены на оценку «удовлетворительно» со значительными отклонениями от требований, с грубыми ошибками и множеством замечаний, обучающийся продемонстрировал частично сформированные навыки работы, затрудняется в объяснении применяемых средств для разработки web-приложений, не способен внести коррективы в работу;
- во время сдачи практического задания обучающийся представлял web-приложение частично, без подробностей программной разработки, не смог ответить на дополнительные вопросы;

- обучающийся участвовал не во всех мероприятиях текущего контроля и получал по ним в основном удовлетворительные отметки.

  Все вышеперечисленное говорит о частичной сформированности запланированных компетенций.

5. ДИАГНОСТИЧЕСКИЕ ЗАДАНИЯ ДЛЯ ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ, ХАРАКТЕРИЗУЮЩИХ УРОВЕНЬ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИЙ

Компетенция ПК-2. Способен проектировать цифровой программный продукт

Обучающийся знает: основы прототипирования и разработки дизайна интерфейса web-приложений; этапы и инструменты проектирования web-приложений и их компонентов.
Обучающийся умеет: продумать и визуализировать логику web-приложения, его пользовательский интерфейс и отдельные компоненты; создавать графические решения, максимально соответствующие функциональности web-приложения.
Обучающийся владеет: инструментами разработки пользовательских интерфейсов web-приложений.

ЗАДАНИЯ ЗАКРЫТОГО ТИПА

1. Что такое прототипирование в веб-приложениях?
   а) Создание макета интерфейса приложения.
   б) Разработка минимального рабочего варианта приложения.
   в) Тестирование готового продукта.
   г) Анализ пользовательского опыта.

Правильный ответ: б) разработка минимального рабочего варианта приложения.

2. Какие этапы включает процесс прототипирования?
   а) Определение целей, создание прототипа, тестирование, улучшение прототипа.
   б) Определение целей, создание прототипа, тестирование, внедрение прототипа.
   в) Создание прототипа, тестирование, улучшение прототипа, внедрение прототипа.
   г) Определение целей, создание прототипа, тестирование, анализ результатов.

Правильный ответ: в) создание прототипа, тестирование, улучшение прототипа, внедрение прототипа.

3. Для чего используется тестирование прототипов?
   а) Для проверки функциональности и пользовательского опыта.
   б) Для определения целей прототипирования.
   в) Для анализа пользовательского опыта и исправления ошибок.
   г) Для проверки совместимости с другими системами.

Правильный ответ: а) для проверки функциональности и пользовательского опыта.

4. Какие инструменты можно использовать для создания прототипов?
   а) Adobe Photoshop, Sketch, InVision Studio.
   б) Adobe Photoshop, Sketch, InVision Studio, Figma.
   в) Adobe Photoshop, Sketch, InVision Studio, Figma, Axure RP.
   г) Adobe Photoshop, Sketch, InVision Studio, Figma, Axure RP, Balsamiq Mockups.

Правильный ответ: г) Adobe Photoshop, Sketch, InVision Studio, Figma, Axure RP, Balsamiq Mockups.

5. Что такое веб-приложение?
   а) Это набор статических веб-страниц, связанных гиперссылками.
   б) Это динамическая система, которая взаимодействует с пользователем через браузер.
   в) Это приложение, работающее на сервере и предоставляющее доступ к данным через веб-интерфейс.
   г) Это набор интерактивных компонентов, работающих на стороне клиента.
   Верный ответ: в) это приложение, работающее на сервере и предоставляющее доступ к данным через веб-интерфейс.
6. Какие основные компоненты входят в состав веб-приложения?

а) База данных, сервер приложений, веб-сервер, клиентское приложение.

б) База данных, сервер приложений, веб-сервер, клиентское приложение, интерфейс пользователя.

в) База данных, сервер приложений, веб-сервер, клиентское приложение, интерфейс пользователя, система управления контентом.

г) База данных, сервер приложений, веб-сервер, клиентское приложение, интерфейс пользователя, система управления базами данных.

Верный ответ: в) база данных, сервер приложений, веб-сервер, клиентское приложение, интерфейс пользователя, система управления контентом.

7. Какие типы веб-приложений существуют?

а) Статические, динамические, интерактивные, мультимедийные.

б) Статические, динамические, интерактивные, мультимедийные, мобильные.

в) Статические, динамические, интерактивные, мультимедийные, мобильные, облачные.

г) Статические, динамические, интерактивные, мультимедийные, мобильные, социальные.

Верный ответ: в) статические, динамические, интерактивные, мультимедийные, мобильные, облачные.

8. Что такое архитектура веб-приложения?

а) Структура и организация компонентов и модулей приложения.

б) Набор правил и стандартов, определяющих взаимодействие между компонентами приложения.

в) Процесс разработки и внедрения веб-приложения.

г) Методы и подходы к проектированию и реализации веб-приложений.

Верный ответ: а) структура и организация компонентов и модулей приложения.

9. Какие принципы проектирования веб-приложений следует учитывать при разработке?

а) Модульность, масштабируемость, безопасность, производительность.

б) Адаптивность, мобильность, интеграция, совместимость.

в) Функциональность, удобство использования, эстетика, юзабилити.

г) Интеграция, модульность, адаптивность, совместимость.

Верный ответ: в) функциональность, удобство использования, эстетика, юзабилити.

10. Как называется этап разработки, отвечающий за создание интерфейсов и части логики программы?

а) Аналитика.

б) Дизайн.

в) Бэкенд.

г) Фронтенд.

д) Тестирование.

Верный ответ: г) фронтенд.

ЗАДАНИЯ ОТКРЫТОГО ТИПА

Задание 1: Представьте с помощью паттернов программирования графических интерфейсов (MVC, MVVM, MVP) отображение списка товаров в интернет-магазине.

Решение: Используйте MVC (Model-View-Controller) архитектуру. Модель будет содержать список товаров, представление будет отображать товары в виде таблицы, а контроллер будет отвечать за получение данных из базы данных и передачу их представлению.

Задание 2: Представьте с помощью паттернов программирования графических интерфейсов (MVC, MVVM, MVP) регистрацию пользователя на сайте.

Решение: Используйте MVP (Model-View-Presenter) архитектуру. Модель будет содержать данные пользователя, представление будет формой регистрации, а (представитель) презентер будет обрабатывать ввод данных и передавать их модели.

Задание 3: Представьте с помощью паттернов программирования графических интерфейсов (MVC, MVVM, MVP) процедуру оформления заказа на сайте интернет-магазина.

Решение: Используйте MVVM (Model-View-ViewModel) архитектуру. Модель будет содержать данные о заказе, представление будет формой оформления заказа, а модель представления (ViewModel) будет связывать данные модели с элементами формы и предоставлять команды для обработки действий пользователя.

Задание 4: Представьте с помощью паттернов программирования графических интерфейсов (MVC, MVVM, MVP) процедуру отображения профиля пользователя на сайте.

Решение: Используйте MVVM (Model-View-ViewModel) архитектуру. Модель будет содержать данные о пользователе, представление будет отображать профиль пользователя, а модель представления (ViewModel) будет связывать данные модели с элементами представления и предоставлять команды для обновления данных профиля.

Задание 5: Представьте с помощью паттернов программирования графических интерфейсов (MVC, MVVM, MVP) процедуру отправки отзывов о товаре на сайте.

Решение: Используйте MVP (Model-View-Presenter) архитектуру. Модель будет содержать данные о товаре и отзывы, представление будет формой отправки отзыва, а (представитель) презентер будет обрабатывать ввод данных и передавать их модели.

Задание 6: Предложить решение для создания личного кабинета пользователя: разработать систему аутентификации и авторизации пользователей, предоставить возможность создания учётных записей, изменения паролей и управления настройками профиля.

Решение: использовать систему контроля версий, такую как Git, для отслеживания изменений кода и истории разработки. Также можно использовать фреймворк для аутентификации, например, Laravel Passport или Firebase Authentication.

Задание 7: Предложить решение для добавления функции поиска: разработать механизм поиска по сайту, который позволит пользователям находить информацию по ключевым словам или фразам.

Решение: использовать поисковые алгоритмы, такие как Lucene или отечественные решения, для индексации содержимого сайта и предоставления быстрого и точного поиска.

Задание 8: Предложить решение для реализации системы комментариев и отзывов: создать возможность оставлять комментарии и отзывы о товарах, услугах или событиях на сайте.

Решение: использовать систему управления комментариями, такую как Disqus или VK Comments, для обработки и отображения комментариев на сайте.

Задание 9: Предложить решение для интеграции с социальными сетями: добавить возможность авторизации через популярные социальные сети, чтобы пользователи могли быстро и удобно регистрироваться на сайте.

Решение: использовать API социальных сетей, таких как ВКонтакте, Telegram или Яндекс.Дзен, для интеграции с сайтом.

Задание 10: Предложить решение для создания системы уведомлений: разработать механизм отправки push-уведомлений или email-рассылок для информирования пользователей о новых событиях, акциях или новостях на сайте.

Решение: использовать сервисы уведомлений, такие как NotiSend или Mindbox, для отправки сообщений пользователям и отслеживания их эффективности.

Задание 11: Проанализируйте деятельность медицинского центра и создайте карту сайта.

Решение: Карта сайта медицинского центра может включать следующие разделы: «Услуги», «Специалисты», «Цены», «Контакты» и «О центре». Указанные разделы следует разместить в виде ссылок на главной странице сайта.

Задание 12: Проанализируйте деятельность и разработайте карту сайта для строительной компании.

Решение: Карта сайта строительной компании может включать следующие разделы: «Проекты», «Услуги», «Цены», «Контакты» и «Новости». Указанные разделы следует разместить в виде ссылок на главной странице сайта.

Задание 13: Проанализируйте деятельность и разработайте карту сайта для образовательного портала.

Решение: Карта сайта образовательного портала может включать следующие разделы: «Курсы», «Обучение», «Тесты», «Библиотека» и «Контакты». Указанные разделы следует разместить в виде ссылок на главной странице сайта.

Задание 14: Проанализируйте деятельность и разработайте карту сайта для развлекательного центра.

Решение: Карта сайта образовательного портала может включать следующие разделы: «Афиша», «Мероприятия», «Билеты», «Услуги» и «Контакты». Указанные разделы следует разместить в виде ссылок на главной странице сайта.

Задание 15: Проанализируйте деятельность и создайте карту сайта для туристической компании.

Решение: Карта сайта образовательного портала может включать следующие разделы: «Направления», «Отели», «Авиабилеты», «Экскурсии» и «Контакты». Указанные разделы следует разместить в виде ссылок на главной странице сайта.

Компетенция ПК-3. Способен разрабатывать цифровой программный продукт

Обучающийся знает: основы языка JavaScript; инструменты разработки Frontend приложений; основы разработки web-приложений на Vue.js, Angular, React; принципы и назначение фреймворков и библиотек (Vue.js, Angular, React); приемы написания минималистического кода с гибкой архитектурой.

Обучающийся умеет: разрабатывать web-приложения с помощью фреймворков и библиотек JavaScript (Vue.js, Angular, React); обеспечивать интеграцию программных модулей и компонентов web-приложений.

Обучающийся владеет: навыками построения приложений с Redux.js; навыками применения компонентного подхода при разработке web-приложений; навыками разработки одностраничных приложений (SPA); методами и инструментальными средствами интеграции программных модулей и компонентов web-приложений.

ЗАДАНИЯ ЗАКРЫТОГО ТИПА

1. Что является основным элементом в JavaScript?
   а) функция;
   б) объект;
   в) переменная;
   г) массив.
   Правильный ответ: в) переменная.

2. Какие типы данных существуют в JavaScript?
   а) логический, числовой, строковый;
   б) логический, числовой, строковый, объектный;
   в) логический, числовой, строковый, массивный;
   г) логический, числовой, строковый, объектный, массивный.
   Правильный ответ: б) логический, числовой, строковый, объектный.

3. Как объявляются переменные в JavaScript?
   а) var;
   б) let;
   в) const;
   г) none of the above.
   Правильный ответ: а) var.

4. Что такое замыкание в JavaScript?
   а) функция, которая возвращает другую функцию;
   б) функция, которая принимает другую функцию как аргумент;
   в) функция, которая вызывает другую функцию внутри своего тела;
   г) функция, которая возвращает массив.
   Правильный ответ: а) функция, которая возвращает другую функцию.

5. Что такое прототип в JavaScript?
   а) объект, который содержит методы и свойства класса;
   б) объект, который наследуется от другого объекта;
   в) функция, которая создаёт новый объект;
   г) ни один из вышеперечисленных.
   Правильный ответ: б) объект, который наследуется от другого объекта.

6. Что такое рекурсия в JavaScript?
   а) метод решения задач, основанный на повторении действий;
   б) процесс создания функций внутри функций;
   в) метод передачи аргументов функциям;
   г) ни один из вышеперечисленных.
   Правильный ответ: а) метод решения задач, основанный на повторении действий.

7. Что такое асинхронный код в JavaScript?
   а) код, который выполняется синхронно;
   б) код, который выполняется параллельно с другими задачами;
   в) код, который выполняется после завершения других задач;
   г) код, который выполняется в фоновом режиме.
   Правильный ответ: б) код, который выполняется параллельно с другими задачами.

8. Что такое JSON в JavaScript?
   а) язык программирования;
   б) формат сериализации данных;
   в) объектная модель документа;
   г) ни один из вышеперечисленных.
   Правильный ответ: б) формат сериализации данных.

9. Что такое фреймворк в JavaScript?
   а) Набор инструментов для создания веб-приложений.
   б) Библиотека функций для упрощения разработки.

в) Среда разработки для программистов.
г) Технология для создания мобильных приложений.
Правильный ответ: б) библиотека функций для упрощения разработки.

10. Что такое SPA (Single Page Application) в контексте фреймворков JavaScript?
а) Тип архитектуры веб-приложений.
б) Технология для создания интерактивных интерфейсов.
в) Метод оптимизации загрузки страниц.
г) Способ организации кода на стороне клиента.
Правильный ответ: а) тип архитектуры веб-приложений.

ЗАДАНИЯ ОТКРЫТОГО ТИПА

Задание 1: Создайте простой список задач на Vue.js: создайте компонент для отображения списка задач с возможностью добавления новой задачи и удаления выбранной задачи.
Решение:

```
Vue.component('task-list', {
 template: '<ul><li v-for="task in tasks" :key="task.id">{{ task.name }}</li></ul>',
 data() {
 return {
 tasks: []
 }
 },
 methods: {
 addTask(task) {
 this.tasks.push(task);
 },
 removeTask(task) {
 this.tasks.splice(this.tasks.indexOf(task), 1);
 }
 }
});

new Vue({
 el: '#app',
 data() {
 return {
 tasks: [
 { id: 1, name: 'Задача 1' },
 { id: 2, name: 'Задача 2' }
 ]
 }
 }
});
```

Задание 2: Создайте форму регистрации пользователя на Vue.js: создайте компонент формы регистрации с полями для ввода имени, электронной почты, пароля и подтверждения пароля.
Решение:

```
Vue.component('register-form', {
 template: '<form><input type="text" v-model="username"><br><input type="email" v-model="email"><br><input type="password" v-model="password"><br><input type="password" v-model="passwordConfirmation"><button>Зарегистрироваться</button></form>',
 data() {
```

```
return {
username: '',
email: '',
password: '',
passwordConfirmation: ''
}
},
methods: {
validateFields() {
if (this.password !== this.passwordConfirmation) {
return 'Пароль и подтверждение пароля не совпадают';
} else if (this.password.length < 6) {
return 'Пароль должен быть не короче 6 символов';
}
}
}
});

new Vue({
el: '#app',
data() {
return {
isValid: false
}
},
methods: {
onSubmit() {
this.isValid = this.$refs.registerForm.validateFields();
if (this.isValid) {
// Отправляем данные на сервер
} else {
// Отображаем ошибки
}
}
}
});
```

Задание 3: Создайте компонент «Обо мне» на Vue.js: создайте компонент AboutMe.vue с информацией о пользователе, такой как имя, профессия, образование и опыт работы.

Решение:

```
Vue.component('about-me', {
template: '<div><h1>Обо мне</h1><p>{{ user.name }} работает {{ user.profession }}.</p><p>Образование: {{ user.education }}.</p><p>Опыт работы: {{ user.experience }}</p></div>',
props: {
user: Object
}
});

new Vue({
el: '#app',
data() {
return {
user: {
```

```
name: 'Иван Иванов',
profession: 'Программист',
education: 'Университет информационных технологий',
experience: '5 лет опыта работы'
}
}
}
});
```

Задание 4: Создайте компонент «Комментарии» на Vue.js: создайте компонент Comments.vue с возможностью просмотра, добавления и удаления комментариев к записи.Решение:Vue.component('comments', {

```
template: '<div><h2>Комментарии</h2><ul><li v-for="comment in comments" :key="comment.id">{{ comment.text }}</li></ul><form v-on:submit.prevent="addComment"><input type="text" v-model="newCommentText"><button>Добавить комментарий</button></form></div>',
data() {
return {
comments: [],
newCommentText: ''
}
},
methods: {
addComment(event) {
event.preventDefault();
this.comments.push({ text: this.newCommentText });
this.newCommentText = '';
}
}
});

new Vue({
el: '#app',
data() {
return {
comments: [
{ id: 1, text: 'Первый комментарий' },
{ id: 2, text: 'Второй комментарий' }
]
}
}
});
```

Задание 5: Создайте компонент с именем AppComponent, который содержит метку, текстовое поле и заголовок на Angular

Решение:

```
import { Component } from '@angular/core';

@Component({
 selector: 'my-app',
 template: `
 <label>Введите имя:</label>
```

```
 <input [(ngModel)]="name" placeholder="name">
 <h1>Добро пожаловать {{name}}!</h1>
 `,
})
export class AppComponent {
 name = '';
}
```
Этот код создаёт.

Задание 6: Создайте простой счетчик на Vue.js
Решение:

```
<body>
	<div class="container pt-5" id="app">
		<div class="card center">
			<h1>Счетчик {{ counter }}</h1>
			<div>
				<button class="btn primary" v-on:click="counter++">+</button>
				<button class="btn danjer" v-on:click="counter--">-</button>
			</div>
		</div>
	</div>

	<script src="https://unpkg.com/vue@next"></script>
	<script type="text/javascript">
		const App = {
			data(){
				return{
					counter:0
				}
			}
		}
		const app=Vue.createApp(App).mount('#app')
		</script>
	</body>
```

Задание 7: Опишите основные способы добавления Vue.js в проект

Решение:

Vue.js изначально разрабатывался быть инкрементально адаптируемым. Это значит, что он может быть интегрирован в проект несколькими способами, в зависимости от требований.

Есть четыре основных способа добавления Vue.js в проект:

1. Импорт CDN пакета на странице
2. Скачивание файлов JavaScript и их самостоятельный хостинг
3. Установка с использованием npm
4. Использование официального CLI для развёртывания проекта, предоставляющего продвинутые инструменты для создания современного рабочего процесса разработки фронтенда (например, горячая перезагрузка, линтинг при сохранении и многое другое).

Задание 8: В чем особенности особенности Redux.js?

Ответ: Особенности Redux.js:

1. Управление состоянием данных: Redux предназначен для управления состоянием данных в приложениях JavaScript.

2. Небольшой размер: Redux занимает около 2 КБ, включая расширения.
3. Прямой доступ к состоянию приложения: компоненты могут иметь прямой доступ к состоянию приложения без необходимости отправлять props или использовать функции обратного вызова.
4. Логика потока данных: Redux обеспечивает бесшовную логику потока данных.
5. Предсказуемость результатов: благодаря одному источнику данных (хранилищу) разработчики могут легко синхронизировать состояние компонентов с действиями и другими частями приложения.
6. Простота тестирования: Redux использует чистые функции, которые изолированы друг от друга, что облегчает тестирование кода.

Задание 9: Создайте на React компонент App, который содержит кнопку.

Решение:
При нажатии на кнопку вызывается функция handleClick, которая обрабатывает событие нажатия.
напиши код компонента на React, который отвечает за нажатие кнопки
Вот пример кода компонента на React, который отвечает за нажатие кнопки:

```
import * as React from 'react';

const App = () => {
 const handleClick = () => {
 // реализация обработки нажатия кнопки
 };

 return (
 <div>
 <button type="button" onClick={handleClick}>Нажмите меня</button>
 </div>
 );
};

export default App;
```

Задание 10: приведи пример шаблона на Angular

Решение:
Пример шаблона на Angular:

```
import { Component } from '@angular/core';

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent {
 title = 'Angular Template';
}
```

Этот шаблон создаёт базовый компонент приложения с текстом «Angular Template» в качестве заголовка.

Задание 11: За что отвечает директива NgForm. ReactiveForms? Приведите пример.

Решение:

Директива NgForm отвечает за создание и управление формой в Angular. Она используется для создания формы, которая не связана с моделью данных, и для передачи данных формы в родительский компонент. ReactiveForms — это подход к созданию форм в Angular, основанный на потоках observable, который обеспечивает синхронный доступ к модели данных, неизменяемость с помощью наблюдаемых операторов и отслеживание изменений с помощью наблюдаемых потоков.

приведи прмиер NgForm. ReactiveForms

Пример использования NgForm и ReactiveForms в Angular:

Импортируйте необходимые модули:

```
import { Component, NgForm } from '@angular/core';
import { ReactiveFormsModule } from '@angular/forms';
```

Создайте компонент:

```
@Component({
 selector: 'my-form',
 templateUrl: './my-form.component.html',
 styleUrls: ['./my-form.component.css'],
 imports: [ReactiveFormsModule]
})
export class MyFormComponent {
 form: NgForm;

 constructor() { }
}
```

Создайте форму в шаблоне компонента:

```
<form #form="ngForm">
 <input type="text" name="name" [(ngModel)]="name" />
 <button type="submit">Отправить</button>
</form>
```

Теперь вы можете использовать форму в компоненте, например, для обработки отправки данных:

```
export class MyFormComponent {
 form: NgForm;

 constructor() { }

 onSubmit() {
 // Обработка отправки данных
 }
}
```

Задание 12: приведи пример onPush в Angular

Пример использования стратегии обнаружения изменений OnPush в Angular:

```
import { ChangeDetectionStrategy, Component } from '@angular/core';

@Component({
 changeDetection: ChangeDetectionStrategy.OnPush,
})
export class MyComponent {
}
```

В этом примере устанавливаем стратегию обнаружения изменений компонента на OnPush с помощью декоратора @Component. Теперь Angular будет выполнять обнаружение

изменений только для поддерева компонентов, если корневой компонент поддерева получает новые входные данные или обрабатываются события.

Задание 13: Приведите пример анимации в React, CSSTransitionGroup

Решение:
Пример использования CSSTransitionGroup для анимации в React:

```
import React, { useState } from 'react';
import { CSSTransitionGroup } from 'react-transition-group';

function App() {
 const [count, setCount] = useState(0);

 function handleClick() {
 setCount(prevCount => prevCount + 1);
 }

 return (
 <div>
 <h1>React CSSTransitionGroup</h1>
 <p>Нажмите на кнопку, чтобы увеличить счётчик.</p>
 <button onClick={handleClick}>+</button>
 <CSSTransitionGroup
 transitionName="example"
 transitionEnterTimeout={500}
 transitionLeaveTimeout={500}>
 {Array.from({ length: count }, (_, i) => (
 <div key={i}>
 {i + 1}
 </div>
 ))}
 </CSSTransitionGroup>
 </div>
 );
}

export default App;
```

В этом примере используется CSSTransitionGroup для анимации элементов списка при их входе и выходе из DOM, определяются переходные классы в файле CSS и передаются их имена и время ожидания перехода в props компонента CSSTransitionGroup.

Задание 14: В чем отличие React.js от других популярных фреймворков.
Ответ:
Отличие React.js от других популярных фреймворков заключается в следующем:

1. React.js разработан компанией Facebook и фокусируется на создании компонентно-ориентированных приложений, в то время как Vue.js создан Эваном Ю и предназначен для создания простых и гибких приложений.
2. React.js использует реактивную модель, которая автоматически обновляет представление при изменении данных модели, в то время как Vue.js также использует реактивную модель.

3. React.js использует алгоритм Virtual DOM, который минимизирует количество изменений в реальном DOM, повышая производительность приложения, в то время как Vue.js не имеет такого алгоритма.
4. React.js использует JSX, что позволяет разработчикам писать компоненты в HTML-подобном синтаксисе, облегчая визуализацию компонентов и понимание их структуры, в то время как Vue.js использует обычный JavaScript.

Задание 15: Приведите пример объекта Observable и библиотека RxJS
Решение:
Пример объекта Observable и использование библиотеки RxJS:
Определим класс User:

```
export class User {
 name: string;
 age: number;
}
```

Создайте сервис для получения данных из API:

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { map } from 'rxjs/operators';

@Injectable()
export class HttpService {
 constructor(private http: HttpClient) {}

 getUsers(): Observable<User[]> {
 return this.http.get('users.json').pipe(
 map((data) => {
 let usersList = data['userList'];
 return usersList.map(function (user: any) {
 return {
 name: user.userName,
 age: user.userAge,
 };
 });
 })
 );
 }
}
```

Используем сервис в компоненте:

```
import { Component, OnInit } from '@angular/core';
import { HttpService } from './http.service';
import { User } from './user';

@Component({
 selector: 'my-app',
 template: `
 <ul>
 <li *ngFor="let user of users">
 <p>Имя пользователя: {{ user?.name }}</p>
 <p>Возраст пользователя: {{ user?.age }}</p>
 </li>
```

```
</ul>
`,
providers: [HttpService],
})
export class AppComponent implements OnInit {
users: User[];

constructor(private httpService: HttpService) {}

ngOnInit() {
this.httpService.getUsers().subscribe((data) => (this.users = data));
}
}
```

В этом примере создан сервис HttpService, который возвращает Observable<User[]> с данными пользователей из JSON файла users.json. Затем используется этот сервис в компоненте AppComponent, где данные пользователей отображаются в списке через цикл *ngFor.

Компетенция ПК-5. Способен обеспечивать качество работы цифрового программного продукта

Обучающийся знает: современные инструменты и практики тестирования JavaScript-кода; процесс интеграции программного обеспечения и интеграцию разработки API.

Обучающийся умеет: проводить интеграционное тестирование кода с помощью различных инструментов.

Обучающийся владеет: навыками создания среды для запуска тестов.

ЗАДАНИЯ ЗАКРЫТОГО ТИПА

1. Какой из следующих фреймворков предназначен для тестирования приложений React?

а) Jasmine;
б) Mocha;
в) Jest;
г) Nightwatch.
Правильный ответ: в) Jest.

2. Какая из перечисленных платформ тестирования поддерживает интеграционное тестирование?

а) Jasmine;
б) Karma;
в) Puppeteer;
г) Mocha.
Правильный ответ: б) Karma.

3. Что относится к целостности кода в контексте тестирования JavaScript?

а) Достижение желаемого результата;
б) Оптимизация кода;
в) Обеспечение функциональности;
г) Все вышеперечисленное.
Правильный ответ: г) Все вышеперечисленное.

4. Какие типы тестирования используются в программном обеспечении?

а) Модульное и интеграционное;
б) Функциональное и интеграционное;
в) Модульное и функциональное;
г) Интеграционное и функциональное.

Правильный ответ: в) Модульное и функциональное.

5. Что является основной функцией платформы тестирования?

а) Автоматическое выполнение тестов;
б) Отображение результатов тестов;
в) Управление тестами;
г) Всё вышеперечисленное.
Правильный ответ: г) Всё вышеперечисленное.

6. Какой из следующих фреймворков предназначен для тестирования кода, написанного на языке TypeScript?

а) Jasmine;
б) Mocha;
в) Cypress;
г) Nightwatch.
Правильный ответ: в) Cypress.

7. Что относится к оптимизации кода в контексте тестирования JavaScript?

а) Сокращение времени выполнения кода;
б) Улучшение производительности;
в) Обеспечение стабильности;
г) Всё вышеперечисленное.
Правильный ответ: г) Всё вышеперечисленное.

8. Зачем нужна интеграция API?

а) Для автоматизации задач и интеграции новых систем.
б) Для обмена данными и функциональными возможностями между системами.
в) Для повышения эффективности и масштабируемости.
г) Для экономии средств и сокращения количества ошибок.
Правильный ответ: б) Для обмена данными и функциональными возможностями между системами.

9. Какие существуют способы достижения интеграции API?

а) Пользовательская интеграция и интеграция на основе стандартов.
б) Интеграция на основе открытых интерфейсов и интеграция на основе проприетарных интерфейсов.
в) Интеграция на основе RESTful API и интеграция на основе SOAP API.
г) Всё вышеперечисленное.
Правильный ответ: г) Всё вышеперечисленное.

10. Что такое пользовательская интеграция API?

а) Процесс подключения одного приложения или службы к другому через пользовательский API.
б) Разработка нового API для обмена данными и функциональными возможностями между системами.
в) Использование существующих API для интеграции систем.
г) Всё вышеперечисленное.
Правильный ответ: а) Процесс подключения одного приложения или службы к другому через пользовательский API.

ЗАДАНИЯ ОТКРЫТОГО ТИПА

Задание 1: Опишите алгоритм проведения тестирования интеграции между модулем, который взаимодействует с базой данных, и API.
Решение:
Алгоритм проведения тестирования интеграции между модулем, который взаимодействует с базой данных, и API:

1. Установите библиотеки Nock и Jest.
2. Настройте mock-сервер с помощью Nock, перехватывая запросы к API и отвечая на них фиктивными данными.
3. Напишите тест Jest, который отправляет запросы к модулю и проверяет соответствие полученных данных ожидаемым значениям.
4. Проведите тестирование, убедитесь, что интеграция работает корректно и данные передаются между модулем и API без ошибок.

Задание 2: Напишите тест Jest для тестирования интеграции между модулем, который взаимодействует с базой данных, и API.
Решение:
Тест Jest:const axios = require('axios');
const nock = require('nock');

```
describe('Integration Testing', () => {
 beforeEach(() => {
 // Настройка mock-сервера с помощью Nock
 nock('https://api.example.com')
 .get('/users')
 .reply(200, { users: [] });
 });

 it('Проверка взаимодействия с базой данных и API', async () => {
 const response = await axios.get('https://api.example.com/users');
 expect(response.status).toBe(200);
 expect(response.data.users).toEqual([]);
 });
});
```

В этом тесте используется Jest для написания интеграционного теста, который отправляет GET-запрос к API и проверяет, что ответ содержит массив пользователей. Также используем Nock для настройки mock-сервера, который возвращает фиктивные данные.

Задание 3: Назовите три основных метода вызова API.
Ответ: XMLHttpReques, Fetch, Axios.

Задание 4: Для чего служит JS-метод fetch()?

Ответ:
JS-метод fetch() используется для запроса к серверу и загрузки данных на веб-страницах. Fetch API — такой же простой, интуитивно понятный интерфейс, как и XMLHttpRequest, применяемый для асинхронного использования ресурсов. Основное отличие заключается в том, что в настоящее время Fetch работает с промисами, а не с обратными вызовами. Разработчики JavaScript стали отказываться от обратных вызовов после появления промисов. Fetch API очень просто использовать. Нужно просто передать URL, путь к нужному ресурсу, методу fetch():

 Fetch API
В качестве параметра в fetch() передается маршрут к необходимому ресурсу. Он возвращает промис, который при выполнении передает ответ в then(). Метод catch() перехватывает ошибки, если запрос не удается завершить из-за сбоя в сети или по какой-либо другой причине.

Задание 5: В чем назначение структурных директив ngIf, ngFor, ngSwitch
Ответ:
Назначение структурных директив ngIf, ngFor и ngSwitch заключается в изменении структуры DOM путём добавления или удаления html-элементов.
ngIf позволяет удалить или отобразить элемент при определённом условии.
ngFor позволяет перебрать элементы массива в шаблоне.
ngSwitch позволяет встроить в шаблон конструкцию switch…case и выводить тот или иной блок в зависимости от результата выполнения.

Задание 6: Приведите пример взаимодействия с пользователем, HostListener
Решение:
Пример использования HostListener для взаимодействия с пользователем в Angular:

```
import {Directive, ElementRef, Renderer2, HostListener} from "@angular/core";

@Directive({
 selector: "[bold]",
 standalone: true
})
export class BoldDirective {
 constructor(private element: ElementRef, private renderer: Renderer2) { }

 @HostListener("mouseenter") onMouseEnter() {
 this.setFontWeight("bold");
 }

 @HostListener("mouseleave") onMouseLeave() {
 this.setFontWeight("normal");
 }

 private setFontWeight(val: string) {
 this.renderer.setStyle(this.element.nativeElement, "font-weight", val);
 }
}
```

В этом примере используется директива BoldDirective, которая содержит декораторы HostListener для обработки событий mouseenter и mouseleave. При наведении указателя мыши на элемент стиль font-weight изменяется на «bold», а при отводе указателя — сбрасывается до значения «normal».

Задание 7: Приведите пример взаимодействия с пользователем HostBinding
Решение:
Пример использования HostBinding для взаимодействия с пользователем в Angular:

```
import {Directive, ElementRef, Renderer2, HostBinding} from "@angular/core";

@Directive({
 selector: "bold",
```

```
 standalone: true
})
export class BoldDirective {
 constructor(private element: ElementRef, private renderer: Renderer2) { }

 @HostBinding("style.fontWeight") get getFontWeight() {
 return this.fontWeight;
 }

 @HostBinding("style.cursor") get getCursor() {
 return "pointer";
 }

 @HostListener("mouseenter") onMouseEnter() {
 this.fontWeight = "bold";
 }

 @HostListener("mouseleave") onMouseLeave() {
 this.fontWeight = "normal";
 }

 private fontWeight = "normal";
}
```

В этом примере используется директива BoldDirective, которая содержит декораторы HostBinding и HostListener для взаимодействия с пользователем. Декоратор @HostBinding связывает свойство style.fontWeight элемента с внутренним свойством fontWeight директивы. Декоратор @HostListener обрабатывает события mouseenter и mouseleave, устанавливая соответствующее значение для свойства fontWeight.

Задание 8: В чем назначение POST-запросов и првиедите пример кода на JavaScript?
Решение:
POST-запросы используются для отправки данных на сервер для обработки, обновления или создания новых ресурсов. Вот пример кода на JavaScript для отправки POST-запроса с использованием Fetch API:

```
const data = { name: "John", age: 30 };

fetch("https://api.example.com/data", {
 method: "POST",
 headers: { "Content-Type": "application/json" },
 body: JSON.stringify(data)
}).then((response) => response.json()).then((data) => {
 console.log("Success:", data);
}).catch((error) => {
 console.error("Error:", error);
});
```

В этом примере отправляем POST-запрос на URL https://api.example.com/data, передавая данные в формате JSON в теле запроса.

Задание 9: Приведите пример использования react-router
Решение:
Пример использования React Router:

```
import React from 'react';
import { BrowserRouter as Router, Route, Link } from 'react-router-dom';
```

```
function Home() {
 return <h2>Домашняя страница</h2>;
}

function Kitchen() {
 return <h2>Кухня</h2>;
}

function LivingRoom() {
 return <h2>Гостиная</h2>;
}

function Bedroom() {
 return <h2>Спальня</h2>;
}

function App() {
 return (
 <Router>
 <div>
 <nav>
 <ul>
 <li><Link to="/">Домой</Link></li>
 <li><Link to="/kitchen">Кухня</Link></li>
 <li><Link to="/living-room">Гостиная</Link></li>
 <li><Link to="/bedroom">Спальня</Link></li>
 </ul>
 </nav>
 {/* Определение маршрутов */}
 <Route path="/" exact component={Home} />
 <Route path="/kitchen" component={Kitchen} />
 <Route path="/living-room" component={LivingRoom} />
 <Route path="/bedroom" component={Bedroom} />
 </div>
 </Router>
 );
}

export default App;
```

В этом примере используется React Router для управления навигацией по веб-приложению. Компоненты Home, Kitchen, LivingRoom и Bedroom соответствуют различным страницам или функциям приложения. С помощью элементов <Link> и <Route> определяется навигация между этими страницами.

Задание 10: Приведите пример создания атрибутивных директив на Angular
Решение:
Пример создания атрибутивной директивы на Angular:
Создайте новый файл с расширением «.ts» в папке «src/app». Назовите его «bold.directive.ts».
Определите директиву внутри файла «bold.directive.ts»:

```
import {Directive, ElementRef} from "@angular/core";

@Directive({
```

```
 selector: "[bold]",
 standalone: true
})
export class BoldDirective {
 constructor(private elementRef: ElementRef) {
 this.elementRef.nativeElement.style.fontWeight = "bold";
 }
}
```
Импортируйте директиву в главный компонент:
```
import { Component } from "@angular/core";
import { BoldDirective } from "./bold.directive";

@Component({
 selector: "my-app",
 standalone: true,
 imports: [BoldDirective]
})
export class AppComponent {}
```
Используйте директиву в шаблоне главного компонента:
```
<div>
 <p bold>Hello Angular</p>
 <p>Приложение Angular состоит из компонентов</p>
</div>
```
Теперь при использовании директивы «bold» в шаблоне главного компонента, текст будет отображаться жирным шрифтом.
Найти в Яндексе
Источник

Задание 11: Приведите пример продвинутые элементы API React.JS
Решение:
К продвинутым элементам API React относятся:
React.memo — создание оптимизированных компонентов, основанных на сравнении старых и новых пропсов и состояния.
React.forwardRef — передача ссылки на компонент в качестве аргумента для передачи пропса.
React.Suspense — задержка загрузки компонентов до получения данных из API.
React.lazy — динамическая загрузка компонентов с использованием React.lazy.
React.createRef — создание ссылки на элемент React.
React.useTransition — управление переходами между состояниями компонентов.
Пример использования React.memo:
```
class Greeting extends React.Component {
 render() {
 return <h1>Привет, {this.props.name}</h1>;
 }
}

const MemoizedGreeting = React.memo(Greeting);
```
В этом примере создаётся оптимизированный компонент MemoizedGreeting, основанный на сравнении старых и новых пропсов и состояния компонента Greeting.

Задание 12: Опишите назначение Redux
Решение:
Назначение Redux заключается в управлении состоянием данных и пользовательским интерфейсом в приложениях JavaScript с большим количеством сущностей. Redux используется

в связке с фреймворками для JavaScript, такими как React, TypeScript, Vue, Angular и другими. Он предназначен для удобной замены встроенных средств работы с состоянием, облегчения масштабирования приложений, избавления от ошибок, связанных с беспорядком в объекте состояния, и повышения производительности и работоспособности программ.

Задание 13: Приведите пример использования структурной директивы ngIf:
Решение:
NgIf:

```
<div *ngIf="condition">
 <p>Текст или элемент, который будет отображаться при истинном значении условия.</p>
</div>
```

В этом примере директива ngIf проверяет условие и отображает или скрывает элемент в зависимости от его значения.

Задание 14: Приведите пример использования структурной директивы ngFor:
Решение:
NgFor:

```
<ul>
 <li *ngFor="let item of items">
 {{item}}
 </li>
</ul>
```

В этом примере директива ngFor перебирает массив items и выводит каждый элемент в списке.

Задание 15: Приведите пример использования структурной директивы ngSwitch:
Решение:
NgSwitch:

```
<div ngSwitch>
 <div *ngSwitchCase="trueValue">
 <p>Элемент 1</p>
 </div>
 <div *ngSwitchCase="falseValue">
 <p>Элемент 2</p>
 </div>
 <div *ngSwitchDefault>
 <p>Элемент 3</p>
 </div>
</div>
```

В этом примере директива ngSwitch отображает один из указанных элементов в зависимости от значения переменной trueValue или falseValue.

6. ЛИСТ СОГЛАСОВАНИЯ

Составил:

О.И. Подулыбина, ст.преп. 

(подпись)

Заведующий кафедрой

Н.Б. Стрекалова, д.п.н., доцент 

(подпись)

Заведующий выпускающей кафедрой

Н.Б. Стрекалова, д.п.н., доцент 

(подпись)