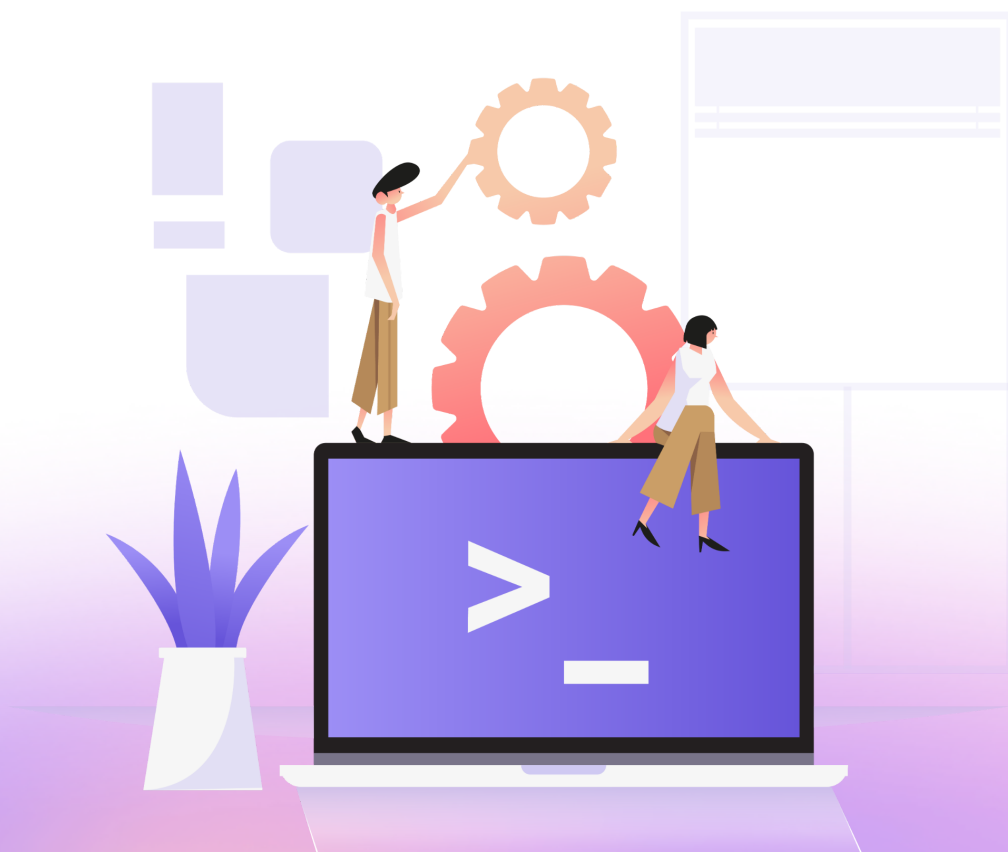


Тестирование веб-приложений

Браузерные движки и кросс-браузерное тестирование



На этом уроке

1. Разберёмся, что такое браузерный движок.
2. Узнаем, какие существуют браузерные движки.
3. Научимся эффективно проводить кросс-браузерное тестирование.
4. Посмотрим, как эффективно проводить тестирование на разных устройствах.

Оглавление

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

[Браузеры и движки](#)

[User-Agent](#)

[Статистика использования браузеров](#)

[Операционные системы](#)

[Мобильные ОС и браузеры](#)

[Цель и необходимость такого вида проверки](#)

[Место кросс-браузерного и кросс-платформенного тестирования в цикле проверки веб-приложения](#)

[Адаптивная вёрстка](#)

[Адаптивная вёрстка через медиазапросы CSS3](#)

[Адаптивная вёрстка с jQuery](#)

[Адаптивная вёрстка с Bootstrap](#)

[Способы тестирования адаптивной вёрстки](#)

[Эффективные подходы к кросс-браузерному тестированию](#)

[Оформление бага при кросс-браузерном тестировании](#)

[Как посмотреть версию?](#)

[Ночная сборка](#)

[Немного белого ящика](#)

[На что обращать внимание?](#)

[С чего начать проверку?](#)

[Надо ли тестировать все комбинации браузеров, ОС и разрешений экрана?](#)

[Надо ли тестировать на старых версиях и на самых-самых новых, например, бета-версиях ОС и браузеров?](#)

[Практическое задание](#)

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).

Браузеры и движки

За отрисовку HTML-документов и интерпретацию JavaScript-кода в браузере отвечают специальные компоненты так называемые движки браузера (browser engine). 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:

1. **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, по сути, это относится к одному и тому же продукту.
2. **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.
3. **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:

4. **Microsoft Trident** — используется в Internet Explorer 11 и 12. Вместе с ним работает JS-движок Chakra.
5. **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.

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 YaBrowser/20.12.0.963 Yowser/2.5 Safari/537.36
```

Из неё довольно сложно понять, какой именно браузер используется — упоминается и Mozilla, и Chrome, и Safari. Но эта строка описывает всё же «Яндекс.Браузер». 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, читается 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 прочесть 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](#).

Статистика использования браузеров

Для эффективного тестирования надо посмотреть, какими устройствами и браузерами пользуются посетители сайта, реальные или предполагаемые.

На скриншотах представлена статистика использования браузеров [по данным Liveinternet.ru](#):

Январь 2021

Январь 2020

✓ Google Chrome	1,925,265	54.5%
✓ Яндекс.Браузер	718,807	20.3%
✓ Mobile Safari	482,063	13.6%
✓ Opera (Blink)	162,849	4.6%
✓ Firefox	88,711	2.5%
✓ Safari	47,404	1.3%
✓ Chrome iOS	40,703	1.2%

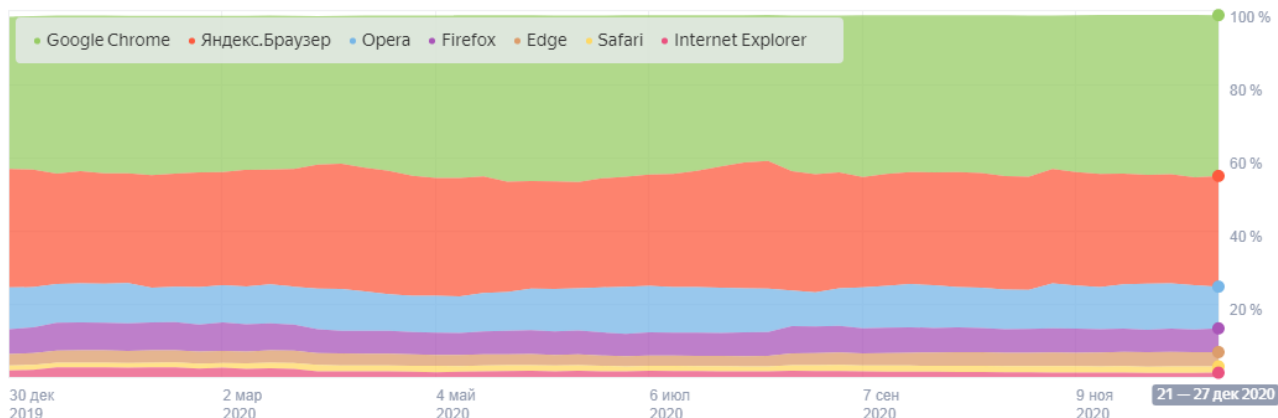
✓ Google Chrome	54,719,059	61.5%
✓ Яндекс.Браузер	12,307,435	13.8%
✓ Mobile Safari	8,695,668	9.8%
✓ Opera (Blink)	4,008,491	4.5%
✓ Firefox	2,782,327	3.1%
✓ VK App	1,420,394	1.6%
✓ Microsoft Edge	1,039,914	1.2%
✓ Explorer 11	920,637	1.0%

Здесь только браузеры с процентом использования больше 1.

У этой статистики есть один недостаток, который не позволяет полностью оценить картину: Google Chrome не поделён на десктопный и мобильный, а они сильно различаются. А вот мобильный Safari и обычный — разные.

Судя по статистике, для тестирования русскоязычного сайта достаточно двух десктоп-браузеров — Chrome и Firefox, и двух соответствующих мобильных браузеров — Chrome и Safari. «Яндекс.Браузер» и Опера используют тот же движок, что и Chrome, поэтому различий в рендеринге нет.

Ещё одним полезным сервисом для анализа популярности браузеров в русскоязычном интернете считается [«Яндекс.Радар»](#).






Его данные могут несколько отличаться от liveInternet, но самые популярные браузеры те же.

Операционные системы

Сейчас широко распространены три операционные системы для ПК:

1. Windows.
2. macOS.
3. Linux.

Подавляющее большинство компьютеров работает на Windows, чуть меньшее количество — хотя оно постоянно растёт — на macOS, а вот Linux используется редко. Зачастую это не персональные, а рабочие ПК, выход в интернет с которых ограничен. По данным Яндекса, с Windows заходит 95,84% всех пользователей ПК, с macOS — 3,21%, и с остальных операционных систем (все Linux) — порядка 1%.

▼  Windows	2 099 408 791	95,84%
▼  Mac OS	70 319 444	3,21%
 Остальные	20 826 385	0,95%

Стоит отметить, что только Windows и macOS обладают уникальными браузерами: для Windows это Internet Explorer и Edge, для macOS — Safari. Для Linux нет широко используемых уникальных браузеров, на подавляющем большинстве Linux-систем установлены Chrome и/или Firefox.

Набор браузеров и ОС, используемых посетителями, зависит от типа сайта, но в целом между ними не такая большая разница, как может показаться на первый взгляд. Посмотрим на таблицу, в которой указаны операционные системы пользователей, заходящих на два совершенно разнотипных сайта. Один из них — «Российская газета», издание Правительства РФ, второй — популярный развлекательный сайт:

Российская Газета — официальное издание Правительства России, rg.ru				Развлекательный портал «ЯПлакаль», yaplakal.com			
<input checked="" type="checkbox"/>	Android	318,783	65.8%	<input checked="" type="checkbox"/>	Android	97,244	45.3%
<input checked="" type="checkbox"/>	Windows 7	51,549	10.6%	<input checked="" type="checkbox"/>	Windows 7	42,492	19.8%
<input checked="" type="checkbox"/>	iOS iPhone	47,244	9.8%	<input checked="" type="checkbox"/>	Windows 10	38,472	17.9%
<input checked="" type="checkbox"/>	Windows 10	39,852	8.2%	<input checked="" type="checkbox"/>	iOS iPhone	15,799	7.4%
<input checked="" type="checkbox"/>	Windows 8	9,724	2.0%	<input checked="" type="checkbox"/>	Windows 8	6,980	3.3%
<input checked="" type="checkbox"/>	iOS iPad	5,990	1.2%	<input checked="" type="checkbox"/>	iOS iPad	4,267	2.0%
<input checked="" type="checkbox"/>	Windows XP	3,999	0.8%	<input checked="" type="checkbox"/>	MacOS	4,039	1.9%
<input checked="" type="checkbox"/>	MacOS	3,952	0.8%	<input checked="" type="checkbox"/>	Windows XP	2,750	1.3%
<input type="checkbox"/>	Unix	1,511	0.3%	<input type="checkbox"/>	Unix	1,889	0.9%
<input type="checkbox"/>	Windows Phone	949	0.2%	<input type="checkbox"/>	Windows Phone	280	0.1%
<input type="checkbox"/>	сумма выбранных	481,093	99.3%	<input type="checkbox"/>	сумма выбранных	212,043	98.8%
<input type="checkbox"/>	всего	484,247		<input type="checkbox"/>	всего	214,653	

Как видим, списки операционных систем, охватывающие 99% пользователей, идентичны, хотя процентное соотношение сильно различается. Если рассматривать 90% пользователей, то останется всего четыре операционные системы: Android, Windows 7, iOS, Windows 10.

Тестирование веб-приложений в Windows 7 и Windows 10 не различается. Сами операционные системы практически ни на что не влияют, разница будет только в браузерах, поэтому можно брать Windows 10. А вот тестирование на мобильных ОС и мобильных браузерах имеет свои особенности.

Мобильные ОС и браузеры

На сегодняшний день есть две популярные мобильные ОС — Android и iOS. Их соотношение — примерно 3.5:1.

▼	✓	🤖	Google Android	2 592 702 465	77,16%
▼	✓	🍏	iOS	763 618 994	22,73%

И, соответственно, два браузера под них — Mobile Chrome и Mobile Safari. Учитывая, что оба мобильных браузера входят в топ самых используемых браузеров, надо тестировать сайты под мобильные устройства.

Для полноценного мобильного тестирования требуются как минимум четыре комплекта: Android и iOS, смартфон и планшет. Версии ОС не критичны, но лучше всего исходить из правила, что версия iOS должна быть последней, а версия Android — предпоследней. Сейчас это iOS 14 и Android 11. Как правило, именно такие версии наиболее распространены среди владельцев устройств. От версии операционной системы зависит конкретная версия браузера, так как старые версии могут иметь устаревшие браузеры и плохую производительность.

Цель и необходимость такого вида проверки

Хотя составляющие веб-страницы (HTML, CSS, JS) стандартизированы, итоговый результат отображения зависит от браузера. Браузер может не поддерживать часть стандартов, например, устаревшие модификаторы, обрабатывать стили особым образом, интерпретировать JavaScript со своими особенностями. На скриншоте ниже видно, что довольно большой пласт свойств поддерживается браузерами совершенно по-разному:

	IE 11	Edge 85	Firefox 82	Chrome 88	Safari 14	Opera 70	
CSS user-select: none	Yes	-ms-	Yes	Yes	Yes	-webkit-	Yes
crypto.getRandomValues	Yes	-ms-	Yes	Yes	Yes		Yes
CSS Scroll Snap	Partial	-ms-	Yes	Yes	Yes		Yes
:placeholder-shown CSS pseudo-class	Partial	-ms-	Yes	Yes	Yes		Yes
CSS Grid Layout (level 1)	Partial	-ms-	Yes	Yes	Yes		Yes
Crisp edges/pixelated images	Partial	-ms-	Yes	Yes	Yes		Yes
Encrypted Media Extensions	Partial	-ms-	Yes	Yes	Yes		Yes
Web Cryptography	Partial	-ms-	Yes	Yes	Yes		Yes
matches() DOM method	Partial	-ms-	Yes	Yes	Yes		Yes
Media Queries: resolution feature	Partial		Yes	Yes	Partial	-webkit-	Yes
Full Screen API	Partial	-ms-	Yes	Yes	Partial	-webkit-	Yes
CSS text-stroke and text-fill	No	Yes	-webkit-	Yes	-moz-	Yes	-webkit-
CSS text-orientation	No	Yes	Yes	Yes	Yes	-webkit-	Yes
CSS line-clamp	No	Yes	-webkit-	Yes	-moz-	Yes	-webkit-
CSS color-adjust	No	Yes	-webkit-	Yes	-webkit-	Yes	-webkit-
Web Audio API	No	Yes	Yes	Yes	Yes	-webkit-	Yes
Screen Orientation	Partial	-ms-	Yes	Yes	No		Yes
CSS Appearance	No	Yes	Yes	Yes	Partial	-webkit-	Yes
CSS Reflections	No	Yes	-webkit-	No	Yes	-webkit-	Yes
CSS image-set	No	Yes	-webkit-	No	Yes		Yes
CSS Cross-Fade Function	No	Yes	-webkit-	No	Yes		Yes
CSS Backdrop Filter	No	Yes	No	Yes	Yes	-webkit-	Yes
CSS scrollbar styling	Partial	Partial	-webkit-	Partial	Partial	-webkit-	Partial
Filesystem & FileWriter API	No	Yes	-webkit-	No	No		Yes

Полная таблица — [на сайте Caniuse.com](https://caniuse.com). Этот сайт позволяет выбрать браузеры и области — CSS, HTML5, JS и так далее — для сравнения и по выбранным критериям строит таблицу.

Кроссплатформенное тестирование важно потому, что, например, мобильный Safari и Safari под Mac, Chrome под Windows и Chrome под Android — это разные сборки браузера. Соответственно, между ними есть различия. Этот же сайт даёт такую таблицу функций, которые поддерживаются по-разному, для обычного и мобильного Safari:

Mixed support		
	Safari 13	Safari & Chrome for iOS 13.3
display: flow-root	Yes	No
::selection CSS pseudo-element	Yes	No
CSS font-stretch	Yes	No
PNG favicons	Yes	No
Autofocus attribute	Yes	No
Web Notifications	Yes	No
Directory selection from file input	Yes	No
Encrypted Media Extensions	Yes	No
CSS touch-action property	No	Yes
HTML Media Capture	No	Yes
Inputmode attribute	No	Yes
Touch events	No	Yes
Server Timing	Partial	No
Media Capture from DOM Elements API	Partial	No
Feature Policy	Partial	No
Web App Manifest	No	Partial
Date and time input types	No	Partial
DeviceOrientation & DeviceMotion events	No	Partial
Web Animations API	No	Partial

Место кросс-браузерного и кросс-платформенного тестирования в цикле проверки веб-приложения

Кросс-браузерное и кросс-платформенное тестирование проводится далеко не в первую очередь. Это проверка на соответствие эталонному поведению в одном из браузеров, и она происходит на одном из последних шагов тестирования — после функционального тестирования, юзабилити-тестирования и так далее. Исключение — случай, когда под разные браузеры пишется заведомо разный код.

Детально проанализировать поведение приложения тяжело даже в одном браузере, не говоря уже обо всём многообразии браузеров, которое поддерживается приложением. Поэтому при постановке задачи на кросс-браузерное тестирование определяется круг браузеров, платформ и функций для проверки.

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.

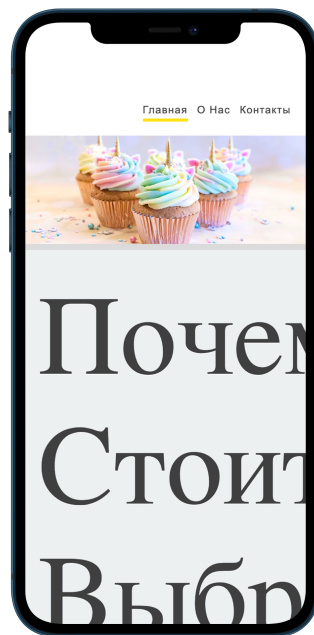
```
@media screen and (max-width: 640px) {  
  .wrap {  
    clear: both;  
    font-size: 1.3em; }  
}
```

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.

На сайте кондитерской Cake Land используются медиазапросы для различных разрешений экранов. Однако, при разработке сайта, разработчик допустил ошибку, и сайт некорректно отображается на устройствах с шириной экрана менее 640 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.

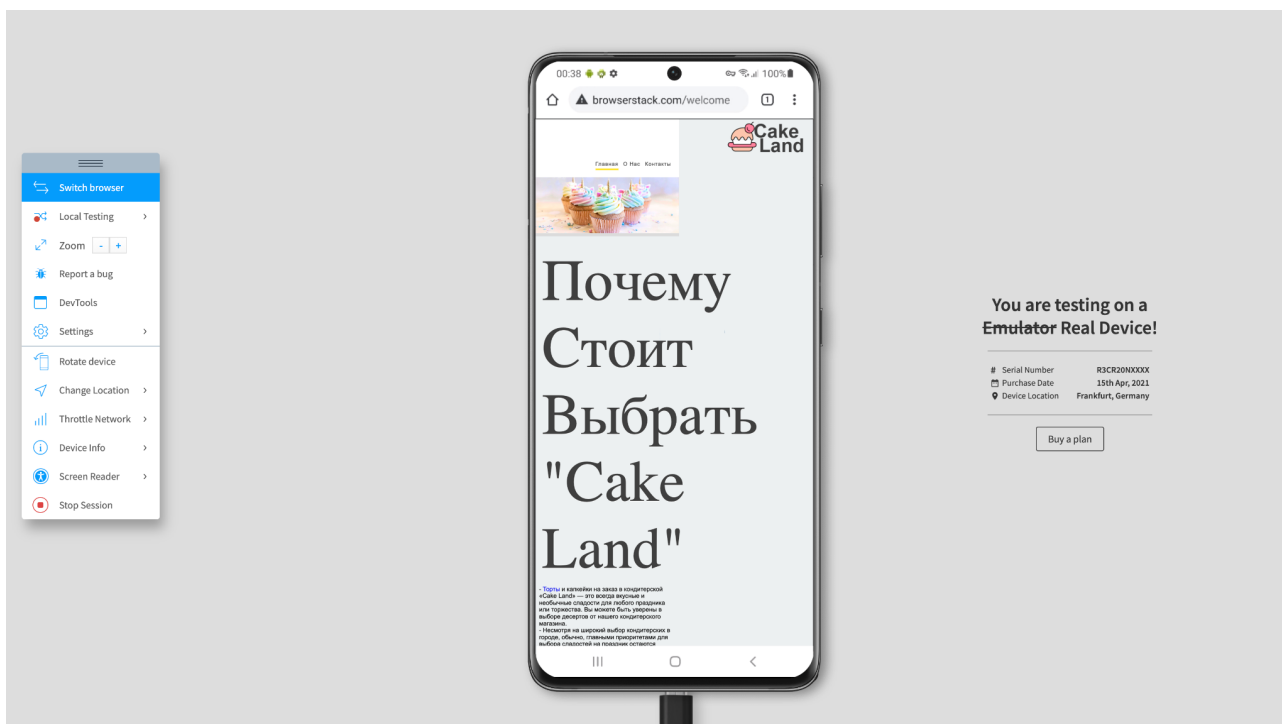
.col-4	.col-4	.col-4
.col-sm-4	.col-sm-4	.col-sm-4
.col-md-4	.col-md-4	.col-md-4
.col-lg-4	.col-lg-4	.col-lg-4
.col-xl-4	.col-xl-4	.col-xl-4
.col-xxl-4	.col-xxl-4	.col-xxl-4

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, 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
🍏 iOS		Google	▶	Galaxy S21+	11	Galaxy S6	5	Galaxy Tab S6	9
				Galaxy S21 Ultra	11	Galaxy S5	4.4	Galaxy Tab S5e	9
🪟 Windows		OnePlus	▶	Galaxy S20	10	Galaxy S4	4.4	Galaxy Tab S4	8
		Motorola	▶	Galaxy S20+	10	Galaxy A51	10	Galaxy Tab S3	7
🪟 10		Xiaomi	▶	Galaxy S20 Ultra	10	Galaxy A11	10	Galaxy Tab S3	8
🪟 8.1				Galaxy S10	9	Galaxy A10	9	Galaxy Tab 4 10.1	4.4
🪟 8		Vivo	▶	Galaxy S9	8	Galaxy A8	7.1		
🌈 7		Oppo	▶	Galaxy S8	7	Galaxy Note 20	10		
🌈 XP				Galaxy S10+	9	Galaxy Note 20 Ultra	10		
🍏 Mac	+	Huawei	▶	Galaxy S10e	9	Galaxy Note 10	9		
				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					



Эффективные подходы к кросс-браузерному тестированию

Оформление бага при кросс-браузерном тестировании

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

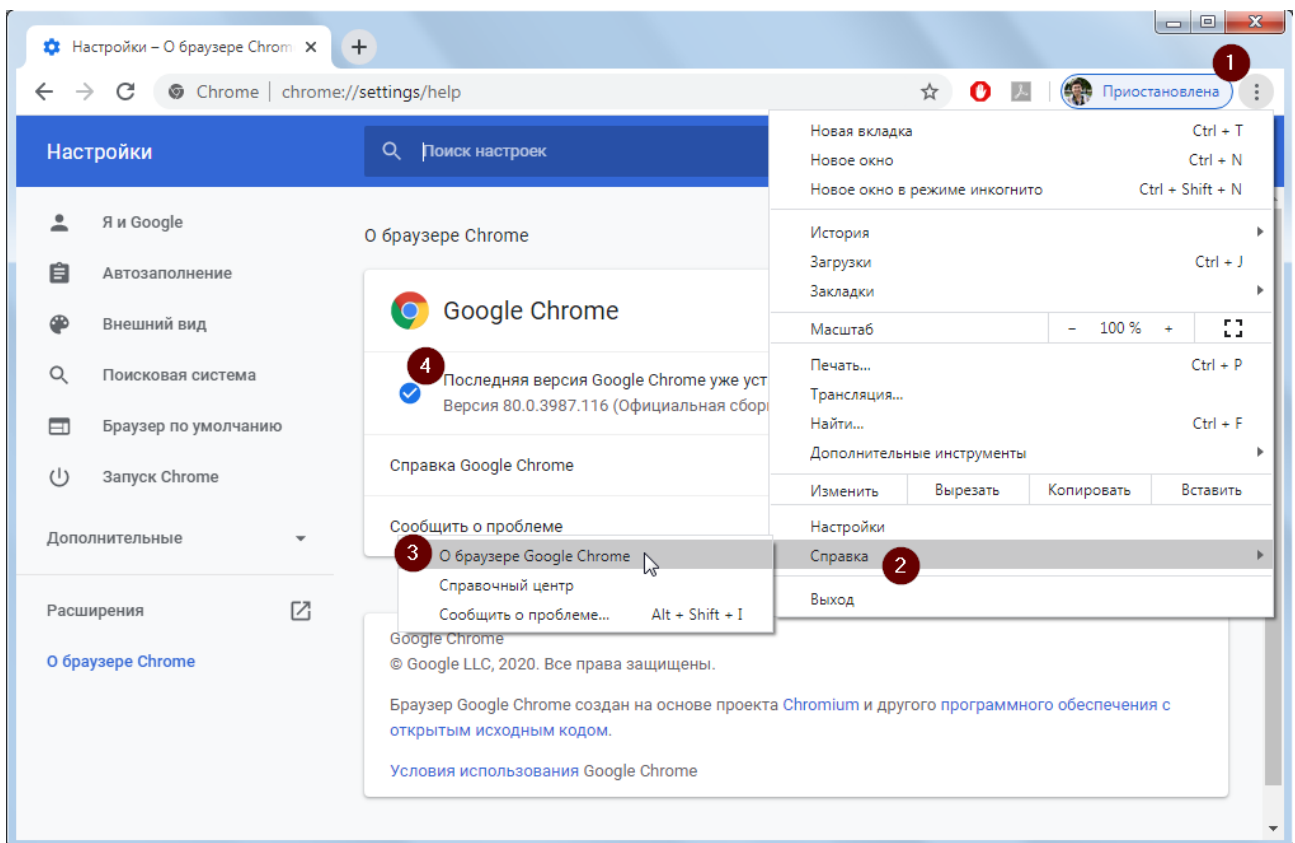
- операционную систему;
- версию ОС;
- браузер и его версию.

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.

Как посмотреть версию?

В Chrome и Firefox по URL about: через меню браузера можно найти информацию о версии и сборке браузера. В Internet Explorer и Edge эта информация находится в пункте меню «О приложении» — последний пункт правого выпадающего меню адресной строки. Для Safari эта информация располагается в меню «О Safari», доступном через меню Apple.

Например, Chrome:



Ночная сборка

Помимо основных сборок браузера, которые считаются официальной, проверенной, поддерживаемой и рекомендованной версией, у браузеров есть ещё так называемая ночная сборка — доступная для установки версия. Но она пока находится в разработке. Её нет смысла использовать для тестирования в контексте охвата пользователей, так как сама версия может содержать какие-то проблемы, а также изменения в поддерживаемых возможностях. Но её полезно использовать для проверки на опережение: не сломается ли что-то при выходе нового браузера.

У основных браузеров есть каналы обновлений. [Chrome Canary](#), [Firefox Developer Version](#) и [Safari Technology Preview](#) дают возможность посмотреть на новую версию заранее.

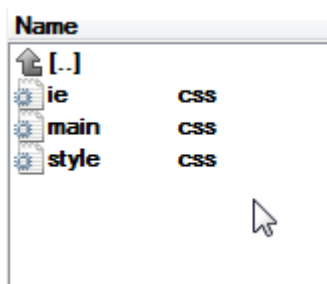
Немного белого ящика

Тестирование белого ящика предполагает ознакомление тестировщика с исходным кодом веб-страницы и кодом, её генерирующим. Если в коде есть места со специфической обработкой под конкретный браузер, например, для такого-то — один стиль, для другого — другой, этот момент обязательно тестируется.

Большинство браузерозависимых багов часто проявляется в Internet Explorer 11 и Safari из-за их закрытого исходного кода. Разработчикам зачастую приходится использовать обходные пути, чтобы

добиться единообразия в поведении приложения в разных браузерах, поэтому поведению веб-элементов в этих браузерах надо уделять больше внимания.

Например, возможен отдельный файл со стилями под конкретный браузер:



Возможны отдельные правила стилей для разных устройств:

```
@media all and (device-width: 768px) and (device-height: 1024px) and
(orientation:portrait) {
  .site_dg { width: 1000px;} /* your css rules for ipad portrait */
  .container { width: 1000px;}
}
```

```
@media all and (device-width: 768px) and (device-height: 1024px) and
(orientation:landscape) {
  .site_dg { width: 1000px;} /* your css rules for ipad portrait */
  .container { width: 1000px;}
}
```

Или «хаки» под разные браузеры:

```
@font-face {
  font-family: 'SegoeUIRegular';
  src: url('segoeui.eot');
  src: url('segoeui.eot?#iefix') format('embedded-opentype'),
  url('segoeui.woff') format('woff'),
  url('segoeui.ttf') format('truetype'),
  url('segoeui.svg#SegoeUIRegular') format('svg');
}
```

Если в коде обнаруживается подобное, значит, есть специальная обработка для браузера, и надо проверить её корректность.

На что обращать внимание?

Помимо общего вида страницы, отдельно проверяем изменения на странице при взаимодействии с пользователем: выпадающие списки, подсветку элементов, использование прозрачности и анимации.

С чего начать проверку?

Элементы интерфейса:

- кнопки;
- поля;
- текстовая область;
- фон;
- шрифты;
- и прочее.

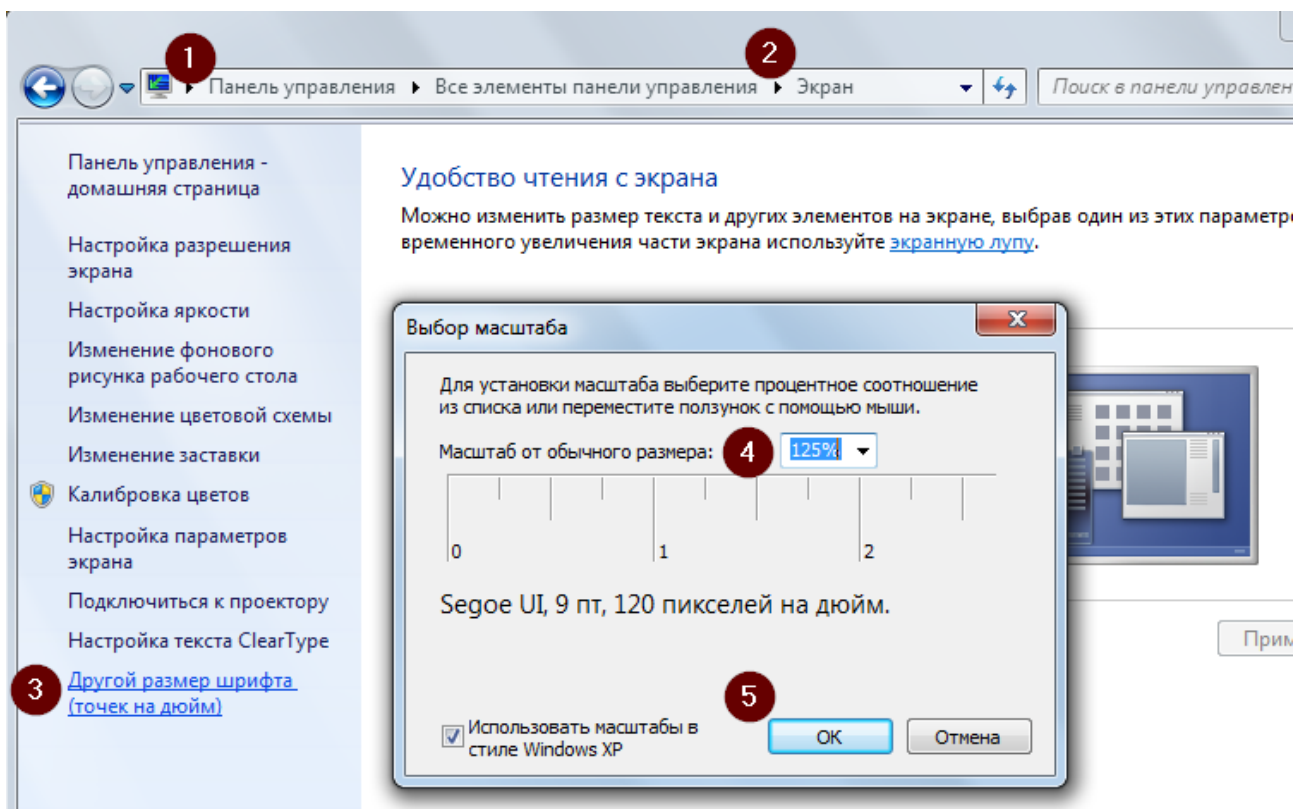
должны выглядеть одинаково во всех браузерах.

Размер окна

Попробуйте уменьшить размер окна до минимально возможного и убедитесь, что все элементы можно использовать вне зависимости от размера экрана. При реактивном дизайне элементы должны каким-либо образом «схлопываться», когда размер экрана уменьшается. Поищите места, где этого схлопывания не произошло, но элементы дизайна пропали.

DPI

DPI (Dots Per Inch) — количество точек на дюйм, настройка экрана в системе.



Убедитесь, что все элементы верно отображаются на экранах и с высоким DPI, и с низким.

Скролл

Можете ли вы проскроллить экран, правильно ли это происходит? Тачпад, тач-кнопка, указатель мыши, колесо мыши, кнопки вверх/вниз — всё, что заставляет приложение скроллиться, должно вызывать это действие. Не забудьте проверить и вертикальный, и горизонтальный скролл там, где применяются оба.

Масштабирование

Изменится ли что-то, если вы увеличите масштаб отображения страницы в браузере? Если используется отзывчивый дизайн, элементы изменяют свои размеры.

Обращайте внимание на необъяснимые ошибки Javascript, связанные с поведением приложения в браузере.

Надо ли тестировать все комбинации браузеров, ОС и разрешений экрана?

Общий ответ: нет, не надо. Важно исходить из имеющихся ресурсов и разумной достаточности. Например, если у вас нет 12-дюймового iPad Pro, а есть только обычный iPad Air, то, конечно, этого планшета будет достаточно для проверки вашего веб-приложения. Тестировать в разрешениях

1280x1024, 1366x768, 1920x1080 и так далее на каждом браузере также излишне. В этом случае эффективен комбинаторный подход к составлению тестового окружения.

На следующем занятии мы узнаем, как для этого используется консоль разработчика в Chrome.

Надо ли тестировать на старых версиях и на самых-самых новых, например, бета-версиях ОС и браузеров?

На самых-самых новых — надо, но во вторую очередь, когда всё проверено на основных связках ОС + браузер.

На старых тестировать стоит только в том случае, если ими по какой-то причине пользуется значительное количество ваших клиентов. Это могут быть корпоративные устройства, встроенные устройства и так далее.

Практическое задание

1. Составьте набор окружений, на которых надо проводить тестирование совместимости сайта <https://gb.ru> или сайта, выбранного вами. Отсортируйте окружения по уменьшению приоритета.
2. Проведите кросс-браузерное и кросс-платформенное тестирование:
 - а. Зарегистрируйте учётную запись на [этом сервисе](#). Воспользуйтесь [этим сервисом](#), если регистрируетесь не по своей почте.
 - б. В отчёт прикрепите следующие скриншоты отображения страницы логина <https://gb.ru/login> в тех браузерах, в которых считаете необходимым проверить:
 - в macOS Big Sur;
 - в OS Windows;
 - в iOS mobile web: iPhone 11;
 - в iOS mobile web: iPad (любой);
 - в Android mobile web (любое устройство).
3. Откройте [проект](#) Cake Land, клонированный ранее. Затем откройте симулятор mobile в DevTools, который открывали на уроке. Найдите баги в вёрстке и заведите баг-репорты — не менее одного баг-репорта.

Формат сдачи: excel, .pdf, .word.

4. Тест для самопроверки <https://coreapp.ai/app/player/lesson/614a22c68478af554b6f4dc1>
(сдавать не нужно)

Глоссарий

Браузерный движок (browser engine) — компонент, отвечающий за отрисовку HTML-документов и интерпретацию JavaScript-кода в браузере.

Кросс-браузерное тестирование — это проверка отображения и работы веб-приложения в разных браузерах.

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

Медиазапросы (media queries) — правила CSS, которые позволяют управлять стилями элементов в зависимости от значений технических параметров устройств.

Используемые источники

1. [Статистика использования браузеров в России.](#)
2. [Версии Android.](#)
3. Статья о том, [как начать кросс-браузерное тестирование](#)».
4. [Генерация скриншотов 1.](#)
5. [Генерация скриншотов 2.](#)
6. [Сравнение изображений.](#)

Дополнительная литература

1. [История браузеров.](#)
2. [Эволюция интернета.](#)