# Описание

Для работы с react необходимо установить node js. Через команду «npx create-react-app WebLesson» создастся новый проект со стандартной структурой create react app.

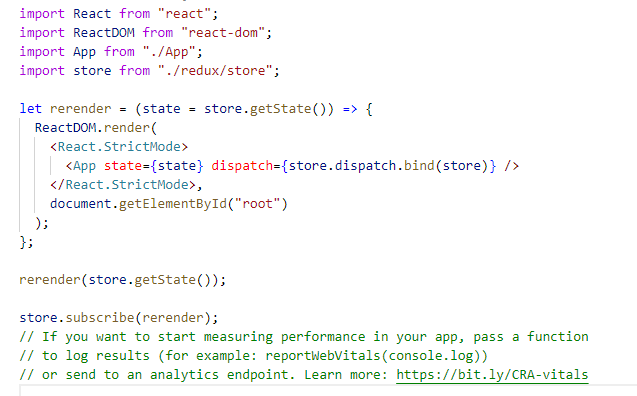
Вся разработка ведется в папке «src». Входной файл «index.js», запускает react и делает первичную отрисовку «app.js» (рис. 3.1).

Рис. 3.1 Содержимое файла «index.js»

К файлу «app.js» можно подключить css файл, в котором будут прописаны стили отображения всех элементов приложения. «app.js» по сути большая компонента которая содержит в себе несколько поменьше (рис. 3.2).

Все компоненты кроме app будут иметь расширение «jsx» вместо обычного «js». Это позволит удобно писать jsx разметку в среде разработки visual studio code. IDE будет понимать когда включать синтаксис html а когда обычный js. Все такие компоненты будут находиться в папке «moduls»

Глобально приложение можно разделить на 2 части: меню и контент. Соответственно можно создать 2 компоненты «menu.jsx» и «content.jsx».

Рис. 3.2 Содержимое компоненты «app.js»

Все компоненты кроме app будут иметь расширение «jsx» вместо обычного «js». Это позволит удобно писать jsx разметку в среде разработки visual studio code. IDE будет понимать когда включать синтаксис html а когда обычный js. Все такие компоненты будут находиться в папке «moduls»

Глобально приложение можно разделить на 2 части: меню и контент. Соответственно можно создать 2 компоненты «menu.jsx» и «content.jsx».

Компонент «menu.jsx» можно поделить на меню с разделами «menuTheme.jsx» и главами по разделам «menuStep.jsx». их содержимое генерируется на ходу, в зависимости от state.

Для того чтобы добавить возможность свободно расширять содержимое программы, добавляя новые разделы и подразделы, необходимо вывести все данные о них в отдельный файл. Этот файл называется «theme1.json» и находится он в папке public, для того чтобы удобно было к нему добраться после работы webpack в конечном electron приложении.

Библиотека react предназначена для отрисовки страницы, за ее наполнение отвечает сторона бизнес логики (BLL). Суть бизнес логики в том, чтобы взять данные из файла «theme1.json» и перенести в state, отвечающий за состояние страницы. В react существует внутренний локальный state, но есть проблема с их использованием при работе с функциональными компонентами. Для более удобной работы со state используется библиотека «redux».

Все дополнительные инструменты скачиваются и интегрируются в проект с помощью пакетного менеджера «npm».

Необходимо получить данные из «theme1.json» и поместить их в компоненты «menu» и «content». Так как компонента - это функция которая возвращает jsx разметку, то это значит, что туда можно передать дополнительные параметры «props». В props’ах можно передать текущее состояние компоненты «state». В дальнейшем программа при взаимодействии с пользователем должна будет менять свой внешний вид, а значит, что react должен перерисовать изменившиеся компоненты с обновленным state. На самом деле при взаимодействии с элементом меняется не сам компонент, а его state. И уже в зависимости от state, react сам перерисовывает элементы. State меняется с помощью dispatch.

Итак, в компонент «menu» параметром приходят state со всеми данными. Задача данной компоненты отрисовать элементы на экране (рис. 3.3).

Слева можно увидеть разделы, а слева поле для поиска, шторку для изменения параметров шрифта, кнопку для скрытия меню, переключатель для изменения цветовой темы и внизу подразделы выбранного раздела.

За отрисовку разделов отвечает компонента «MenuTheme». За отрисовку подразделов отвечает компонента «menuStep».

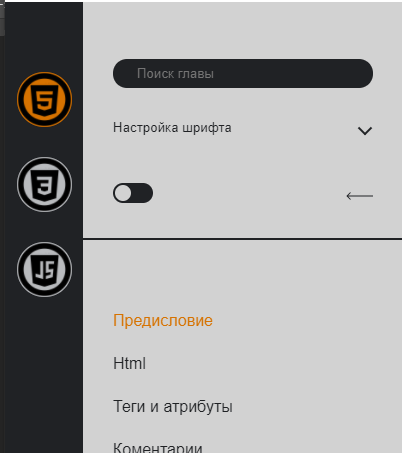


Рис. 3.3 Итог отрисовки компоненты «menu.jsx»

Все input’ы и кнопки, запускает dispatch, который принимает в параметры объект с обязательным свойством «type». Благодаря этому свойству программа понимает, какой сценарии изменения state, нужно запустить. Все сценарии содержаться в reducer’ах. Reducer’ов всего 2, как и глобальных компонентов.

Пример работы метода dispatch можно рассмотреть на кнопке скрытия меню. По нажатию на данную стрелочку выполняется функция (рис. 3.4).

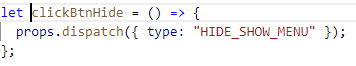


Рис. 3.4 Функция кнопки скрытия меню

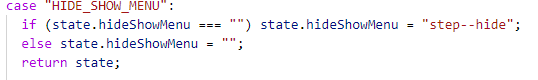
Тем временем, в reducer меню «menuReducer.js» выполняется следующий сценарий (рис. 3.5).

Рис. 3.5 Пример изменения state

По такому же принципу работают все интерактивные элементы.

«menuTheme.jsx» как параметр принимает название раздела и возвращает ее разметку. С помощью цикла map программа проходиться по массиву всех разделов и при каждой итерации вызывает данный компонент (рис. 3.6).

Рис. 3.6 Вызов «menuTheme.jsx»

«menuStep.jsx» похожим образом принимает параметр название подраздела и возвращает ее разметку (рис. 3.7).

Рис. 3.7 Вызов «menuStep.jsx»

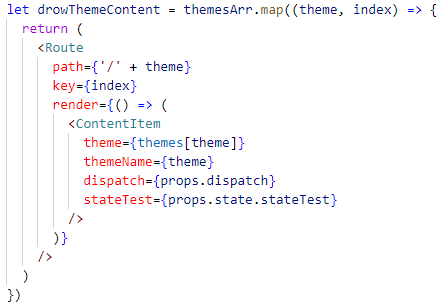
Content генерирует свое наполнение благодаря функции «drowThemeContent» (рис. 3.8).

Рис. 3.8 Функция «drowThemeContent»

Создается новый route. Система route’ов имитирует переход на другую страницу. React рисует интерфейсы не перезагружает страницу, рисуя все в одном единственном html файле, поэтому остается лишь пользоваться route’ми для логичного удобного разделения контента.

Задача route поменять содержимое страницы в зависимости от url. С помощью navLink’ов можно менять url.

На (рис. 3.8) видно, что вызывается компонента «ContentItem», которая принимает в качестве параметра данные выбранного раздела и возвращает готовую разметки целой страницы.

«ContentItem» пробегает по содержимому раздела, разбивая ее на подразделы и после разбивая и их на элементы, например абзац, определение, картинка или видео youtube. Делает она это с помощью функции «contentParse» (рис. 3.9).

Рис. 3.9 Функция «contentParse»

В файле «theme1.json» данные хранятся в виде текста. Чтобы конвертировать элементы из строк в html разметку, необходимо сначала узнать какой конечный элемент нужно узнать. Для этого используется функция «stepItemsParse» (рис. 3.10).

Рис. 3.10 Функция «stepItemsParse»

После определения типа данных, необходимо его отрисовать.

Если элемент является картинкой, то запускается сценарий функции «imgParse» (рис. 3.11).



Рис. 3.11 Функция «imgParse»

Если элемент является ссылкой, то запускается сценарий функции «linkParce» (рис. 3.12).



Рис. 3.12 Функция «linkParce»

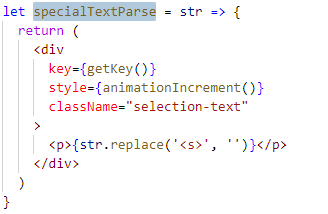
Если элемент является определением, то запускается сценарий функции «specialTextParse» (рис. 3.13).

Рис. 3.13 Функция «specialTextParse»

Если элемент является ссылкой на видео youtube, то запускается сценарий функции «youtubeVideoParse» (рис. 3.14).

Рис. 3.14 Функция «youtubeVideoParse»

И наконец если это просто абзац, то возвращается соответствующая разметка без использования дополнительной функции.

В результате можно увидеть контент (рис. 3.15).

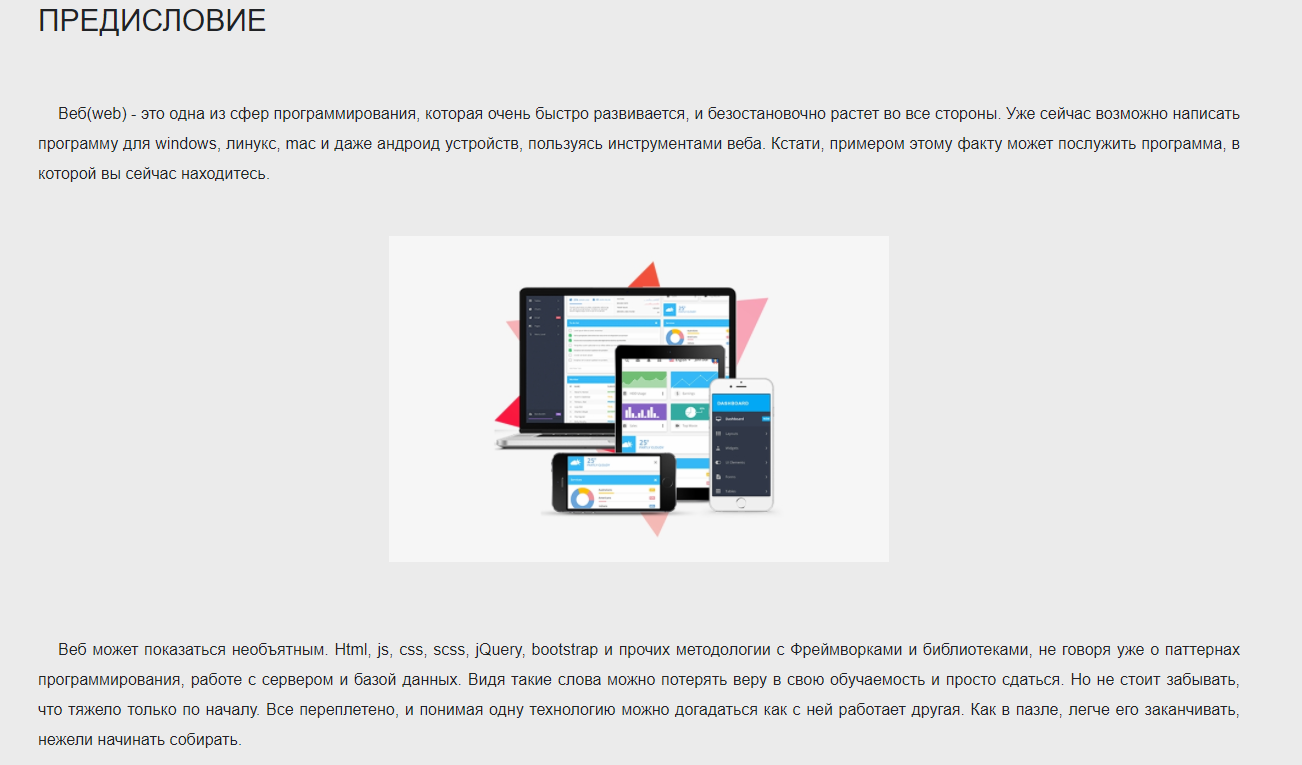
Программа так же поддерживает тесты. Чтобы их добавить их нужно создать объект в файле с данными (рис. 3.16).

Если в разделе находиться такой объект, то создается определенная система route’ов. Она позволит отрисовать вопросы при нажатии на кнопку (рис. 3.17).

В целом программа выглядит так (рис. 3.18).

Так программа выглядит с темной темой и скрытым меню (рис. 3.19).

При наведении на тему будет отображаться ее название (рис. 3.21).



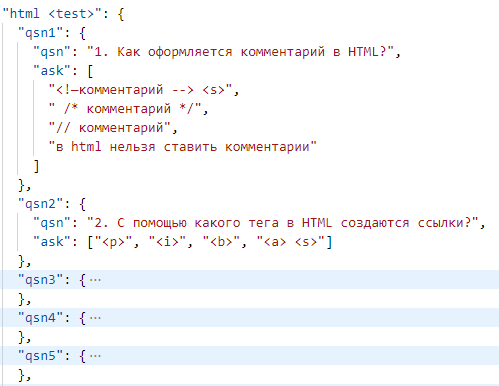
Рис. 3.15 Результат парсинга

Рис. 3.16 Заполнение теста



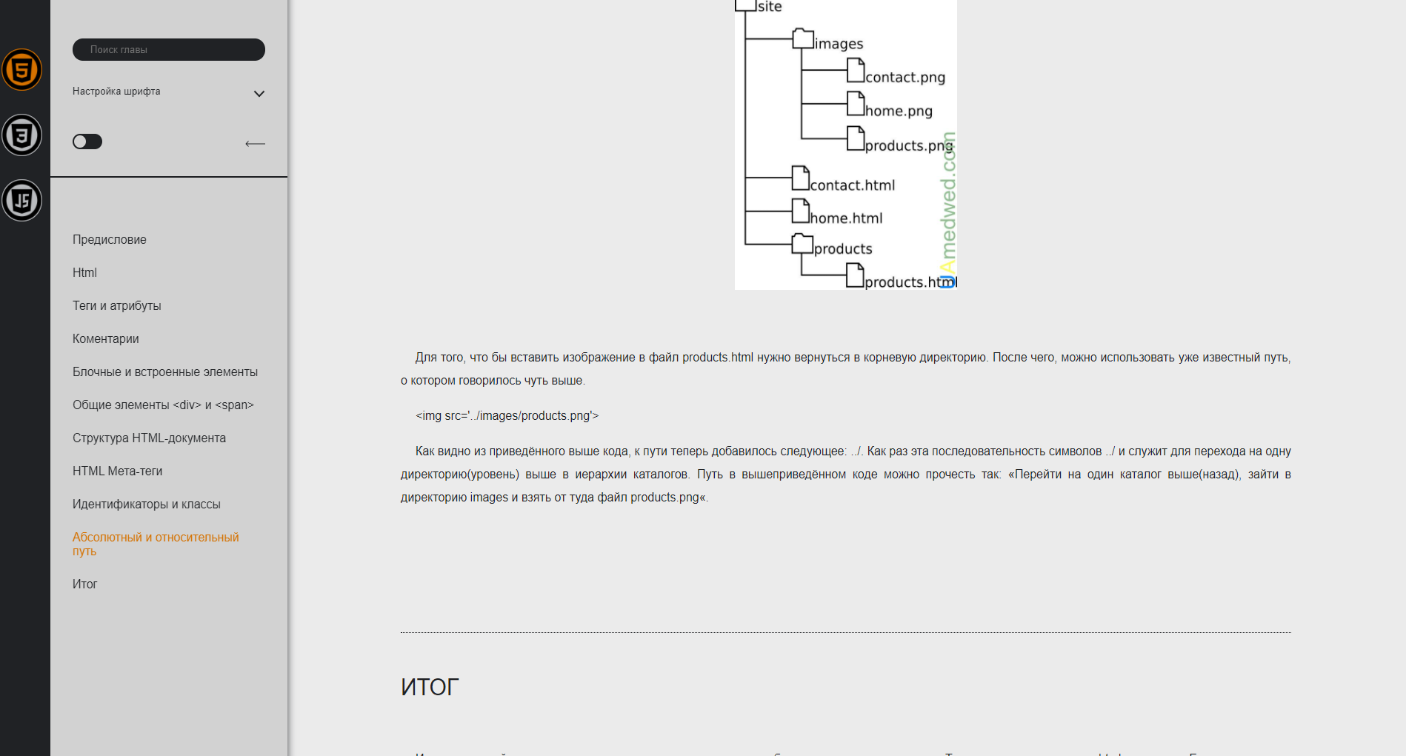
Рис. 3.17 Кнопка теста

Рис. 3.18 Как выглядит программа в результате

Рис. 3.21 Наведение на тему

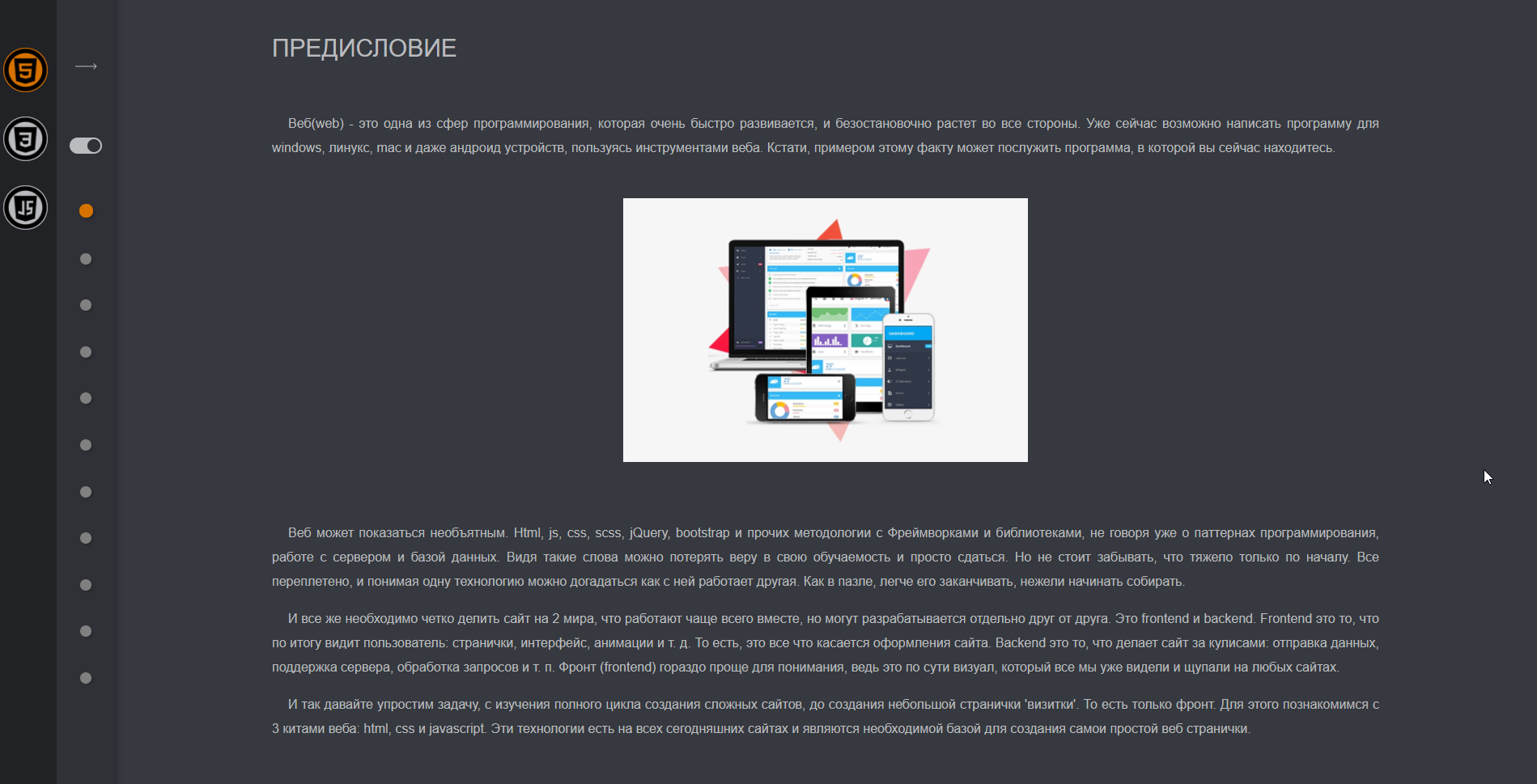


Рис. 3.19 Темная тема

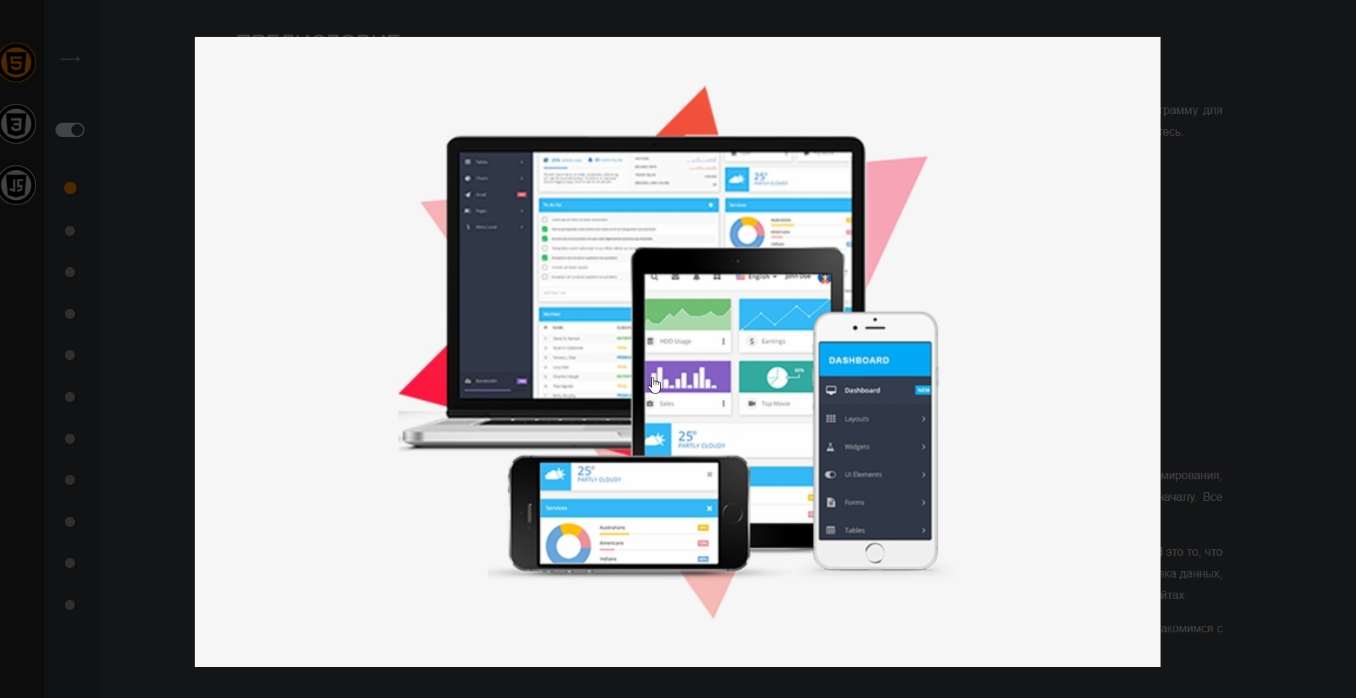
При нажатии на картинку она развернется на полный экран, для более удобного просмотра (рис. 3.20).

Рис. 3.20 Расширенная фотография

Так же в программе реализованы подсказки. При наведении на любой интерактивный объект и по истечении 1 секунды появиться подсказка о том, что эта кнопка делает (рис. 3.22).

Рис. 3.22 Подсказка

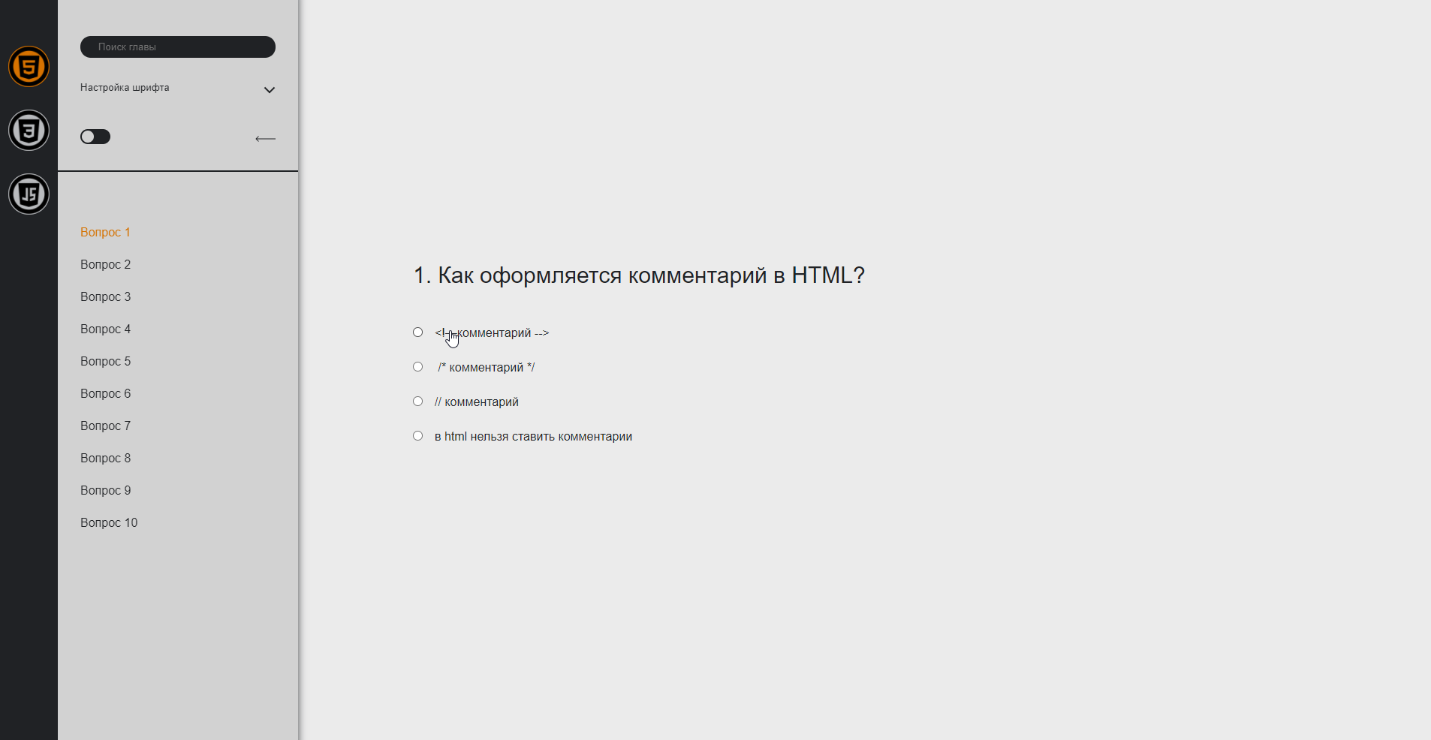
Страница с тестами выглядит так (рис. 3.23).

Рис. 3.23 Страница с вопросами