



Speaker of the report Didenko Inna

**QA Padawans**

**Начнем!**

Вы  
Готовы?

## Что это?



**Тестирование веб-приложений** – это комплекс услуг, который может включать в себя различные виды тестирования ПО.

Основная цель любого тестирования, в том числе и тестирования веб-приложений, – обнаружить все ошибки в программном обеспечении и разработать рекомендации по их предотвращению в будущем.

### **Три основных вопроса, которые можно решить с помощью тестирования веб-приложений**

1. Отвечает ли функциональность приложения заявленным требованиям?
2. Будет ли приложение правильно работать на всех устройствах, популярных у конечных пользователей?
3. Выдержит ли приложение большое количество пользователей?

## **Начинаем тестировать не с тестирования: с чего начать?**

Я хочу акцентировать ваше внимание на необходимости согласовать все ключевые аспекты с ответственными лицами (аналитиками, проектными менеджерами, разработчиками) еще до начала тестирования. Необходимо заранее выяснить следующие моменты:

- цель тестирования (с их точки зрения);
- виды тестирования, которые необходимо провести;
- специфику приложения и его целевую аудиторию;
- перечень устройств, на которых необходимо проводить тестирование;
- список ОС и браузеров для тестирования;
- разрешения экранов на которых необходимо проверить приложение;
- предусмотрены ли требования к разного рода формам;
- необходимость предоставления конкретной документации по результатам тестирования (отчет, чек-листы, тест-кейсы и т. д.).

Любое тестирование требует содержательного подхода с применением техник тест-анализа и тест-дизайна. Без стойкого понимания методик и способов применения тест-анализа и тест-дизайна тестировать качественно ПО практически невозможно.

Рассмотрим классический набор методик **тест-дизайна**, которые можно применять при тестировании веб-приложений:

- разбиение на классы эквивалентности;
- попарное тестирование;
- тестирование переходов состояний;
- анализ граничных значений;
- предугадывание ошибок;
- тестирование с помощью таблиц решений и множество других.

# Кроссбраузерность

Первое — это проверка на правильность отображения и функционирования web-приложения на различных браузерах. Под правильностью понимается соответствие стандартам и требованиям. Важно: перед началом тестирования надо выяснить, какие конкретно браузеры поддерживаются.

Что проверяем при кроссбраузерном тестировании:

- функциональные возможности продукта, которые реализуются на стороне клиента;
- правильно ли отображаются элементы графики;
- корректно ли отображаются шрифты, размеры текстовых символов; доступны ли формы, присутствующие на сайте, выполняют ли они свои функции, являются ли интерактивными.

# Функциональное

**Функциональное тестирование** – процесс оценки поведения приложения, позволяющий определить, все ли разработанные функции ведут себя так, как нужно. Для корректной работы продукта все процессы должны работать так, как это предусмотрено в требованиях: от разграничения прав доступа при авторизации до корректного выхода из системы.

Функциональное тестирование может быть выполнено с использованием заранее подготовленных тестовых сценариев или методами исследовательского тестирования.

это  
как?

это  
куда?

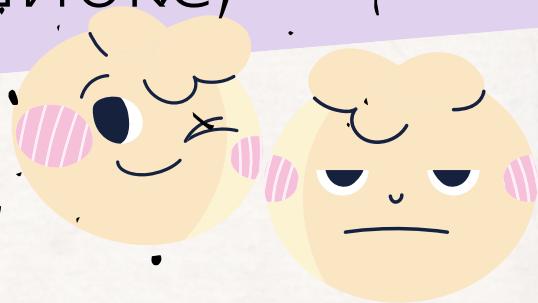
## 1. Web-формы

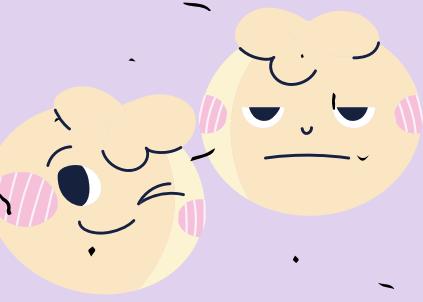
**Формы для заполнения** — важнейшие составляющие веб-приложений.

Именно с помощью форм осуществляется взаимодействие клиента с сервером (клиент — это, к примеру, веб-браузер, через который пользователь обращается к серверу).

На что стоит обращать особое внимание:

- обязательные поля для заполнения: отмечено ли, что они являются обязательными, а также являются ли они обязательными на самом деле, по факту (проверка пустых значений);
- ввод специальных данных;
- спецсимволов: #%@№&^\$\*<>{}[];
- наличие валидации (что происходит, когда пользователь вводит невалидные значения, получает ли пользователь сообщение об ошибке, что происходит с данными после их ввода).





## Валидация со стороны сервера

Перед вами стандартная форма входа на сайт с полями для логина и email. Введенные вами значения будут отправлены на сервер и проверены на наличие в базе данных (автентификация). Если вы уже зарегистрированы и ввели правильные данные, то получите доступ и права (авторизация), если нет — увидите сообщение о том, что указанный вами email/логин не существует.

## Валидация со стороны клиента

Проверка значений непосредственно при вводе данных. Для этого подключают специальные скрипты валидации (то, что мы видим, UI).

Что проверяют:

- ввел ли пользователь в поле email символ @;
- заполнены ли обязательные поля; — ввел ли пользователь цифры в поле ввода номера телефона;
- не совпадают ли поля, которые не должны совпадать, и т. п.

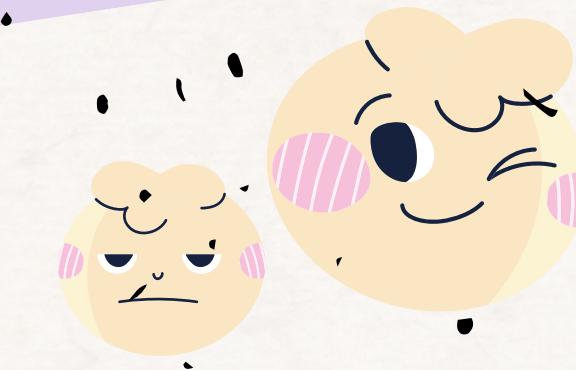
Для чего это нужно? Ответ очевиден: валидация на стороне клиента уменьшает число обращений к серверу, снижая нагрузку на него.

## **2.Поиск**

- Результаты существуют/не существуют
- Корректное сообщение о пустом результате
- Пустой поисковой запрос
- Поиск по эмодзи

## **3.Всплывающие сообщения**

- Протестируйте всплывающие сообщения («Это поле ограничено N знаками»).
- Подтверждающие сообщения отображаются для операций обновления и удаления.
- Сообщения об ошибках ввода.



## **4.Фильтры**

- Протестируйте функциональность сортировки (по возрастанию, по убыванию, по новизне)
- Задать фильтры с выдачей
- Задать фильтры, по которым нет выдачи
- Фильтры по категориям/подкатегориям
- Фильтры с радиусом поиска

## **5.Протестируйте функциональность доступных кнопок**

## **6.Проверка обработки различных ошибок** (страница не найдена, тайм-аут, ошибка сервера и т.д.)

## **7.Протестируйте, что все загруженные документы правильно открываются**

## **8.Пользователь может скачать/прикрепить/загрузить файлы/медиа** (картинки, видео). А также удалить эти файлы из вложений

## **9.Протестируйте почтовую функциональность системы**

## **10. Кеш, cookie и сессии**

- Пользователь очистил кэш браузера
- Посмотрите, что будет, если пользователь удалит куки, находясь на сайте.
- Посмотрите, что будет, если пользователь удалит куки после посещения сайта.

## **11. DevTools**

- Ошибки в Console.
- Все стили загружаются.
- Картинки загружаются

# Локализация

Это  
только  
язык?

Локализация в разработке программного обеспечения означает адаптацию программы к какой-либо стране, культуре или языку. Слово local в переводе с английского значит «местный». Проще говоря, локализация это перевод интерфейса программы на местный язык.

Локализация не всегда ограничивается тупым переводом. Часто нужно поменять в программе гораздо больше, чем языковые файлы.



что  
смотрим?

- Дата и время. Например отображение времени, даты в соответствии с часовым поясом пользователя.
- Смена языка и проверка перевода всех элементов WEB приложения исходя из выбранного языка.
- Выбор номера телефона с разными кодами стран.
- Определение местоположения пользователя и отображение соответствующего пермишена ГЕО.
- Отображение соответствующих символов валюты.
- Цветовые схемы, символы, значки и другие графические элементы, которые могут быть по-разному истолкованы в различных регионах;
- Правовые нормы, которые следует учитывать.



# Интеграционное

**Интеграционное тестирование** – это тип тестирования, при котором программные модули объединяются логически и тестируются как группа. Как правило, программный продукт состоит из нескольких программных модулей, написанных разными программистами. Целью нашего тестирования является выявление багов при взаимодействии между этими программными модулями и в первую очередь направлен на проверку обмена данными между этими самими модулями.

- Проверяем работу сторонних модулей: оплата картой.
- Реклама (просмотр, переходы по рекламе).
- Метрики (переходы по страницам, показы элементов, клики).

Это  
модули?

Или  
что-то  
другое?



# Графический интерфейс

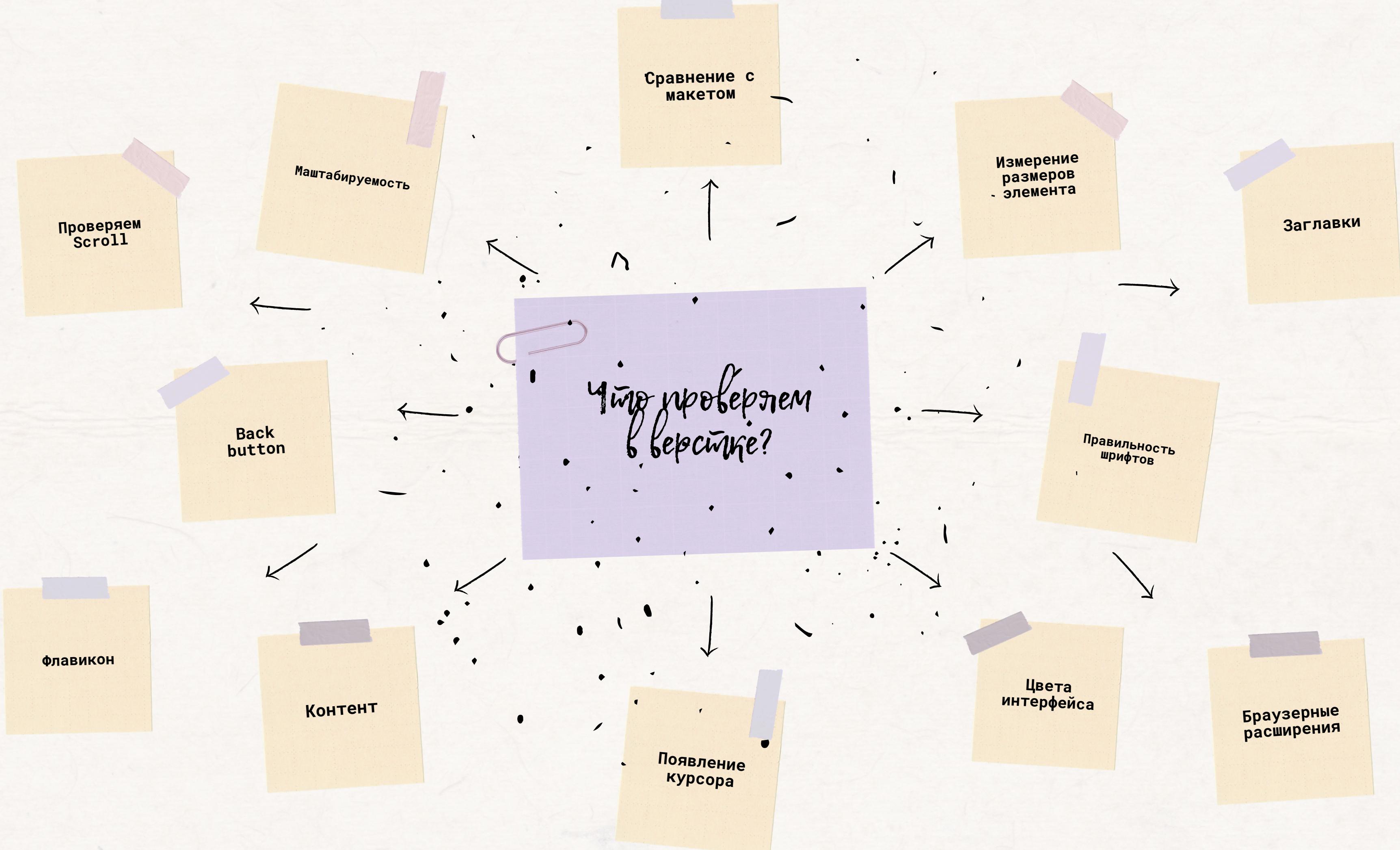
**GUI** — у любого тестируемого предмета и веб-приложения есть внешний вид, поэтому тестирование графического интерфейса или попросту, внешнего вида — это самое первое, что мы можем сделать.

**Верстка** — размещение элементов веб-приложения (изображения, текст, кнопки, видео...) в соответствии с макетом или требованиями.

Проверяем:

- наличие всех элементов;
- их размер и цвет;
- расположение относительно друг-друга.

GUI



# Удобство

удобно  
или нет?

Usability

О каких моментах нужно помнить при тестировании **usability** веб-приложения?

- Соответствует ли приложение ожиданиям конечного пользователя;
- Логичность интерфейса;
- Самое нужное «сверху»;
- Продуманная навигация;
- Локализация;
- Совместимость с другим софтом (соцсети);
- Скорость работы приложения;
- Информативность (сообщения / обязательные поля);
- Возможность отмены действий пользователя;
- Help — должна быть инструкция, как работать с приложением;
- Возможность печати (если нужно).

# Безопасность

Тестирование безопасности:

- Начинаем всегда с составления матрицы уровней доступа;
- Конфиденциальность — никто не может получить доступ к данным несанкционированно;
- Целостность данных:

а) возможность восстановить данные в полном объеме при их повреждении;  
б) доступ на изменение информации только определенной категории пользователей.

# Производительность

Производительность:

- Имитируем нагрузку пользователями (JMeter);
- Пробуем загрузить большие объемы данных, файлы, медиа;
- Нагружаем БД;
- Понижаем скорость инета (NetLimiter);
- Понижаем скорость передачи данных (Throttling);
- Тестируем восстановление системы после падений.



# Конфигурационное

Configuration

- Берем список софта и железа, на котором и с которым должно работать наше приложение.
- Думаем над тем, с чем еще взаимодействует приложение.
- Выписываем это все в список (ОС, браузеры, их версии для ПК, мобильных телефонов, планшетов, также (если это важно) выписываем на каком разрешении или с какими настройками (например, для камеры съемка в режиме HD) нужно проводить тестирование).
- Далее можем использовать метод классов эквивалентности, pairwise или просто руководствуемся тем, что есть в наличии, и настраиваем тестовое окружение с нужными конфигурациями.

## Установка

В общем случае такое тестирование проверяет множество сценариев и аспектов работы инсталлятора в таких ситуациях:

- новая среда исполнения, в которой приложение ранее не было инсталлировано;
- обновление существующей версии приложения;
- изменение текущей версии на более "старую";
- переинсталляция
- повторный запуск инсталляции после ошибки, приведшей к невозможности продолжения инсталляции;
- удаление приложения;
- установка нового приложения из семейства приложений;
- автоматическая инсталляция без участия пользователя.

Уже все?

Что в итоге?

Запоминаем схему тестирования приложений и сделаем ее удобной для запоминания:

*Внешнее — > Внутреннее — > Стойкость — > Взаимодействие*

**Внешнее** — проверка внешнего вида и функций, которые доступны только обычному пользователю (GUI, Usability).

**Внутреннее** — все функции приложения (Functional).

**Стойкость** — сюда мы отнесем устойчивость приложения к нагрузкам и к попыткам нарушить его безопасность (Security, Performance (load/stress/recovery)).

**Взаимодействие** — работа приложения с другим софтом и железом (Configuration).

Итоги

«Что?

Важно помнить, что тестирование ПО ставит перед каждым вступившим в стройные ряды сферы обеспечения качества ПО такие задачи, которые практически невозможно решать однозначно и по четкому алгоритму. Тестирование – это философия, творчество, полет мысли, основанный на четких и безоговорочных технических аспектах. Пожалуй, только тестировщикам приходится одновременно быть разработчиком, системным аналитиком, пользователем, заказчиком.

Чтобы тестирование вашего веб-приложения не оказалось запутанным и долгим процессом, не приносящим ожидаемых результатов, выберите нужный вид тестирования, придумайте эффективную стратегию тестирования.

И можно браться за тестирования любого приложения.



*Ссылки на используемые материалы:*

- <https://www.a1qa.ru/blog/vidy-testirovaniya-web-prilozhenij-kak-vybrat/>
- <https://dou.ua/lenta/articles/scheme-for-qa/>
- <https://quality-lab.ru/blog/key-principles-of-web-testing/>
- <https://otus.ru/nest/post/1851/>

*Всем спасибо)*