Модуль 5

Мы уже выяснили, что сайты состоят из набора HTML-страниц и CSS-стилей и что каждый раз, когда мы открываем новую страницу, сервер браузера с помощью HTTP отправляет нам эту страницу. В целом схема действительно такая, но современные сайты бывают очень сложными. Они не только предоставляют нам какую-то информацию, но и умеют обрабатывать информацию от пользователей.

Например, когда вы заходите на сайт для покупки авиабилетов, вам нужно не только посмотреть, куда можно улететь, но и узнать цену билета и купить его. Получается, на самом деле, сайт состоит из двух частей: красивых страниц и какой-то логики. Как мы уже знаем, первая часть называется *frontend,* а вторая часть — *backend.*

Для разработки **backend-части** чаще всего используются: JavaScript (да, всё тот же, но немного в другом окружении, которое называется Node.js), PHP, Python, Java и множество других языков. На этой части сайта работает вся основная логика: обработка запросов, сохранение данных, введённых пользователем, и многое другое.

Frontend-часть сайта пишется с использованием HTML, CSS и языка JavaScript. JavaScript позволяет добавить несложную логику в работу frontend-части, например, удалить что-то из списка без обращения к backend-части сайта. Это значительно ускоряет процесс работы сайта и делает его более «живым».

В качестве упражнения давайте попробуем создать простую веб-страничку.

Анатомия веб-страницы

Давайте рассмотрим, из чего же состоит ваша первая веб-страница.

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

```
</head>
<body>
<p>Привет! Это мой первый сайт на курсе от SkillFactory!</p>
</body>
</html>
```

HTML-документ состоит из тегов, которые заключены в < >.

Тег — это базовый элемент языка разметки, основа *HTML*-документа. Закрывающий тег образуется путем добавления слэша / перед именем тега: `<имя тега>…</имя тега>`.

Между начальным и закрывающим тегами находится содержимое тега — **контент**.

В самой первой строке написано `<!DOCTYPE html>` — это информация для браузера о том, что это HTML-документ.

На следующей строчке открывается тег `<html>`. Он является контейнером, который содержит в себе всё содержимое веб-страницы.

Дальше открывается тег `<head>`. В `<head>` мы выделяем ту информацию, которая будет потом доступна при поиске сайта: заголовок (My HTML), информацию о том, какая кодировка используется на этой странице. Обычно в этом же теге подключаются CSS-стили, но у нас совсем простая страничка, поэтому их здесь нет.

В `<body>` добавляются элементы, которые пользователь будет видеть на сайте, например: меню, текст, картинки и так далее. В нашем случае это тег `<p>`, который используется, чтобы хранить в нём текст.

Давайте попробуем добавить ещё текста на наш сайт, а также картинку и немного стилей. Для этого в код нужно добавить следующее:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>My HTML</title>
  </head>
  <body>
    <p>Привет! Это мой первый сайт на курсе от SkillFactory!</p>
    <h1>А вот тут будет кот!</h1>
    <img src="https://images.unsplash.com/photo-1583083527882-
4bee9aba2eea?ixlib=rb-
1.2.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&
fit=crop&w=477&q=80">
  </body>
</html>
```

После этого нужно сохранить изменения. Для этого можно нажать в редакторе CTRL+S. И снова открыть страницу в браузере.

Тег <img> добавил на сайт картинку, а тег <h1> добавил заголовок.

Давайте добавим немного стилей этому сайту. Мы уже упоминали, что за внешний вид сайта отвечает **CSS**. Изменим цвет фона и заголовка. Внесите изменения из примера ниже в свой код, сохраните их и обновите страницу в браузере.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>My HTML</title>
  </head>
  <body style="background-color: cadetblue;">
    <p>Привет! Это мой первый сайт на курсе от SkillFactory!</p>
    <h1 style="color: DarkSlateGray;">А вот тут будет кот!</h1>
    <img src="https://images.unsplash.com/photo-1583083527882-
4bee9aba2eea?ixlib=rb-
1.2.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&
fit=crop&w=477&q=80">
  </body>
</html>
```

Теперь немного оживим страницу и поместим на неё кнопку, которая добавит нашему сайту интерактивности. Для оживления сайта используется **JavaScript.**

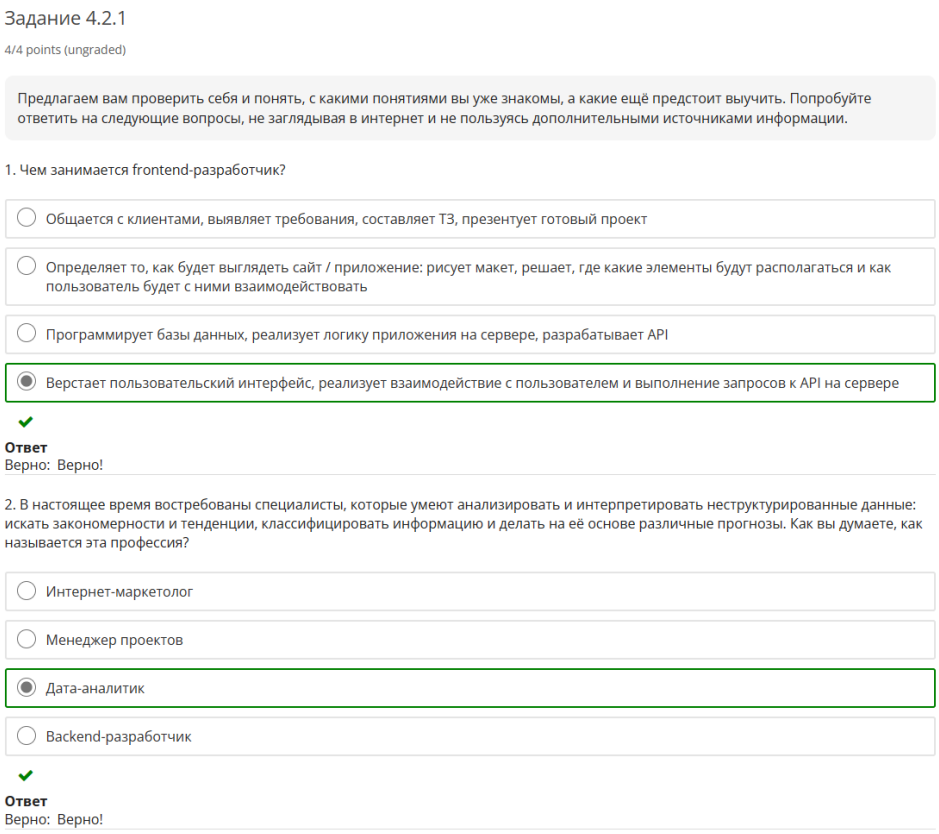
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>My HTML</title>
     <script>
          function changeBackground() {
               let color = document.body.style.backgroundColor;
               if (color === "palegreen") {
                    color = "coral";
               } else {
                    color = "palegreen";
               }
               document.body.style.backgroundColor = color;
          }
     </script>
  </head>
  <body style="background-color: cadetblue;">
    <p>Привет! Это мой первый сайт на курсе от SkillFactory!</p>
     <button onclick="changeBackground()"
          style="width: 200px; height: 50px; background-color:
deepskyblue; font-size: 20px;">
          Нажми меня!
     </button>
    <h1 style="color: DarkSlateGray;">А вот тут будет кот!</h1>
    <img src="https://images.unsplash.com/photo-1583083527882-
4bee9aba2eea?ixlib=rb-
1.2.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&
fit=crop&w=477&q=80">
  </body>
</html>
```

После того как добавили код, обновите веб-страницу в браузере и попробуйте нажать кнопку. Давайте разберёмся, что произошло.

В head мы добавили тег `script`, внутри которого написана функция на языке JavaScript. В этой функции меняется цвет фона. Для того чтобы получить или задать текущий цвет фона, используется `document.body.style.backgroundColor`.

HTML-документ состоит из набора тегов, которые образуют вложенную структуру. Для доступа к элементам этой структуры из JavaScript используется **DOM** (Document Object Model), которая, по сути, является представлением HTML-тегов в виде JavaScript-объектов.

В теге body была добавлена кнопка с помощью тега `button`. Свойство `onclick` задаёт действие, которое будет выполнено при нажатии кнопки.



Задание 4.2.1

4/4 points (ungraded)

Предлагаем вам проверить себя и понять, с какими понятиями вы уже знакомы, а какие ещё предстоит выучить. Попробуйте ответить на следующие вопросы, не заглядывая в интернет и не пользуясь дополнительными источниками информации.

1. Чем занимается frontend-разработчик?

- ○ Общается с клиентами, выявляет требования, составляет ТЗ, презентует готовый проект
- ○ Определяет то, как будет выглядеть сайт / приложение: рисует макет, решает, где какие элементы будут располагаться и как пользователь будет с ними взаимодействовать
- ○ Программирует базы данных, реализует логику приложения на сервере, разрабатывает API
- ◉ Верстает пользовательский интерфейс, реализует взаимодействие с пользователем и выполнение запросов к API на сервере

✔

Ответ

Верно: Верно!

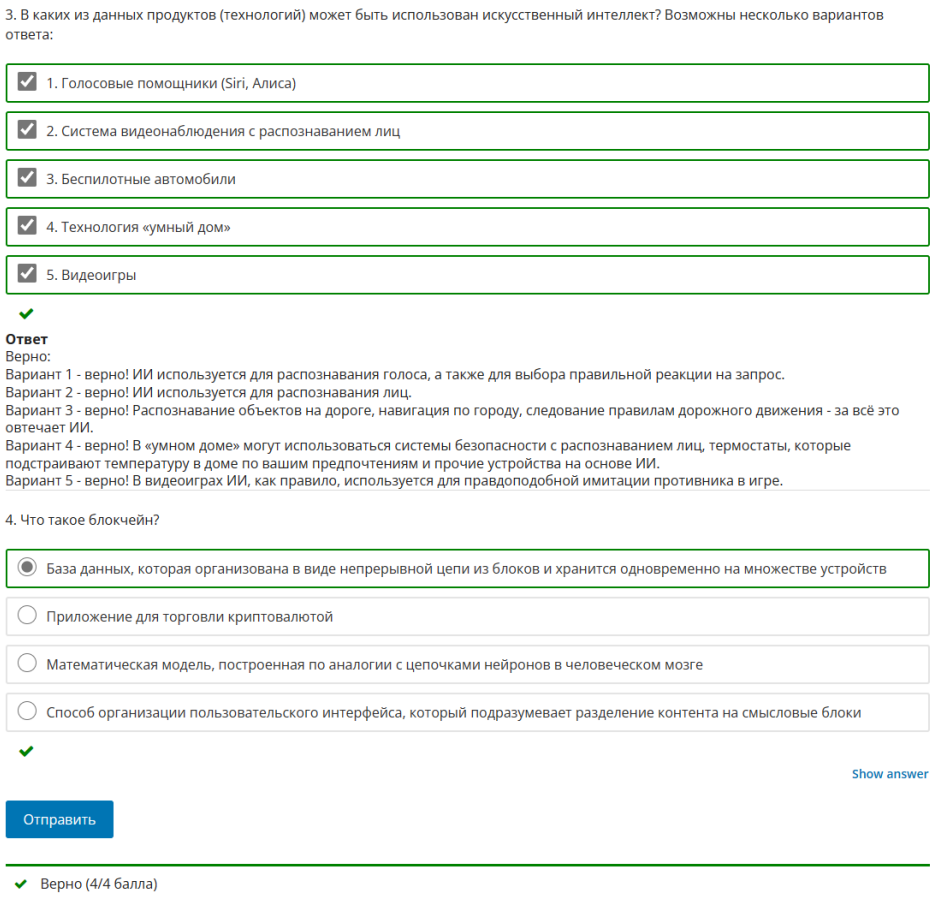
2. В настоящее время востребованы специалисты, которые умеют анализировать и интерпретировать неструктурированные данные: искать закономерности и тенденции, классифицировать информацию и делать на её основе различные прогнозы. Как вы думаете, как называется эта профессия?

- ○ Интернет-маркетолог
- ○ Менеджер проектов
- ◉ Дата-аналитик
- ○ Backend-разработчик

✔

Ответ

Верно: Верно!



Вашей основной задачей на эту неделю будет составление подробной ментальной карты (mind map) ключевых понятий в IT. Давайте для начала разберёмся, что это за карты и зачем их нужно составлять.

Mind map — это способ организации информации в виде ответвлений, которые раскрывают суть идеи или вопроса, который вы хотите проанализировать. Внешне ментальная карта напоминает дерево. С помощью ментальной карты можно описать практически что угодно: идею, процесс, понятие, сферу, сюжет книги или фильма, структуру сайта или приложения и даже человека.

Пример ментальной карты для понятия «Маркетинг»:

А вот, например, ментальная карта со сферами применения ментальных карт:

Ментальные карты были придуманы в 70-х годах Тони Бьюзеном и используются для самых различных целей, например:

- планирование;
- конспектирование;
- запоминание информации;
- осмысление сложных идей.

Основная идея карты заключается в том, что наш мозг лучше усваивает информацию, которая **визуализирована** и **связана**.

Простой пример: попробуйте вспомнить точный адрес своего друга, родственника или магазина, в котором вы закупаетесь каждую неделю. В большинстве случаев это будет сложно сделать, при этом вы отлично помните, как дойти до нужного места от остановки общественного транспорта, как оно выглядит, что находится вокруг. Это огромное количество информации окажется вспомнить легче, чем адрес — обычную строчку текста. Так происходит потому, что когда задействуется пространственно-образное мышление, память активизируется и работает эффективнее.

Правила составления mind map

Существуют некоторые правила по составлению mind map (вам необязательно следовать им на 100%, но они могут помочь сделать карту более полезной и понятной).

1. **В центре карты** помещается основная идея (объект, понятие), которое вы будете разбирать.
2. От главного понятия идут ответвления — **подтемы**. Каждая подтема также может иметь любое количество ответвлений. Количество уровней карты не ограничено и зависит от сложности темы и подробности карты. Ответвления могут представлять собой не только подтемы, но и различные ассоциации с понятием, его характеристики, подзадачи, и вообще любые сущности, связанные с ним.
3. Блоки карты **могут сопровождаться рисунками** для наглядности.
4. Чтобы визуально отделить различные смысловые части карты, можно использовать выделение **разными цветами**.
5. Не рекомендуется размещать на карте большие блоки текста, они делают её избыточной и сложной для понимания.

Ещё несколько примеров более сложных ментальных карт:

- Языки программирования

- Объяснение языка разметки HTML
- Правила SEO (поисковой оптимизации)
- План бизнеса по продаже готовых сайтов

Карты можно составлять любым удобным способом: как вручную (на листе бумаги), так и на компьютере, с помощью специальных приложений. Раз уж мы занимаемся темой IT, воспользуемся вторым вариантом.

Приложение для составления mind map

Самое популярное приложение для составления mind map — **Xmind.** Оно доступно для всех платформ, скачать можно по ссылке с официального сайта.

В программе доступен весь необходимый функционал для создания даже сложной и развернутой карты: можно группировать и связывать блоки, выделять ветви разными цветами, добавлять картинки и файлы. Получившуюся карту можно экспортировать в разных форматах: PNG, SVG, PDF, документ Word и других. Для наших целей будет вполне достаточно бесплатной версии.

Пример mind map с планом рождественской вечеринки, созданный в Xmind.

В Xmind довольно простой и интуитивно понятный интерфейс, при первом запуске вам предложат пройти обучение. В качестве тренировки попробуйте составить карту ваших планов на следующие выходные. Например, такую:

С помощью ментальной карты можно отображать различные сущности, в том числе веб-сайты и приложения. Пример карты для Интернет-магазина шин:

✍ **Дополнительное задание**

Выберите сайт (или приложение), который вам нравится, и попробуйте описать его с помощью ментальной карты. Каждый раздел будет представлять собой отдельное ответвление. С помощью ответвлений от разделов можно также показать функции, которые этот раздел выполняет.

Критерии по выполнению задания на модуль

В каждом юните вы будете знакомиться с новой сферой в IT. После прохождения юнита вам нужно будет отобразить на карте следующую информацию:

- Какие задачи решаются в этой сфере?
- Какие технологии и инструменты при этом используются?
- Какие специалисты задействованы?
- Несколько примеров практического применения
- Как эта сфера взаимосвязана с другими сферами?

Старайтесь заполнять карту сразу же после прохождения каждого юнита.

Как искать информацию для заполнения карты?

При составлении mind map IT-сферы часть информации вы получите из юнитов, но другую часть информации вам будет необходимо найти и разобрать самостоятельно. Поиск и систематизация информации — важный и необходимый навык для любого IT-специалиста.

Вот несколько советов, как сделать поиск информации более эффективным.

Правильно составляйте поисковые запросы
Избегайте орфографических ошибок, уточняйте запрос, добавляя больше ключевых слов (в разумном количестве). Пользуйтесь функцией расширенного поиска: она позволяет отфильтровать результаты по дате, местоположению, формату файла и другим параметрам.

Используйте операторы поиска
У Google и Яндекс есть специальные логические операторы, встроенные в поисковик, которые помогают более точно задать условия поиска. Например, поисковый запрос в кавычках будет искать только точные совпадения, а минус перед словом в запросе исключит результаты, в которых используется это слово. Полные списки операторов поиска: Яндекс и Google.

Копайте глубже
Информация на первых 10-ти позициях в поиске не всегда является наилучшей. Это могут быть сайты с рекламой или просто популярные сайты, что не гарантирует, что вы найдёте там то, что ищете. Кроме того, лучше использовать оба поисковика: Яндекс и Google, потому что правила ранжирования у них отличаются, и результаты поиска могут быть разными.

Пользуйтесь качественными источниками
Всегда обращайте внимание на источник, в котором вы берёте информацию: кто является автором, насколько этот источник надежный, как давно была опубликована информация. Надежными источниками могут быть, к примеру, проверенные издания, книги, энциклопедии, эксперты в своей области.

Пользуйтесь преимуществами рассылок
Если вам важно постоянно держать руку на пульсе и оперативно узнавать новости по вашей сфере деятельности, подпишитесь на рассылку одного или нескольких авторитетных изданий по теме. Так вы потратите меньше времени на поиск информации о новинках — с рассылкой она найдёт вас сама.

Задание 4.2.2

Составьте для себя список из 5 сайтов IT-направленности, на которых вы могли бы искать информацию и читать новости.

В качестве рекомендаций можем предложить следующие сайты: habr.com/ru, tproger.ru, vc.ru.

Дополнительно
Если вас заинтересовала тема ментальных карт и вы неплохо владеете английским языком, предлагаем вам посмотреть выступление создателя mind map Тони Бьюзана на конференции TED.

- The Power of a Mind to Map: Tony Buzan at TEDxSquareMile (YOUTUBE)

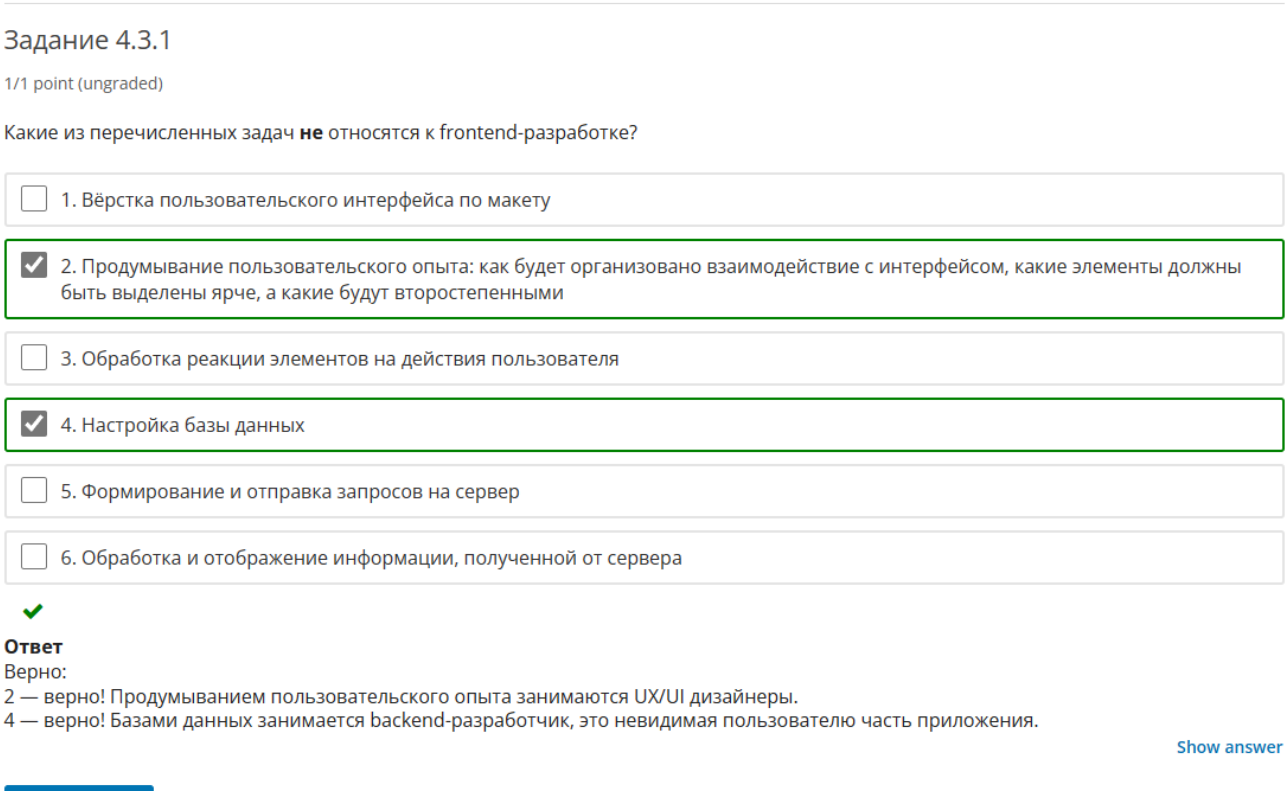
В данном юните мы рассмотрим первое понятие из мира IT — **фронтенд.** Вспомним, какие задачи решаются во фронтенде, какие технологии при этом используются и какие специалисты задействованы в этой сфере.

Понятие «фронтенд» в основном применяют в веб-разработке, то есть в разработке сайтов или веб-приложений. По сути, фронтенд — это всё, с чем непосредственно контактирует и взаимодействует пользователь. К примеру, возьмём приложение Яндекс.Такси. Открыв приложение, вы увидите примерно такую картину, как на изображении слева.

Внешний вид и расположение элементов, реакция элементов интерфейса на нажатие или «свайпы» карты — всё это создано заботливыми руками frontend-разработчиков.

После того, как вы выберете отправную и конечную точки, программа оформит эти данные в виде запроса и отправит на сервер. Кроме того, при открытии приложение также успело сделать запрос на сервер, чтобы узнать необходимую исходную информацию (ваша учетная запись, местоположение, ваши сохраненные точки для маршрутов, способы оплаты и т.д.) и отобразило её на интерфейсе. Эта логика тоже реализуется на стороне фронтенда.

Таким образом, можно сказать, что **frontend-разработка** — это создание публичной части веб-приложения, которая непосредственно доступна пользователю, а также функционала, который выполняется на стороне клиента.



Что входит в область фронтенда?

Раньше, когда персональные компьютеры и тем более телефоны не отличались высокой производительностью, основной задачей frontend-разработчика было сверстать и оформить интерфейс согласно макету и добавить простейшую логику взаимодействия (всплывающее окно, выпадающий список и т.д.). Основная часть логики приложения (или сайта) была перенесена на сервер и относилась к сфере бэкенда.

Сейчас, когда у нас в распоряжении есть мощные браузеры и ещё более мощные компьютеры, всё больше и больше процессов перетекает из бэкенда во фронтенд, поэтому с каждым годом сфера фронтенда всё больше разрастается и включает в себя всё больше задач.

Можно выделить следующий **ряд задач, которые решает фронтенд**:

Генерация и отображение пользовательского интерфейса
Пример: расположение, внешний вид кнопок и подписей.

Обновление интерфейса в ответ на действия пользователя
Пример: открытие меню после нажатия на кнопку «Меню», отображение индикатора загрузки на время, пока ведётся поиск, вывод информации о поездке после заказа.

Формализация и проверка данных, полученных от пользователя, перед отправкой на сервер
Пример: первичная проверка данных (например, что значение в поле «Телефон» соответствует номеру телефона, текстовые поля не содержат вредоносный код),

перевод данных в специальный формат для отправки на сервер (JSON или XML).

Выполнение запросов на сервер
Пример: отправка запроса с местоположением, чтобы узнать количество свободных машин в этом районе.

Важно отметить, что **к сфере фронтенда не относится веб-дизайн и UX/UI** (*User Experience/User Interface*). UX/UI-дизайнер проектирует пользовательский интерфейс так, чтобы он цеплял пользователя и при этом был удобным (было легко найти нужную информацию, прочитать текст, нажать на кнопку), и разрабатывает прототип интерфейса. На основе прототипа веб-дизайнер рисует макет, добавляет цвета, изображения, различные дизайнерские решения и отдаёт макет разработчику.

Отличия прототипа (слева) и макета (справа)

 Источник: krasnodar.stk-promo.ru

Frontend-разработчик реализует уже готовый макет, ему не нужно задумываться над оформлением, главная задача — как можно более точно реализовать интерфейс, задуманный дизайнерами.

HTML, CSS и JS

Frontend-разработка держится на трёх китах: HTML, CSS и JavaScript. Разберём вкратце, для чего они предназначены.

HTML
Язык гипертекстовой разметки, который используют для построения структуры web-страницы: заголовков, абзацев, списков и прочего. Определяет состав и количество элементов в интерфейсе, их порядок следования и содержимое.

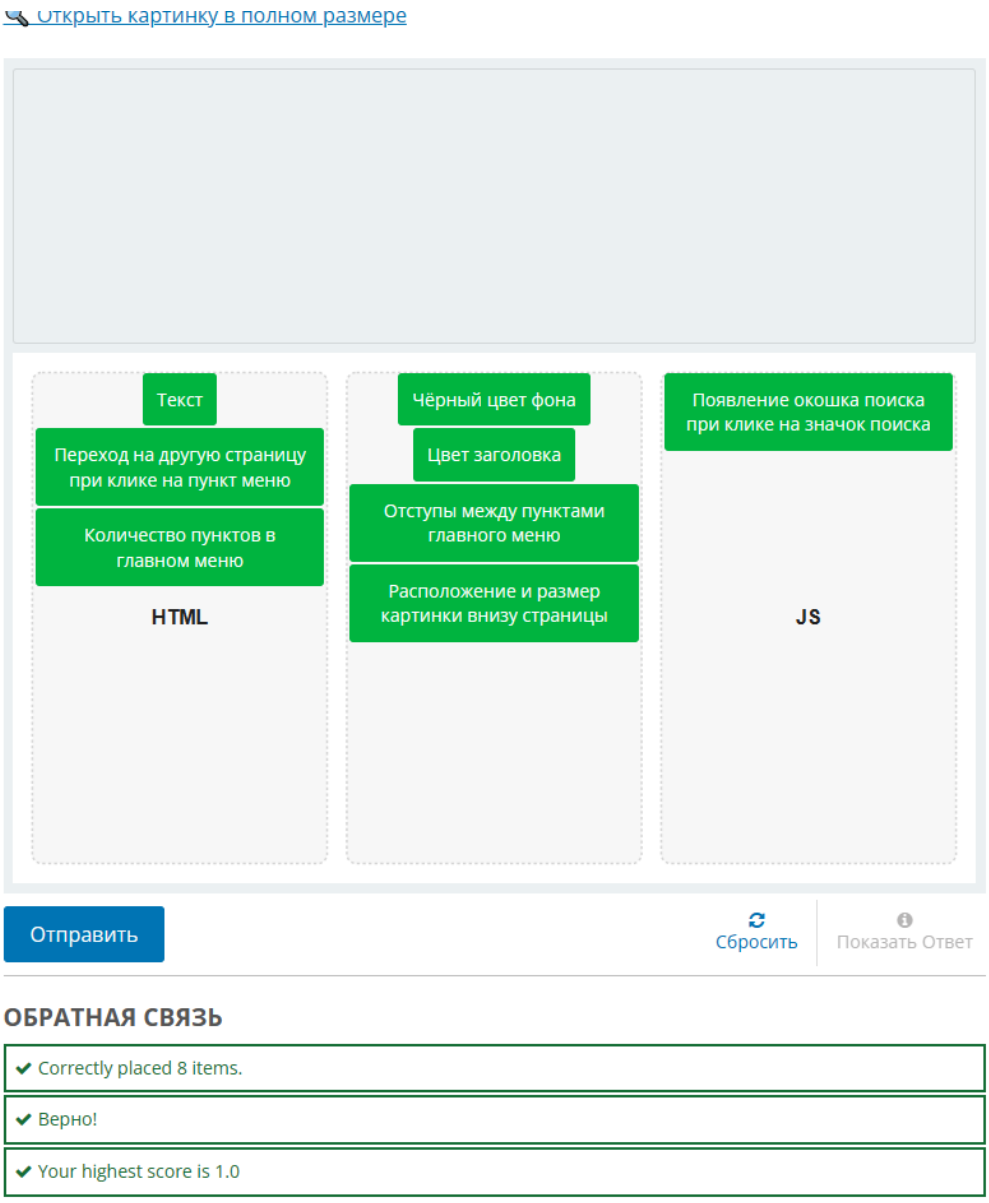
CSS

Язык, используемый для описания внешнего вида страниц. Именно этот код отвечает за распознавание браузером отдельных элементов на странице. CSS определяет внешний вид каждого элемента на странице: цвет, размер, форма, отступы, оформление текста и т.д.

JavaScript

Мультипарадигменный язык программирования, разработанный для использования в браузерах. JavaScript добавляет интерактивности и буквально оживляет страницы сайтов.

Хорошая иллюстрация того, за что отвечает каждая технология:



Как правило, frontend-разработчик работает со всеми тремя этими технологиями (HTML, CSS, JS) и должен освоить их в совершенстве.

Кроме того, в современном фронтенде существует куча «надстроек» над классическими HTML, CSS и JS: препроцессоры, сборщики, JS-библиотеки и фреймворки. Во всём этом frontend-разработчик должен хорошо разбираться.

Существует также профессия **верстальщик** — это человек, который занимается только HTML и CSS. Его задача — правильно и точно сверстать макет, а взаимодействием с пользователями занимаются другие разработчики.

Задание на mind map

Итак, мы вкратце разобрались с тем, что такое фронтенд. Пришло время нанести его на mind map IT-сферы.

Шаг 1

Отобразите на карте следующую информацию:

- задачи, которые решаются в фронтенде;
- три технологии, которые лежат в его основе;
- как называются специалисты, которые работают в сфере;
- три конкретных примера, за что отвечает фронтенд.

Найдите 3-5 примеров конкретных продуктов (библиотек, препроцессоров и фреймворков), которые в данный момент наиболее востребованы и часто используются, и также нанесите их на карту.

В этом юните мы подробнее изучим следующее понятие — бэкенд-разработка. Мы продолжим разбирать пример с приложением Яндекс.Такси, определим круг задач, который решается на стороне бэкенда и стек используемых технологий.

Вы уже знаете, что **backend-разработка** отвечает за логику работы приложения и хранение данных. В отличие от фронтенда, эта часть приложения скрыта от глаз пользователя.

Представьте, что вы заходите в поисковик, вбиваете в строку запрос и нажимаете кнопку «Поиск». Ваш запрос отправился на сервер, и с этого момента в процесс вступает бэкенд. Программа на сервере обработает ваш запрос, обратится в базу данных за нужными результатами поиска и, получив их, отправит информацию обратно на фронтенд.

В этом заключается **главная суть бэкенда**: получить запрос от фронтенда, обработать, собрать нужную информацию (в базе данных или у другого сервиса) и отправить ответ обратно.

Продолжим разбирать пример с Яндекс.Такси из предыдущего юнита. Что же происходит после того, как пользовательский запрос (который содержит отправную точку и точку назначения) приходит на сервер?

Схематически процесс можно отобразить так:

↑ Настоящий алгоритм работы Яндекс.Такси в разы сложнее. В целях обучения мы привели сильно упрощенную и сокращенную версию.

Задание 4.4.1
12/12 points (ungraded)

Попробуем разобраться в процессе взаимодействия фронтенда и бэкенда на примере другого приложения — почтового клиента. Представим, что пользователь зашёл в почтовое приложение, выбрал адрес из записной книжки, написал и отправил письмо. Определите последовательность действий, которая будет соответствовать этому сценарию.

Все шаги в перемешанном порядке:

A. Бэкенд-часть обратилась к базе данных, получила информацию о последних письмах в ящике пользователя и его список контактов и отправила этот пакет данных на фронтенд.

B. Бэкенд отправил в базу данных запрос о внесении информации о письме в таблицу исходящих писем

C. Бэкенд сделал запрос на специальный сервис, чтобы убедиться, что почтовый адрес получателя существует.

D. Пользователь написал письмо и нажал кнопку «Отправить».

E. Бэкенд проверил, не содержат ли полученные данные вредоносный код.

F. Бэкенд отправил на фронтенд информацию об успешной отправке письма. Фронтенд выводит всплывающее сообщение «Письмо успешно отправлено».

G. Пользователь зашёл в приложение.

H. Фронтенд отобразил индикатор загрузки с текстом «Подождите, письмо отправляется».

I. Фронтенд получил ответ от бэкенда и вывел на интерфейс последние письма и книгу контактов.

J. Фронтенд-часть отправила запрос на бэкенд с данными об идентификации пользователя (например, его id или логин в системе).

K. Бэкенд сформировал пакет данных и отправил на почтовый сервис, дождался ответа, что письмо успешно отправлено.

L. Фронтенд отправил запрос на сервер с содержанием письма и адресом получателя.

Правильная последовательность:

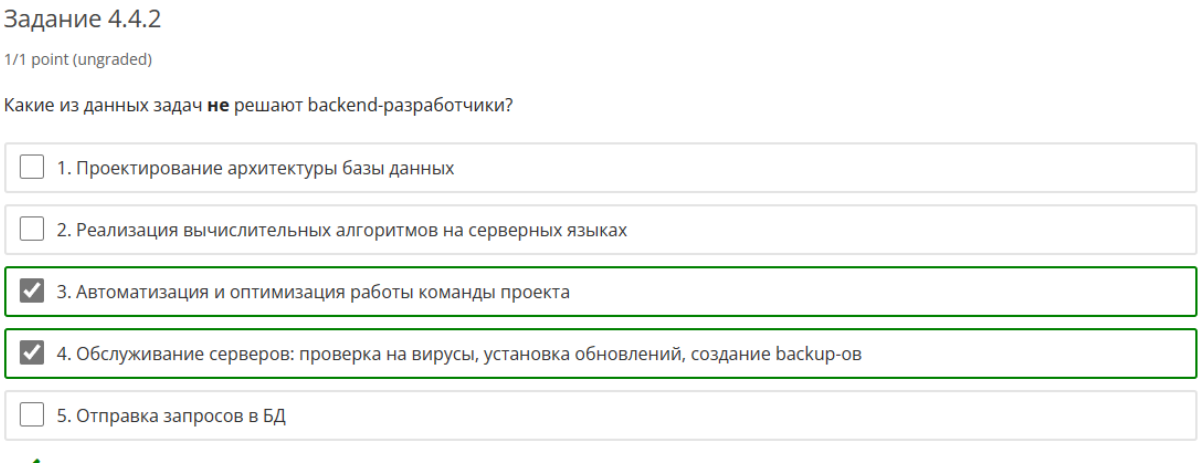
G ✔
J ✔
A ✔
I ✔
D ✔
H ✔
L ✔
E ✔
C ✔
K ✔
B ✔
F ✔

Задачи бэкенд-разработки

Пусть бэкенд и не виден конечному пользователю, его качество имеет значение ничуть не меньшее, а иногда даже и большее, чем качество фронтенда. Бэкенд отвечает за множество аспектов, критически важных для работы программного продукта: скорость загрузки, быстрота и правильность вычислений, цельность и сохранность данных в базе данных (БД) и многое другое.

Итак, в чём же заключаются главные задачи бэкенд-разработки?

6. Обработка запросов от пользователя и отправка ответа.
7. Реализация вычислительной логики и алгоритмов работы приложения.
8. Организация работы баз данных и написание запросов к ним.
9. Разработка API (Application Programming Interface — программный интерфейс приложения).
10. Интеграции с внешними сервисами. В случае, если информация, которую хочет получить пользователь, находится не в нашей базе данных, а в базе данных другого сервиса, программа должна будет скорректировать запрос (ведь у каждого сервиса свои требования ко входным данным) и перенаправить его другому сервису.



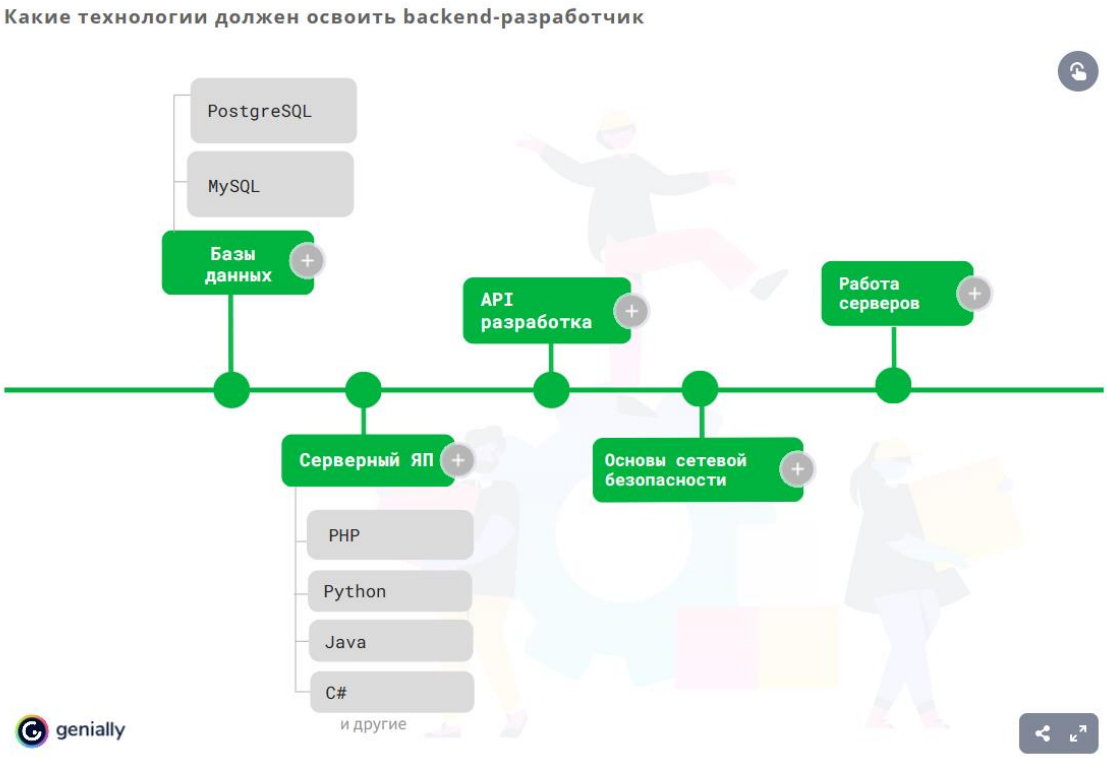
Умение работать с базами данных: проектировать БД и писать запросы. Для написания запросов к БД существует особый язык — SQL.

В последнее время стало возможным писать бэкенд и на JavaScript, который знаком нам из юнита про фронтенд, используя программную платформу Node.js.

Знание принципов разработки API (Application Programming Interface — программный интерфейс приложения).

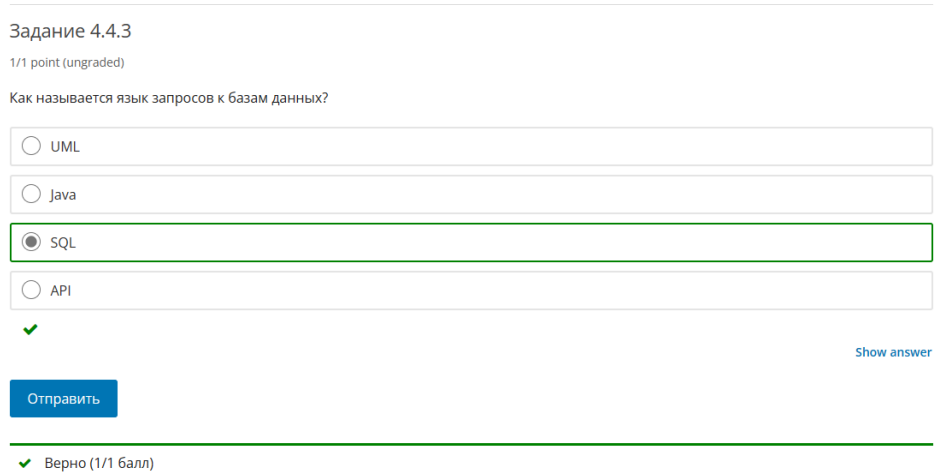
Необходимо знать, какие угрозы безопасности бывают и как можно от них защититься.

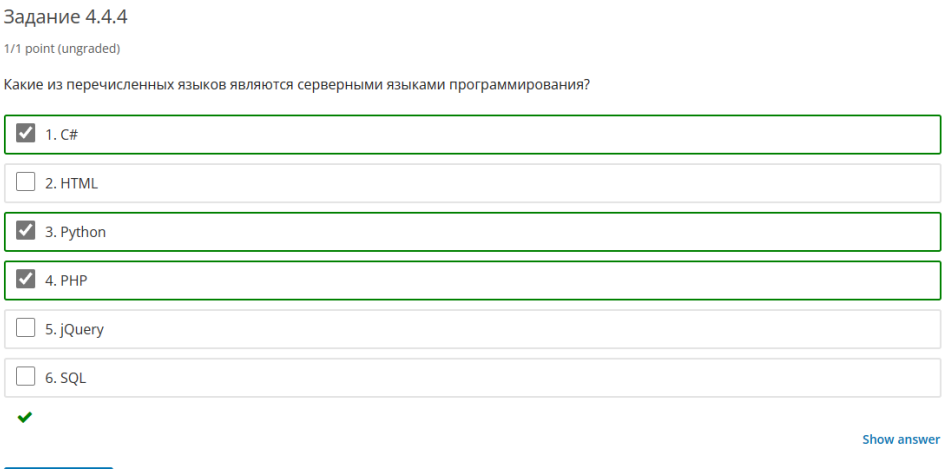
Два самых распространенных веб-сервера — Apache и Nginx. В основном в рамках бэкенда решаются самые тривиальные задачи с серверами: перезапустить, изменить какие-то настройки, ограничить или расширить доступ.



В более мелких компаниях все эти функции могут быть сосредоточены в рамках одной должности — **backend-разработчик**. В более крупных компаниях или при разработке сложного продукта могут участвовать такие специалисты, как:

- **SQL-разработчик**
  Узкоспециализированный разработчик, который занимается написанием запросов к БД.
- **Архитектор БД**
  Занимается проектированием и созданием баз данных. В случае, когда в базе содержатся сотни тысяч записей с разветвленной структурой и большим количество связей, правильно спроектированная структура БД — важнейшее условие нормальной работы программы.





Frontend vs Backend

Скорее всего, вы уже задавались вопросом, что лучше: frontend-разработка или backend-разработка? Однозначного ответа на этот вопрос не существует. У каждой из этих профессий есть свои особенности.

Backend	Frontend
Больший выбор языков программирования: Python, Java, Ruby, C#, PHP и многие другие.	Один и тот же набор основных технологий: HTML, CSS и JS. Однако помимо классических HTML+CSS+JS нужно знать различные надстройки: препроцессоры и фреймворки.
Более стабильная сфера, обновления языков выходят редко.	Очень динамичная сфера, постоянно появляются новые тренды, инструменты, библиотеки. Часто приходится переписывать и модернизировать код.
Более высокий порог вступления в профессию: помимо навыков программирования, нужно обладать знаниями об устройстве баз данных, архитектуре приложений, знаниями математики.	Начать работать по профессии проще, стек HTML+CSS+Javascript проще изучать новичкам.
Процесс работы более однообразный, нет наглядного результата работы.	Результат работы виден сразу (пользовательский интерфейс).

Если вам интересны обе эти сферы и сложно выбрать что-то одно, есть вариант профессии, которая объединяет фронтенд и бэкенд — это **fullstack-разработчик** (фулстэк-разработчик). Такой специалист является в каком-то смысле «мастером на все руки» и хорошо разбирается в обеих сферах. Конечно,

фулстэк-разработчиком стать гораздо сложнее, ведь нужно усвоить в 2 раза больше навыков. Но зато такие разноплановые специалисты высоко ценятся на рынке труда.

Давайте добавим на mind map сферу бэкенда.

Шаг 1

Отобразите на карте следующую информацию:

- задачи, которые решает бэкенд;
- 5 частей, из которых состоит бэкенд: серверные языки программирования, базы данных, устройство серверов, API, информационная безопасность;
- специалисты, которые могут работать в сфере бэкенда;
- 5 серверных языков программирования;
- отобразите на карте, как связаны сферы фронтенда и бэкенда.