



Основы тестирования веб-приложений

Тестирование веб-приложений



Оглавление

[На этом уроке](#)

[Для чего необходимо тестирование веб-приложений](#)

[Браузеры и движки](#)

[В чём разница между WEB и mobile WEB](#)

[Адаптивная вёрстка](#)

[Адаптивная вёрстка через медиазапросы CSS3](#)

[Адаптивная вёрстка с jQuery](#)

[Адаптивная вёрстка с Bootstrap](#)

[Способы тестирования адаптивной вёрстки](#)

[User-Agent](#)

[UI-элементы на странице](#)

[Navbar](#)

[Buttons](#)

[Spinner](#)

[Input](#)

[Dropdown](#)

[Select](#)

[Search & Placeholder](#)

[Cards](#)

[Carousel](#)

[Tags input](#)

[Progress bars](#)

[Slider](#)

[Navigation](#)

[Pagination](#)

[Checkbox](#)

[Radio Button](#)

[Hamburger menu](#)

[Kebab-menu](#)

[Switches](#)

[Tooltip](#)

[Form](#)

[Modal window](#)

[Footer](#)

[Тестирование UI и UX](#)

[Инструменты для создания интерфейсов](#)

[Тестирование вёрстки \(PerfectPixel, Page Ruler\)](#)

[Кросс-браузерное и кросс-платформенное тестирование](#)

[Что такое SPA-приложение?](#)

[Контрольные вопросы](#)

[Дополнительные материалы](#)

На этом уроке

1. Узнаем, для чего необходимо тестирование веб-приложений
2. Рассмотрим разные браузеры и движки
3. Обсудим в чём разница между WEB и mobile WEB
4. Познакомимся с адаптивной вёрсткой
5. Рассмотрим User-Agent, UI-элементы на странице и тестирование UI и UX
6. Расскажем, какие бывают инструменты для создания интерфейсов
7. Подробнее познакомимся с тестированием вёрстки (PerfectPixel, Page Ruler), кросс-браузерным и кросс-платформенным тестированием
8. Рассмотрим SPA-приложения

Для чего необходимо тестирование веб-приложений

Веб-приложения на сегодняшний день — один из самых популярных видов приложений. В браузере можно делать практически всё. И каждый тестировщик в своей работе обязательно столкнётся с тестированием веб-приложений. В этом курсе мы рассмотрим принципы работы веб-приложений, основные инструменты, применяемые для тестирования, особенности, связанные именно с вебом. Для чего это необходимо, спросите вы, если можно просто посмотреть на сайт и сказать, что такой-то элемент отображается не так или кнопка «Купить» не работает. Верно, но на же необходимо понять, а почему эта кнопку не работает, а что именно не работает, а верно ли вообще разработчик сверстал страницу. Вот для того, чтобы дать ответ, что продукт качественный, нам и необходимо знать названия элементов, какие подходы к тестированию использовать и многое другое.

Браузеры и движки

За отрисовку HTML-документов и интерпретацию JavaScript-кода в браузере отвечают специальные компоненты так называемые движки браузера (browser engine). Иными словами, движок браузера похож на движок автомобиля, а сам автомобиль — наш браузер, нам не обязательно разбираться в движках, чтобы управлять «автомобилем». У современных браузеров обычно выделяется два: один отвечает за HTML и CSS, второй — за интерпретацию скриптов, написанных на JavaScript. Это отделение скорее архитектурно-логическое, нежели физическое: не получится запустить интерпретатор JavaScript из состава Google Chrome как отдельную программу.

На сегодняшний день есть несколько актуальных движков:

- **Gecko** — сегодня используется в Firefox и других продуктах, входящих в Mozilla, например, в почтовом клиенте Thunderbird. Соответствующий JS-движок — SpiderMonkey. Разные версии движка носят слегка различные наименования, так что мы можем встретить такие обозначения, как IonMonkey или OdinMonkey, но это относится к одному и тому же продукту.
- **Blink** — сейчас это наиболее популярный движок. Он основа браузера с открытым кодом Chromium, на базе которого, в свою очередь, разрабатывается браузер Chrome и многочисленные сторонние браузеры: Opera, «Яндекс.Браузер», Atom от «Майл.Ру» и другие. За основу Blink взят другой движок — WebKit, но сейчас они уже сильно отличаются. За JS-часть отвечает движок V8, который разработала компания Google.
- **WebKit** — движок для браузера Safari — стандартный браузер в macOS. WebKit основан на другом движке KHTML, на котором построен браузер Konqueror в Linux/KDE. JavaScript в браузере Safari обрабатывается движком Nitro.

Не так давно в эту компанию входили движки от Internet Explorer:

- **Microsoft Trident** — используется в Internet Explorer 11 и 12. Вместе с ним работает JS-движок Chakra.
- **EdgeHTML** — основа для Microsoft Edge, браузера в Windows 10.

Но Trident уже давно не поддерживается самой компанией Microsoft, а EdgeHTML просуществовал очень недолго и сейчас заменён на Blink, то есть внутри Internet Explorer находится Google Chrome. С

полным списком браузеров, которые основаны на Blink/Chromium, можно ознакомиться, например, в «Википедии».

Таким образом, при кажущемся многообразии браузеров на рынке, существует всего 3 актуальных технологии рендеринга веб-страниц, на которых проверяются веб-приложения.

В чём разница между WEB и mobile WEB

Современный темп жизни таков, что большинство пользователей посещает те или иные ресурсы с мобильных устройств.

Веб-приложение (WEB) — приложение, не требующее установки, все обновления происходят на сервере, доставляются пользователям сразу — достаточно просто перезагрузить страницу или выйти, а потом снова зайти в аккаунт. Но иногда для его работы нужно установить дополнительные библиотеки или использовать защищённые сетевые протоколы.

Мобильная версия (Mobile WEB) — приложение, адаптированное под мобильную версию. Адаптирован, значит, что пользователю будет комфортно работать с веб-приложением мобильного устройства. Чаще всего, такие мобильные версии открываются по отдельному URL: <https://m.auto.ru>, <https://www.m.avito.ru> и многие другие.

Скорее всего, некоторые из вас уже задались вопросом, когда вы открываете на мобильном устройстве сайт <https://youla.ru>, <https://alfabank.ru>, то в ссылке не приписывается буква m и у вас всё равно показывается мобильная версия сайта. Почему же так происходит? Всё просто, существует несколько способов перенаправления пользователей: с помощью PHP или .htaccess. Данный редирект работает ориентируясь на юзер-агент (user agent — о нём мы поговорим несколько позже). Иными словами, сервер понимает, с какого устройства (мобильного или ПК) к нему обращается пользователь и на такую версию сайта переадресовывает. Если у посетителя будет неизвестный юзер-агент, то редиректа не произойдёт. Существует второй способ — определять разрешение экрана и показывать ту версию сайта, подходящую под размер экрана пользователя. В обоих случаях используется адаптивная вёрстка.

Адаптивная вёрстка

Сверстать сайт для каждого устройства отдельно не получится. Количество устройств огромное, и у каждого устройства есть своя диагональ экрана. Адаптивная вёрстка меняет дизайн страницы в зависимости от размера экрана и ориентации девайса и считается неотъемлемой частью современной веб-разработки. Она позволяет существенно экономить и не отрисовывать новый дизайн для каждого разрешения, а менять размеры и расположение отдельных элементов. Рассмотрим несколько способов задания адаптивной вёрстки.



Адаптивная вёрстка через медиазапросы CSS3

Медиазапросы (media queries) — правила CSS, которые позволяют управлять стилями элементов в зависимости от значений технических параметров устройств. Иными словами, это конструкции, позволяющие определять на основании некоторых условий, какие стили надо использовать на веб-странице, и наоборот.

```
@media screen and (max-width: 640px) {  
  .wrap {  
    clear: both;  
    font-size: 1.3em; }  
}
```

В этом случае класс .wrap станет работать при ширине экрана меньше или равной 640 px.

```
@media screen and (min-width: 800px) and (max-width: 1200px) {  
  .wrap {  
    clear: both;  
    font-size: 1.3em; }  
}
```

А в этом примере класс .wrap будет работать при ширине экрана от 800 до 1200 px.

Адаптивная вёрстка с jQuery

Адаптивная вёрстка реализуется также через jQuery:

```

<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"></script>
<script type="text/javascript">
    $(document).ready(function(){
        $(window).bind("resize", resizeWindow);
        function resizeWindow(e){
            var newWindowWidth = $(window).width();

            // Если ширина меньше 600 px, используется таблица стилей для мобильного
            if(newWindowWidth < 600){
                $("link[rel=stylesheet]").attr({href : "mobile.css"});
            } else if(newWindowWidth > 600){
                // Если ширина больше 600 px, используется таблица стилей для десктопа
                $("link[rel=stylesheet]").attr({href : "style.css"});
            }
        }
    });
</script>

```

Адаптивная вёрстка с Bootstrap

Bootstrap — HTML, CSS и JS фреймворк с открытым исходным кодом, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений. С ним верстаются сайты в несколько раз быстрее, чем на «чистом» CSS и JavaScript. Этот фреймворк — набор CSS- и JavaScript-файлов, и, чтобы использовать эти файлы, они подключаются к странице. После этого доступны инструменты этого фреймворка: колоночная система (сетка Bootstrap), классы и компоненты.

.col-4	.col-4	.col-4
.col-sm-4	.col-sm-4	.col-sm-4
.col-md-4	.col-md-4	.col-md-4
.col-lg-4	.col-lg-4	.col-lg-4
.col-xl-4	.col-xl-4	.col-xl-4
.col-xxl-4	.col-xxl-4	.col-xxl-4

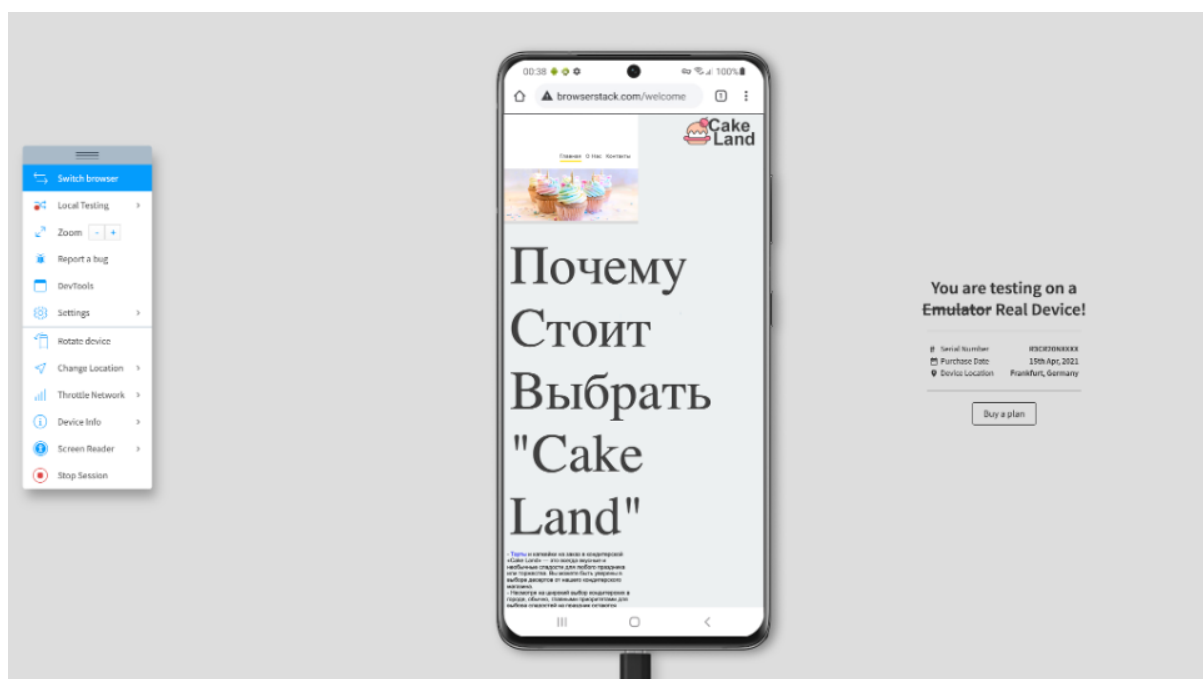
Способы тестирования адаптивной вёрстки

Как мы уже убедились, веб-приложение отображается по-разному на различных устройствах с разным разрешением экрана. Однако, где, например, взять столь разнообразное количество устройств? Для решения этого вопроса используются облачные фермы устройств (онлайн-сервисы, предоставляющие доступ к множеству устройств), такие как: BrowserStack, Firebase Test Lab, Samsung Remote Test Lab, AWS Device Farm, Sauce Labs, Xamarin Test Cloud, Perfecto и многие другие.

В качестве примера рассмотрим облачную, веб и мобильную платформу тестирования — BrowserStack. Она предоставляет как разработчикам, так и тестировщикам, возможность тестировать веб-приложения, а также мобильные приложения в браузерах на устройствах с различными операционными системами и даже на реальных мобильных устройствах.

Quick Launch		Real Devices (37)					
Android	Samsung	Galaxy S21	11	Galaxy S7	6	Galaxy J7 Prime	8
		Galaxy S21+	11	Galaxy S6	5	Galaxy Tab S6	9
iOS	Google	Galaxy S21 Ultra	11	Galaxy S5	4.4	Galaxy Tab S5e	9
	OnePlus	Galaxy S20	10	Galaxy S4	4.4	Galaxy Tab S4	8
Windows	Motorola	Galaxy S20+	10	Galaxy A51	10	Galaxy Tab S3	7
	Xiaomi	Galaxy S20 Ultra	10	Galaxy A11	10	Galaxy Tab S3	8
10	Vivo	Galaxy S10	9	Galaxy A10	9	Galaxy Tab 4 10.1	4.4
8.1	Oppo	Galaxy S9	8	Galaxy A8	7.1		
8	Huawei	Galaxy S8	7	Galaxy Note 20	10		
7	Others	Galaxy S10+	9	Galaxy Note 20 Ultra	10		
XP		Galaxy S10e	9	Galaxy Note 10	9		
Mac		Galaxy S9+	9	Galaxy Note 10+	9		
		Galaxy S9+	8	Galaxy Note 9	8.1		
		Galaxy S8+	9	Galaxy Note 8	7.1		
		Galaxy S8+	7	Galaxy Note 4	4.4		

Drag a browser here to add to Quick Launch



User-Agent

User-Agent — это HTTP-заголовок, отправляемый клиентом серверу, который содержит строку, описывающую клиентский браузер, операционную систему, разрядность и другие характеристики клиента.

Типичная строка заголовка User-Agent для современного браузера выглядит примерно так:

Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.60

Из неё довольно сложно понять, какой именно браузер используется — упоминается и Mozilla, и Chrome, и Safari. Такая «мешанина» пришла к нам из глубин браузерной истории, войн браузеров, систем с открытым исходным кодом и от разных стандартов.

Вначале был один браузер Mosaic, а у него — User-Agent NCSA_Mosaic/2.0. Через некоторое время появился второй браузер, который в процессе разработки назывался Mozilla, но его коммерческим названием стало Netscape Navigator. А вот в коде он так и остался Mozilla, поэтому его User-Agent был Mozilla/1.0.

Netscape считался более продвинутым браузером и поддерживал многие новейшие на тот момент технологии HTML, например, куки, фреймы и JavaScript. Веб-разработчики зачастую создавали более продвинутые страницы для Netscape, который идентифицирует себя как Mozilla, и версии попроще для остальных браузеров — в них не было слова Mozilla в строке User-Agent.

Чуть позже из того же браузера Mosaic вышел ещё один популярный продукт — Internet Explorer. Начиная с версии 2.0 он также поддерживал фреймы и другие полезные штуки, но сервера не спешили отдавать ему продвинутые версии страниц, потому что он — не Netscape.

Тогда в Microsoft решили добавить строку Mozilla в свой User-Agent, и заголовок стал выглядеть примерно так: Mozilla/1.22 (compatible; MSIE 2.0; Windows 3.1). А расшифровывался он как mozilla-совместимый MicroSoft Internet Explorer версии 2.0 на Windows 3.1.

Остальные браузеры, которые стали появляться позже, также унаследовали эту традицию и дописывали себе слово Mozilla.

Netscape Navigator и Internet Explorer активно развивались, и в определённый момент Microsoft добавила в User-Agent упоминание не только браузера, но и движка, и эта строка стала вида Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0) — так себя идентифицировал браузер Internet Explorer 8. Такая же штука появилась и в браузере Netscape, который к тому времени переименовался в Mozilla Firefox, и его User-Agent тоже стал содержать название движка: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.11) Gecko/20071127 Firefox/2.0.0.11.

Как ни удивительно, но «новый» браузер Mozilla Firefox неплохо справлялся с реализацией стандартов HTML/CSS/JS и очень хорошо отображал страницы. Слово Mozilla не было уникальным маркером для определения браузера, поэтому веб-разработчики стали вычислять Firefox по слову Gecko в строке User-Agent.

А тем временем под Linux вышла очередная версия браузера Konqueror, работающего на движке KHTML. Новая версия стала поддерживать различные стандарты не хуже Firefox, и чтобы получать те же современные версии сайтов, что и Firefox, в User-Agent добавилось упоминание его движка Gecko. Эти разработчики хотели сказать, что KHTML совсем как Gecko: Mozilla/5.0 (compatible; Konqueror/3.2; Linux) (KHTML, like Gecko).

KHTML, как и подавляющее число программ под Linux, считается продуктом с исходным кодом. Хорошее качество движка и открытый код — и компания Apple берёт именно эту разработку для создания собственного браузера Safari, движком которого стал переработанный KHTML под названием Webkit. Safari унаследовал от своего прародителя упоминание like Gecko: Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en-us) AppleWebKit/85.7 (KHTML, like Gecko) Safari/85.6, но добавил к нему название своего движка — AppleWebKit, а также название браузера — Safari.

Затем на основе Webkit появился движок Blink и браузер Chrome, идентифицируя себя практически как Safari, добавляя к той строке лишь версию Chrome:

Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.60 Safari/537.36

Такой User-Agent можно прочитать как «Я браузер Chrome 87, но работаю совсем как Safari на движке WebKit версии 537, который, вообще-то, движок KHTML, который совсем как движок Gecko, на котором основана Mozilla».

Краткий и смешной пересказ истории браузеров можно найти [на «Хабре»](#), а также есть [оригинал того поста на английском](#).

UI-элементы на странице

UI — это User Interface (дословно «пользовательский интерфейс») — то, как выглядит интерфейс. При тестировании любого веб-приложения вы будете сталкиваться со множеством элементов. Важно знать названия элементов и понимать принцип работы того или иного элемента. Мы рассмотрим часто встречающиеся элементы и поговорим про наиболее важные моменты, которые стоит учитывать при тестировании.

Kebab-menu, placeholder, navbar, switch и многие другие элементы часто можно заметить на различных сайтах. Чтобы помочь вам разобраться, кто из них кто, мы собрали гайд-шпаргалку по основным WEB UI.

Navbar



Navigation Bar (Навигационный бар) это часть пользовательского интерфейса, помогающая пользователю перемещаться по разным страницам веб-приложения. Во время тестирования панели навигации проверять необходимо корректную переадресацию на страницу, а также возвращение на предыдущую страницу, откуда мы пришли.

Buttons

Кнопки — элементы, которые позволяют осуществить или подтвердить какое-либо действие на странице. Во время тестирования стоит обращать внимание на состояние кнопок (активна/неактивна), на кейсы при клике на кнопки с отключённым соединением с интернетом.



Spinner

Спиннер — компонент, предназначенный для создания на веб-проектах загрузочной анимации, которая в основном используется для индикации на сайте не очень длительных по времени процессов (в среднем выполняющихся в пределах от 1 до 4 секунд). Следует проверять отображение и что этот элемент не «зависает» при загрузке контента.



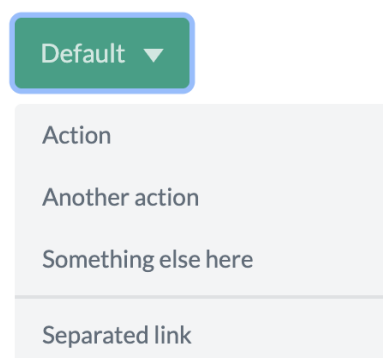
Input

Инпут — поле, позволяющее ввести какое-либо значение. При тестировании инпутов стоит вспомнить техники тест-дизайна (граничные значения, классы эквивалентности). Часто инпуты ограничивают по количеству допустимых символов как на клиенте, так и на сервере. Стоит учитывать проверки на отображение ошибок, если поле неверно заполнено.



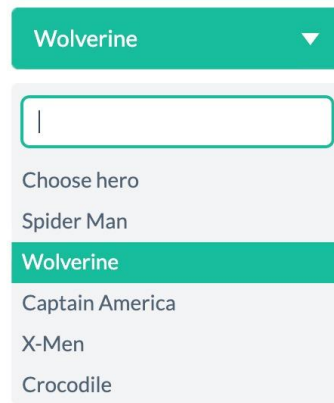
Dropdown

Раскрывающийся список — элемент, служащий для отображения списка выбора, из которого пользователь может выбрать одно или несколько значений. Выпадающие списки могут содержать неактивные элементы (те элементы, которые невозможно выбрать).



Select

Элемент, служащий для отображения списка выбора, с возможностью поиска по списку.

A UI element showing a dropdown menu. At the top is a teal button with the text "Wolverine" and a downward arrow. Below it is a white input field with a teal border. Under the input field is a list of options: "Choose hero", "Spider Man", "Wolverine" (highlighted with a teal background), "Captain America", "X-Men", and "Crocodile".

Search & Placeholder

Поле ввода даёт возможность указать значение с помощью клавиатуры, например, выполнить поиск.

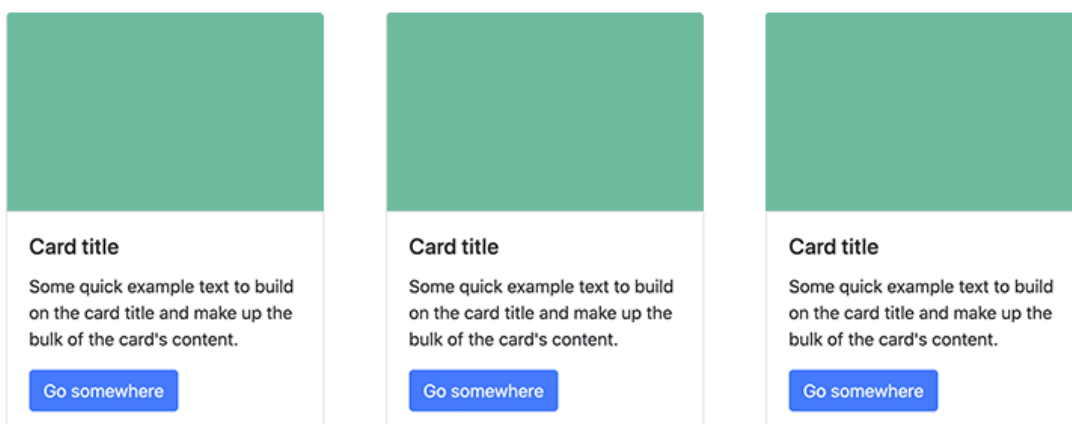
Если из названия не очевидно, как заполнять поле, используется плейсхолдер — подсказка, которая отображается внутри поля, пока оно не заполнено. При получении полем фокуса плейсхолдер становится светлее, при вводе первого символа исчезает.

Поле поиска может содержать садджесты (подсказки), которые должны совпадать по смыслу с вводимыми значениями.

A horizontal search bar. On the left is a magnifying glass icon followed by the text "Поиск" in a light gray font. On the right is a teal button with the text "Найти" in white.

Cards

Карточки — контейнер (какая-то группа, набор элементов) содержимого с множеством вариантов (например: карточки товаров, объявлений, блюд и т. д.). Часто в различных веб-приложениях ломается выдача и отображение карточек на разных разрешениях экранов.

Three identical card templates arranged horizontally. Each card has a teal header, a white body, and a teal footer. The body contains the text "Card title" and "Some quick example text to build on the card title and make up the bulk of the card's content." The footer contains a teal button with the text "Go somewhere".

Carousel

Карусель — это слайд-шоу для циклического просмотра серии контента. При тестировании каруселей уделять стоит отдельное внимание анимации контента.



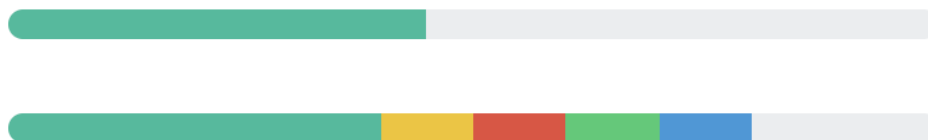
Tags input

Тэги или баблы — элемент выбора какого-либо параметра, чаще всего используется в фильтрах. Теги могут быть выбраны, а может пользователь их удалить.



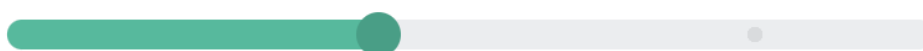
Progress bars

Индикатор выполнения — элемент, который позволяет показывать прогресс выполнения определённых задач вроде скачивания или загрузки, в основном всё, что занимает некоторое время. Во время тестирования стоит обращать внимание на анимацию и действительный прогресс выполнения (загрузки или индикатор выполнения каких-либо действий пользователя).



Slider

Слайдер или ползунок — предназначен для ввода данных в указанном диапазоне.



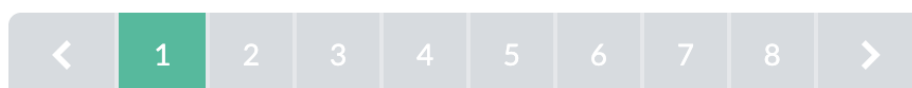
Navigation

Навигационная панель — элемент, позволяющий перемещаться между элементами страницы.



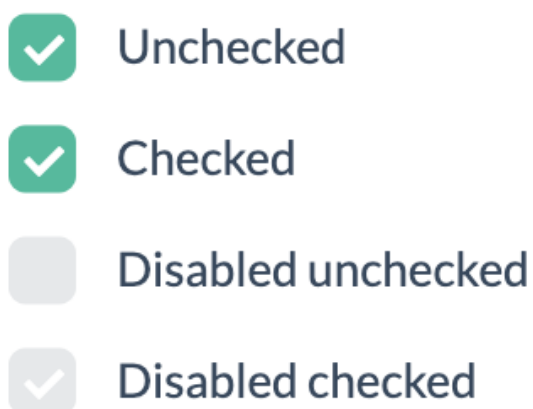
Pagination

Пагинация — отображение разбивки на страницы. При выборе той или иной страницы стоит обращать внимание на то, что действительно эта страница открывается.



Checkbox

Чекбокс используется для управления параметром с двумя состояниями. Используют чекбоксы в фильтрах, в различных меню для выбора дополнительных ингредиентов, опций и много другого.



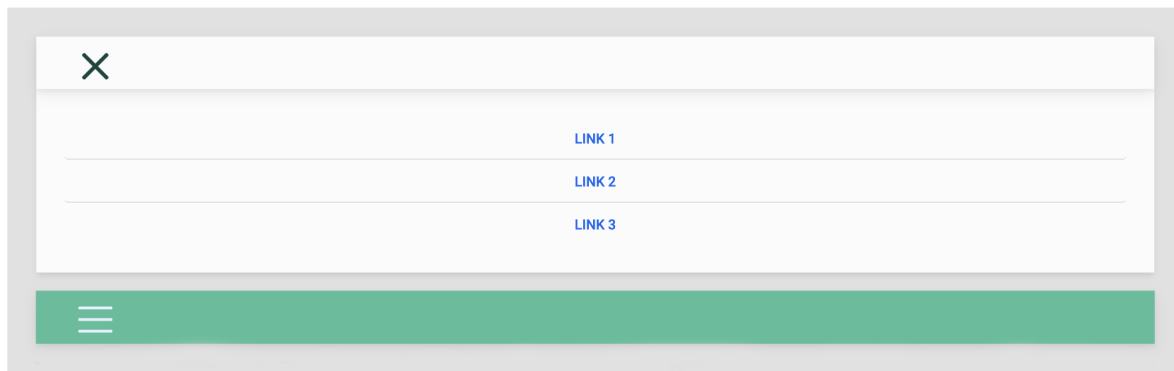
Radio Button

Группа радиокнопок используется для выбора одного значения из нескольких. Важно во время проверки учитывать кейс, что должно быть выбрано хотя бы одно значение, в отличие от чекбокса.

- ☐ Radio is off
- ☒ Radio is on
- ☐ Disabled radio is off
- ☐ Disabled radio is on

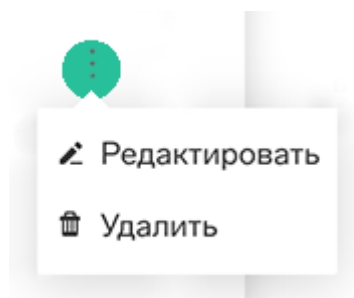
Hamburger menu

Гамбургер меню — меню, скрытое от пользователей.



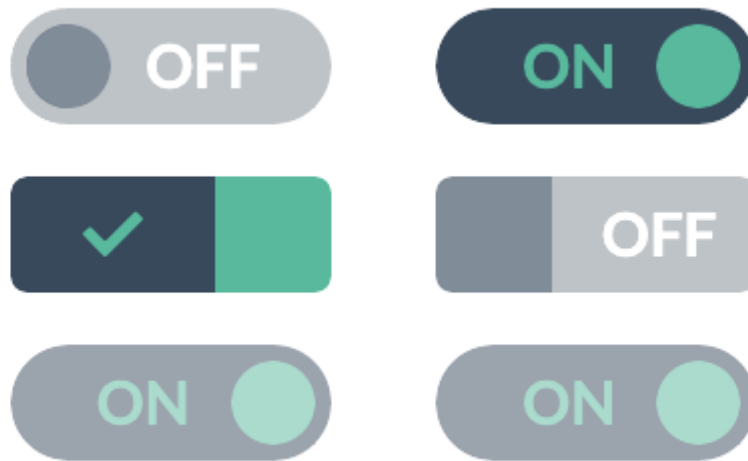
Kebab-menu

Кебаб-меню (потому что три точки, расположенные вертикально, напоминают люля-кебаб) содержит действия с объектом.



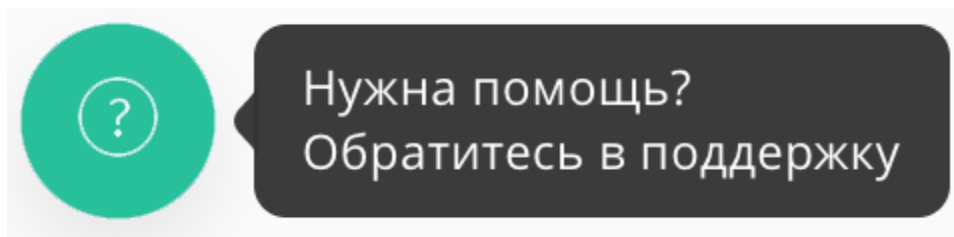
Switches

Переключает состояния. Например, включает или отключает уведомления в настройках.



Tooltip

Тултип — это подсказка, появляющаяся при наведении на элемент.



Form

Форма — раздел, позволяющий пользователю вводить информацию для последующей обработки системой.

Адрес электронной почты

Мы никогда никому не передадим вашу электронную почту.

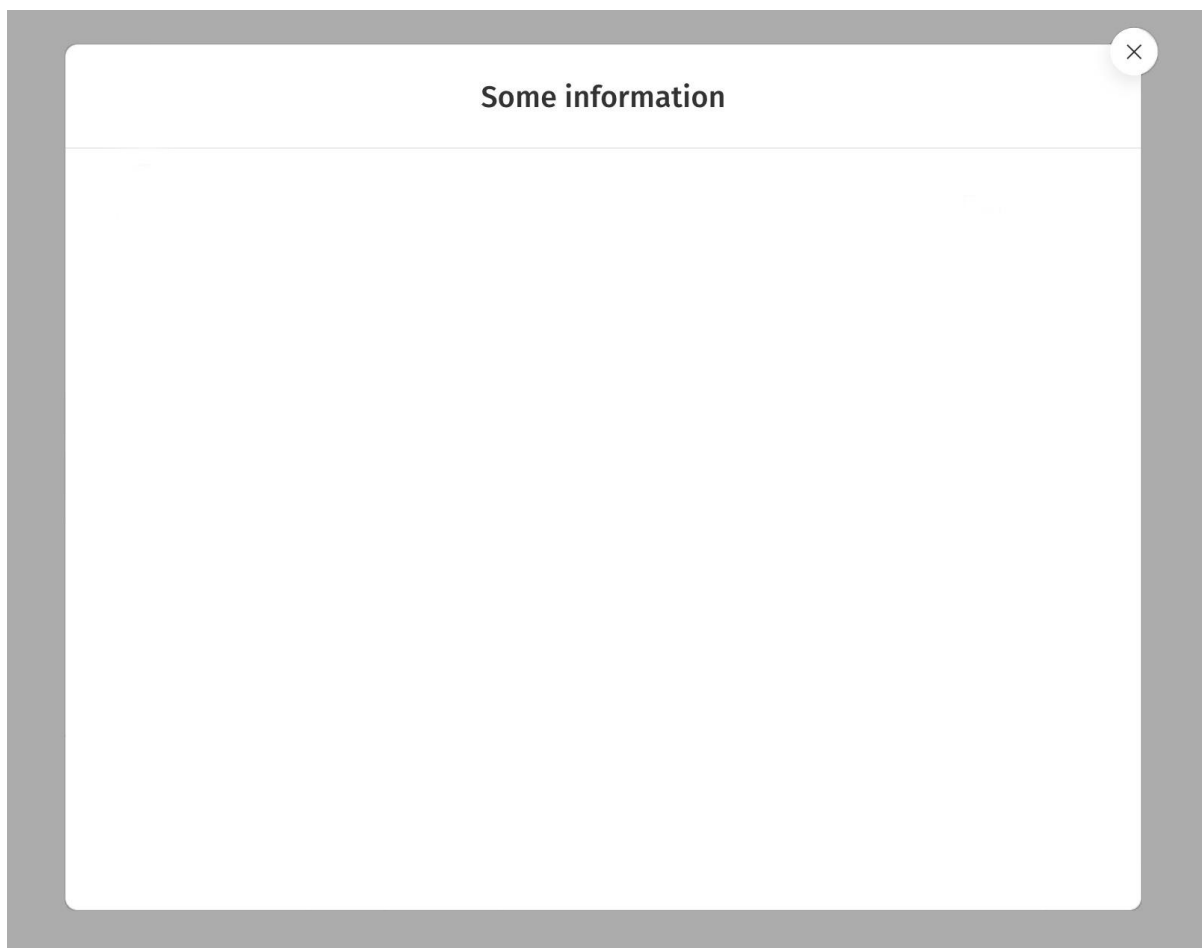
Пароль

☐ Проверить меня

Представлять на рассмотрение

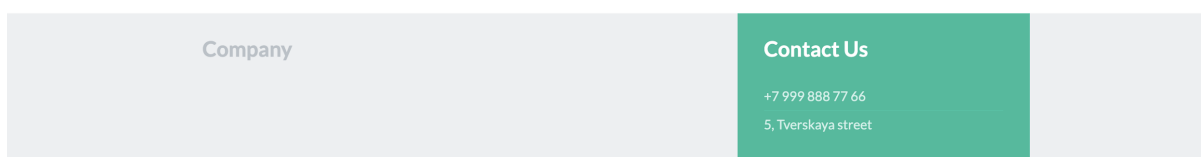
Modal window

Модальное окно — это элемент интерфейса, который визуально представляет собой «всплывающее окно», отображающееся над остальной частью страницы. При этом показ окна обычно сопровождают затемнением всей другой части страницы. Это действие позволяет визуально отделить его от остального содержимого страницы, а также показать, что в данный момент только оно одно является активным элементом.



Footer

Подвал сайта — раздел, где располагается автор сайта, дата документа, контактная и правовая информация и многое другое.



Тестирование UI и UX

UI и UX — многие из вас уже слышали об этих понятиях. Но что же на самом деле кроется за этими аббревиатурами на английском языке и так ли они просты, как кажутся?

Начнём с того, что сравнивать их между собой, не то чтобы трудно, скорее невозможно, и даже неправильно. Это два совершенно разных понятия, которые необходимо рассматривать по отдельности, не сливая суп и компот в один котёл. Вряд ли кто-то захочет дегустировать такое блюдо, не правда ли?

В то же самое время оба эти элемента имеют решающее значение для конечного результата и не могут существовать друг без друга. В общих чертах, UX Design или User Experience Design — это

аналитическая и техническая область разработки, в то время как UI Design или User Interface Design — то, что мы иногда ошибочно называем графическим дизайном. Хотя, на самом деле, всё гораздо сложнее.

~~Представим, что наш будущий продукт — это дом. Фундамент, каркас, опоры и балки в нём — это программный код, который придаёт ему структуру и не даёт развалиться на части. Стены, комнаты, дверные и оконные проёмы — это измерение UX, оптимизация пространства для его максимально комфортного использования. И User Interface — это уже отделка стен, ламинат или кафель, всё создающие внешний вид нашего дома и его уют для пользователей.~~

Представим, что у нас имеется проект внутреннего двора — это UI. Ландшафтный дизайнер поработал над проектом и визуализировал расположение дорожек. Проект реализовали, однако когда заселились жители, то стали появляться на газоне тропинки. Людям не удобны дорожки, которые ведут по более длинному пути — это измерение UX, оптимизация пространства для его максимально комфортного использования.

Итак, UI и UX — это сферы дизайна, которые нельзя смешивать воедино, но которые не могут существовать друг без друга. Отличный продукт начинается с UX, за которым неотрывно следует UI. Давайте же разберёмся, что представляет из себя каждый из них.

User experience — это процесс определения проблем пользователя и решение этих проблем с целью повышения его удовлетворённости. Улучшение удобства использования, простота использования и удовлетворение, которое клиент получает при взаимодействии с продуктом, определяют качество и востребованность этого продукта.

Вы не замечаете результата работы UX дизайнера, пока всё работает хорошо. Но если при работе с продуктом вы начинаете задаваться вопросом «Что же делать дальше, чтобы решить мою задачу?», то вам попался плохой UX.

User experience — это настоящая наука, сродни архитектуре или психологии. В её основе лежат исследования пользовательских групп, информационная архитектура, удобство использования, контент-стратегия. В результате объединения этих компонентов получают решение целей и задач тем способом, который имеет наибольший смысл для человека, использующего конкретную функцию. UX дизайнер — это помощник для пользователя. Он выслушает и вникнет во все проблемы клиента лучше любого психолога, создаст самые благоприятные условия для достижения поставленных перед дизайном целей.

Специалисты по тестированию (т. е. мы с вами, QA) занимаемся тестированием UI, а тестированием UX занимаются совершенно другие специалисты.

Инструменты для создания интерфейсов

Когда дизайнер подготовил макет, он передаёт его разработчику. Часто это выглядит просто как экспорт PNG и подготовка спецификации (ТЗ) со всеми необходимыми аннотациями. Для большого проекта подготовка спецификации может быть большой головной болью, потому что дизайн может часто меняться. Таким образом, традиционные (статичные) стайлгайды (это документ, где собраны пожелания заказчика относительно стиля) устаревают практически мгновенно.

Подготовка спецификации — трудозатратный процесс. Кроме того, каждый дизайнер делает её по-разному. В конечном счёте многие дизайнеры больше занимаются подготовкой спецификации, а

не дизайном. Разработчикам в таком случае приходится уточнять у дизайнеров информацию о деталях, которые недостаточно хорошо описаны в спецификации, или пользоваться Photoshop и Sketch, чтобы получить эти сведения.

Однако существуют инструменты для совместной работы дизайнеров пользовательских интерфейсов и фронтенд-разработчиков и нацелены на процесс перевода макета из Photoshop или Sketch в код. Загружается макет дизайна, инструменты превратят его в спецификацию и стайлгайд. Эти инструменты позволяют:

- изучать дизайн, подготовленный в Photoshop или Sketch, без необходимости использования оригинальных программ и быстро получать размеры в соответствии с платформой, для которой ведётся разработка;
- получать все необходимые элементы дизайна, такие как изображения и иконки, а также все свойства объектов (шрифты, цвета и размеры);
- экспортировать ресурсные файлы для любых элементов, будь то текст, кнопка или что-то ещё;
- добавлять комментарии или заметки для членов команды прямо в текущем макете.

Zeplin — один из инструментов для совместной работы дизайнеров UI и фронтенд-разработчиков. Дизайнеры экспортируют файлы из Photoshop или Sketch в Zeplin, и он отображает все свойства дизайна для разработчиков.

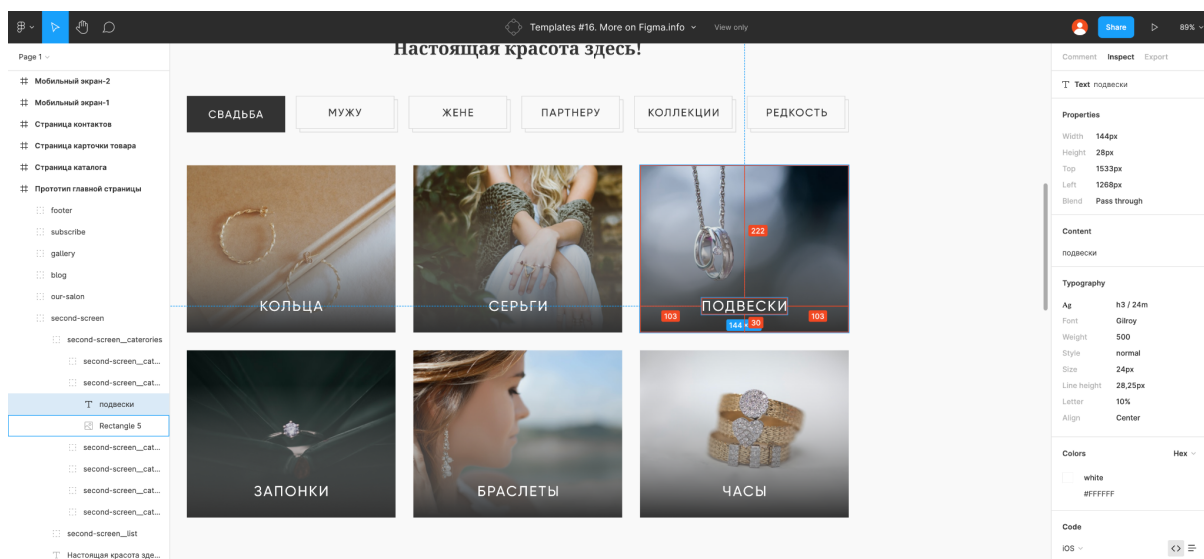
Figma — графический редактор для совместного проектирования сайтов, приложений и других дизайнерских продуктов. Она появилась как аналог Sketch и Adobe XD, но спустя пару лет стала одним из самых популярных инструментов у дизайнеров.

Тестировщику необходимо владеть базовыми навыками работы с одним из инструментов для того, чтобы уметь корректно протестировать интерфейс приложения.

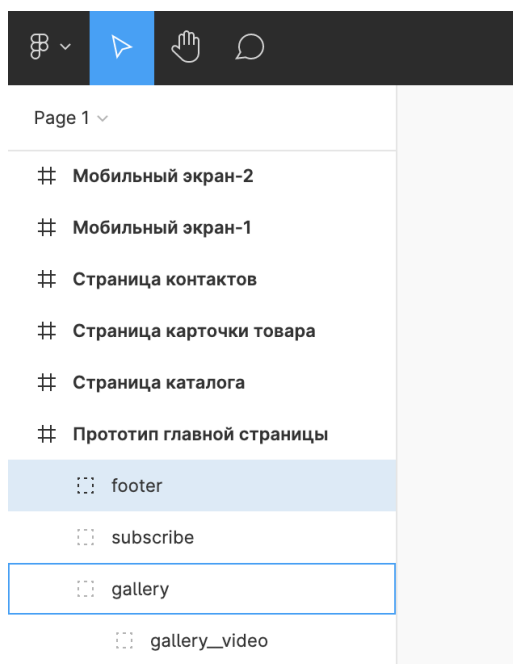
Давайте рассмотрим один из макетов, расположенных по адресу:

<https://www.figma.com/file/gYAFcIy0N9RaE0L5rsxlR8/Templates-%2316.-More-on-Figma.info?node-id=25%3A44>.

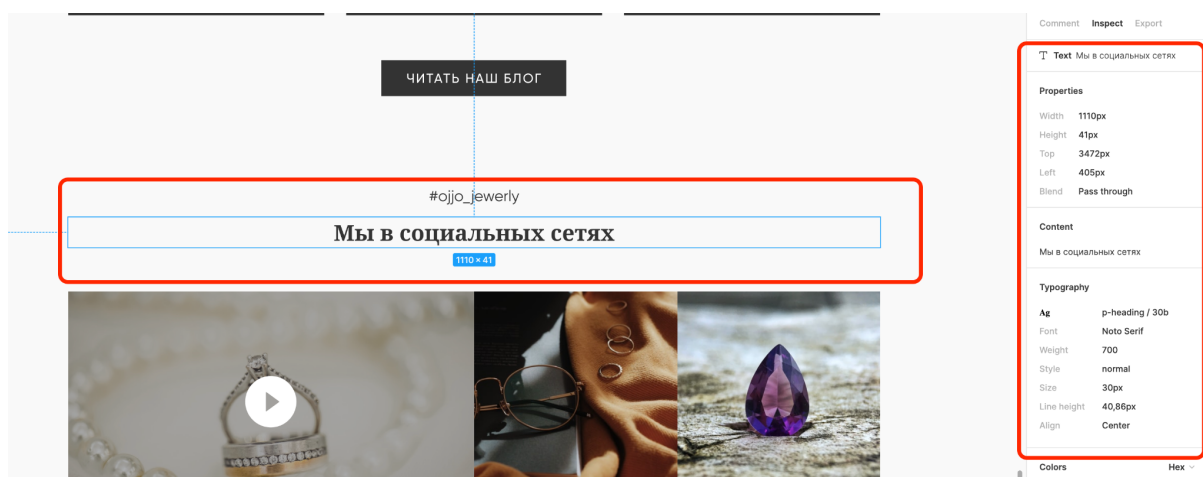
Перейдя по ссылке, мы попадаем на главный экран, где отображаются все отрисованные страницы нашего приложения.



В левой части экрана располагается навигация по разделам.



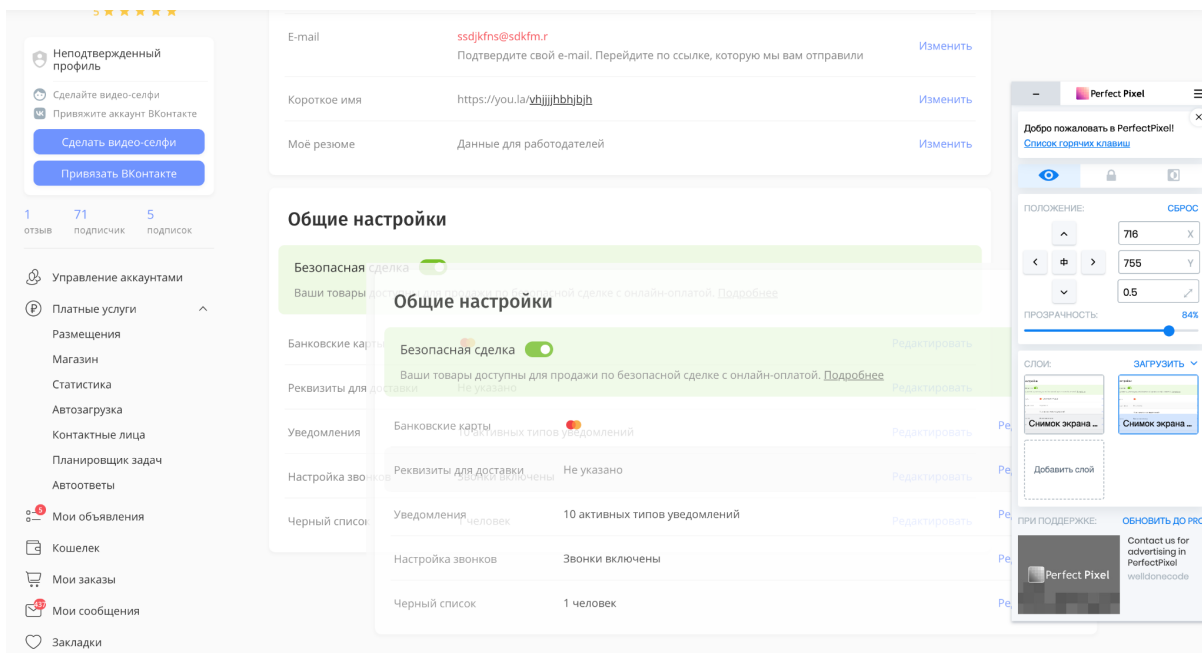
По центру расположен макет. Если мы наведём на конкретный элемент на макете курсор, то справа увидим, детальную информацию по выделенному элементу.



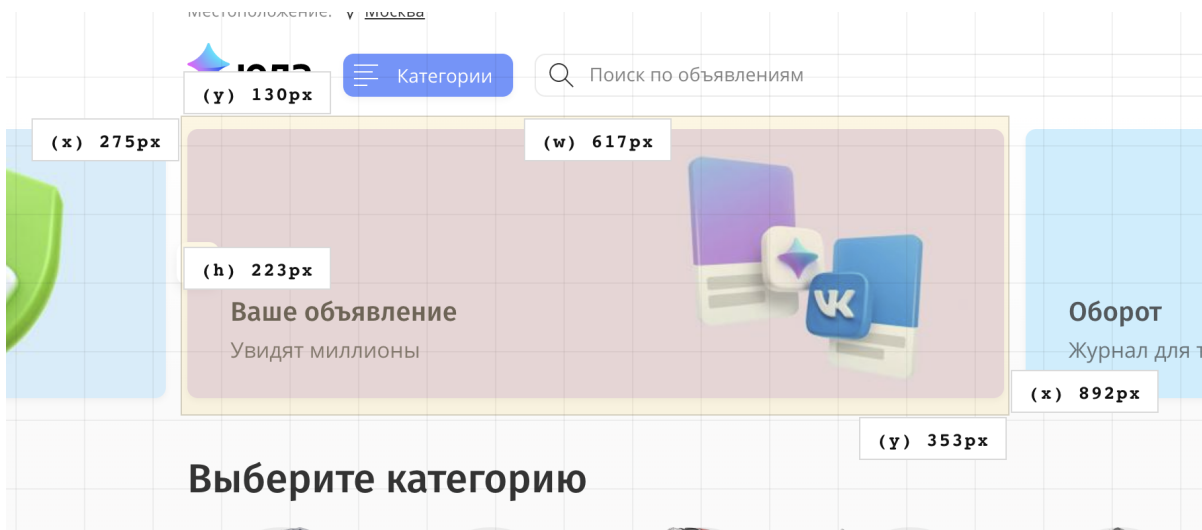
Тестирование вёрстки (PerfectPixel, Page Ruler)

Просто посмотреть на реализованное веб-приложение и сказать, что оно соответствует макету — не правильно. Необходимо проводить тестирование вёрстки. Существует множество инструментов, рассмотрим несколько из них.

Расположение кнопок, различных элементов должно быть 1:1 по сравнению с макетом. Допускается расхождение с макетом, чтобы не затягивать релиз, но чаще всего расположения, цвета — сверяют с макетом. Существует плагин Pixel Perfect (для [Mozilla Firefox](#) и [Google Chrome](#)).



Также существует плагин [Page Ruler](#) — это отличный инструмент для frontend разработчиков и тестировщиков, который помогает определить ширину, высоту и положение любого элемента на странице.



Кросс-браузерное и кросс-платформенное тестирование

Кросс-браузерное тестирование — это проверка отображения и работы веб-приложения в разных браузерах. Ручное кросс-браузерное тестирование достаточно доступно, так как браузеры — это бесплатное программное обеспечение, и любой пользователь может установить себе несколько браузеров, чтобы посмотреть отображение сайта и сравнить результаты работы приложения в разных браузерах.

Кросс-платформенное тестирование — это тестирование веб-приложения под разными операционными системами либо под разными видами устройств (десктоп, мобильный, планшет).

Такое тестирование, конечно, сложнее, так как требует аппаратных средств, установки и настройки операционной системы. Поэтому глубина кросс-платформенного тестирования определяется охватом пользователей, критичностью бага и готовностью компании инвестировать в это направление средства для подготовки и поддержания среды.

Условно всё «кросс» тестирование делится на следующие направления:

1. Браузеры: Chrome, Firefox, Safari, Internet Explorer, Edge, UC Browser, Opera и другие.
2. Операционные системы: Windows, Android, iOS, Mac OS, Linux.
3. Устройства: ПК (десктопы и ноутбуки), смартфоны, планшеты, телевизоры и другие.

Для эффективного тестирования требуется скомбинировать эти направления и составить список устройств, операционных систем и браузеров, в которых надо проводить тестирование. Этот список должен содержать информацию о приоритете того или иного набора. Например, высокоприоритетными могут быть связки «Ноутбук + Windows 10 + Chrome» или «iPhone 8 + iOS 13 + Safari», а низкоприоритетными, например, «PC + Linux + Firefox».

Если рассматривать кросс-браузерное тестирование со стороны работы приложения, то надо обратить внимание на две вещи:

1. Отображение страниц, то есть вёрстка (HTML + CSS).
2. Работоспособность различного интерактивного инструментария (JavaScript).

При оформлении бага, найденного при кросс-тестировании, не забываем указывать:

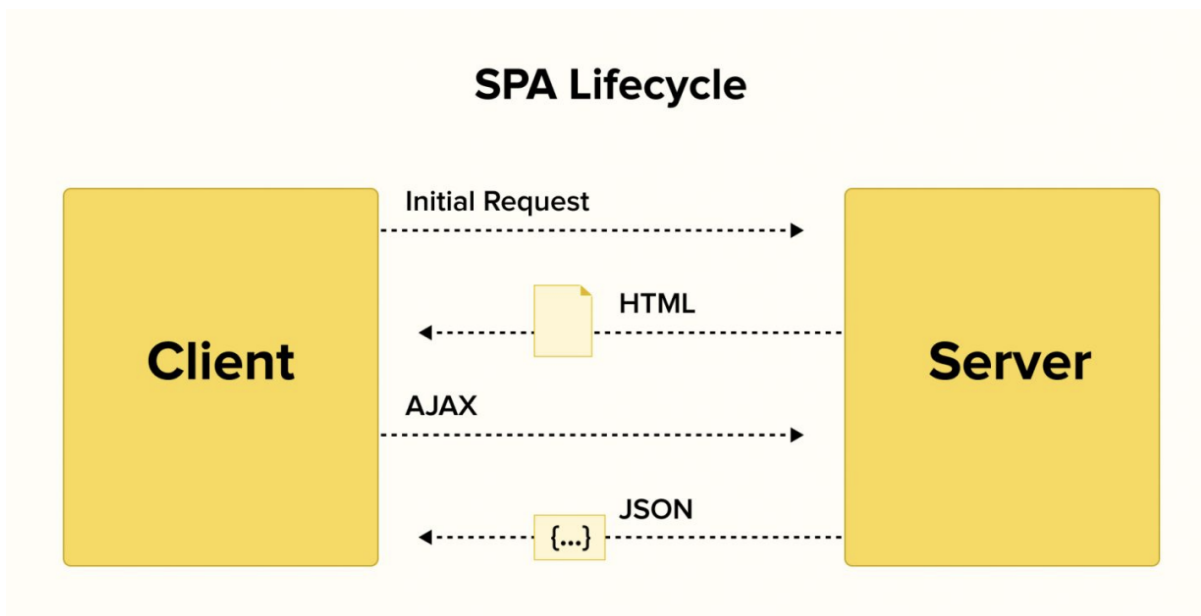
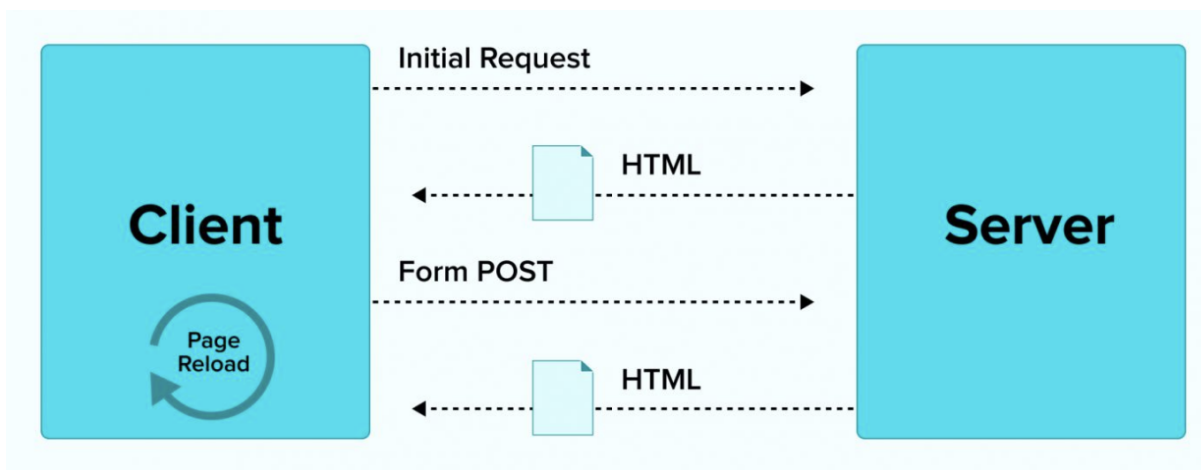
- операционную систему;
- версию ОС;
- браузер и его версию.

Некоторые тестировщики планируют кросс-браузерное тестирование, используя таблицу поддержки браузеров и операционных систем. Она позволяет убедиться, что покрыты все требуемые вариации и поддерживаемые десктопными и мобильными операционными системами браузеры, а неподдерживаемые или не нуждающиеся в проверке — исключены из тестирования.

Что такое SPA-приложение?

Значение термина SPA (Single Page Application) кроется внутри него самого. SPA — это одностраничное приложение, содержащее HTML-страницу, которая динамически (без полной перезагрузки) обновляется в ответ на действия пользователя. Архитектура приложения устроена так, что при первоначальном запуске посетитель видит основной контент-сайта в браузере, а новые данные загружаются на ходу по мере необходимости, например, при прокрутке или клике на иконку. Если вы когда-нибудь листали ленту ВКонтакте, то вы понимаете, о чём идёт речь.

Стандартная схема работы клиента и сервера выглядит следующим образом. Клиент запрашивает у сервера информацию, перезагружается страница и данные обновляются.



При переходе на новую страницу обновляется только часть контента, позволяя не загружать одни и те же элементы сайта множество раз. Подобный эффект для SPA обеспечивают современные JavaScript фреймворки (jQuery для небольших проектов и Angular для крупных).

Ajax (Asynchronous JavaScript and XML) — подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером.

AJAX работает так: он запрашивает у сервера недостающую информацию и добавляет её на страницу. А сама страница не перезагружается.

Например:

1. Посетитель нажимает в форме обратной связи кнопку «Отправить».
2. Браузер запускает скрипт, привязанный к кнопке.
3. Этот скрипт отправляет запрос на сервер и получает в ответ новую порцию информации от сервера. Страница не перезагружается, всё происходит внутри скрипта.
4. Скрипт смотрит, что ответил сервер, и вживляет новую информацию в старую страницу.
5. Страница не перезагрузилась, посетитель остался там же, где и был, только с новыми данными.

Примеры таких запросов:

- Получить список новых сообщений в чате, не перезагружая весь чат.
- Подгрузить новых товаров на витрину магазина, не перезагружая витрину.
- Получить новые рекламные баннеры на странице, не заставляя пользователя перезагружать страницу.
- Получить новые сообщения на стене, не дожидаясь, пока пользователь сам её перезагрузит.
- Ютуб этим пользуется, чтобы свернуть видео в маленький плеер в углу.
- Яндекс — чтобы показать поисковые подсказки.
- Службы доставки — чтобы в режиме реального времени показывать статус заказа.
- Бесконечные ленты в соцсетях — тоже отсюда. Когда вы доскроливаете до конца, на сервер улетает новый AJAX-запрос, и в ответ приходит новый контент.

Дополнительные материалы

1. Статья «[Архитектура микросервисов](#)».
2. [Документация по HTTP от Mozilla](#).
3. [Hypertext Transfer Protocol \(HTTP\) Status Code Registry](#) — список всех кодов ответа.
4. [Fiddler — Free Web Debugging Proxy](#) — сайт прокси-сервера Fiddler.
5. [Charles — Web Debugging Proxy Application](#) — сайт прокси-сервера Charles.
6. Для получения более детальной информации по работе сетей, в том числе и интернета, рекомендуем почитать о сетевой модели OSI. В качестве начальной точки возьмите [статью в «Википедии»](#).
7. Статья «[Как приручить Charles Proxy?](#)»
8. [Видеоурок](#) по работе с Charles Proxy.
9. Статья «[Первые шаги с Fiddler Classic](#)»