draftpodium@gmail.com

1. Опишите своими словами, как вы понимаете фронтенд и бекенд. Приведите по 1 примеру вида тестирования, применимого к фронтенду и бекенду, соответственно.

**Фронтенд** — все, что браузер может читать, выводить на экран и / или запускать. То есть это HTML, CSS и JavaScript.

HTML (**H**yper**T**ext **M**arkup **L**anguage) говорит браузеру, каково содержание страницы, например, «заголовок», «параграф», «список», «элемент списка».

**Бэкенд** — все, что работает на сервере, то есть «не в браузере» или «на компьютере, подсоединенном к сети (обычно к Интернету), который отвечает на сообщения от других компьютеров».

Для бэкенда вы можете использовать любые инструменты, доступные на вашем сервере (который, по сути, является просто компьютером, настроенным для ответов на сообщения). Это означает, что вы можете использовать любой универсальный язык программирования: Ruby, PHP, Python, Java, JavaScript / Node, bash. Это также означает, что вы можете использовать системы управления базами данных, такие как MySQL, PostgreSQL, MongoDB, Cassandra, Redis, Memcached.

Что такое кросс-браузерное тестирование?

1. **Кроссбраузерное тестирование** (Cross-browser testing) **Тестирование** кроссбраузерности — вид **тестирования**, направленный на поддержку и правильное полное отображение программного продукта в разных браузерах, мобильных устройствах, планшетах, экранах различного размера
2. Предположим вы нашли баг на веб-странице. Какие данные помимо шагов воспроизведения, вы можете приложить к баг-репорту, чтобы разработчику было проще локализовать проблему?
3. Опишите своими словами, как вы понимаете GET и POST запросы. В чем их основное различие?

Методы GET и POST используются для [отправки данных HTML формы на сервер.](https://space-base.ru/library/?book=41)

В целом оба метода выполняют аналогичную функцию – передают на сервер введенные в форме данные. Отличия определяются применением каждого из методов.

Например, для формы:

<form action="myform.php" method="post">  
<input type="text" name="data1">  
<input type="text" name="data2">  
<input type="submit" value="Отправить">  
</form>

Если, в поля два текстовых поля формы ввести значения 15 и 20, то при выполнении GET-запроса, в адресной строке браузера будет явно виден url (адрес) страницы. Для такой формы это будет:

http://my\_site.ru/myform.php?data1=15&data2=20

При выполнении передаче методом POST мы увидим лишь:

http://my\_site.ru/myform.php

Никаких сведений о самих передаваемых данных здесь не увидеть.

Если, например, создается форма авторизации, то удобнее будет использовать POST запрос, т.к. в сохраненной браузером строке можно будет явно увидеть и логин и пароль.

GET же стоит использовать тогда, когда его результат можно полезно использовать для получения необходимой страницы повторно. Например, с параметрами необходимой сортировки или выборкой. Строку GET запроса можно увидеть в любом интернет поисковике.

Сохранив url, полученный методом GET, из адресной строки, можно всегда получить ту же страницу с уже подставленными данными, не заполняя форму отправки данных заново.

\* Это не всегда срабатывает с поисковиками. Через некоторое время результат для сохраненной страницы будет меняться.

Если же форма будет отправляться методом POST, адрес полученной страницы будет всегда один, какие бы данные не вводились.

Объем передаваемой информации у этих методов тоже различен. С помощью GET лучше отправить небольшие тестовые данные. Максимальный объем здесь 4 Кб.

Для POST такого явного ограничения нет. Максимальный размер для него задается настройками сервера. Поэтому он подходит для [загрузки файлов на сервер](https://space-base.ru/library/?book=18) и передачи больших объемов текста.

5. Опишите своими словами, как вы понимаете API-тестирование.

API предоставляет нам возможность использовать чужие наработки в своих целях. Впервые я столкнулся с API на примере Windows API. Это набор функций, которые может использовать любое приложение, запущенное на данной ОС. К примеру, оно может использовать стандартные функции для отрисовки интерфейса.  
  
Современные API часто принимают форму веб-сервисов, которые предоставляют пользователям (как людям, так и другим веб-сервисам) какую-то информацию. Обычно процедура обмена информацией и формат передачи данных структурированы, чтобы обе стороны знали, как взаимодействовать между собой.