**Лабораторная работа №15**

**«Документирование результатов тестирования»**

**Цель работы:** составить итоговый отчет о результатах тестирования

приложения.

Используемые источники:

<http://old.code.mu/tasks/javascript/base/priemy-raboty-s-massivami-na-javascript.html>

**Порядок выполнения работы**

1. Ознакомиться с краткими теоретическими сведениями и заданиями к работе.
2. Выполнить задания работы.
3. Оформить отчет о выполнении.
4. Сделать выводы.
5. Ответить на контрольные вопросы.

**Теоретические указания**

Итоговый отчет можно разделить на части с соответствующей информацией:

* Приветствие.
* Общая информация (Common Information).
* Тестовое окружение (Test Platform).
* Рекомендации QA (QA Recommendations).
* Детализированная информация (Detailed Information).
* Окончание содержимого.

Приветствие

Свое письмо с отчетом необходимо начать с приветствия всех адресатов. Если по каким-либо причинам произошла задержка данных отчета, либо не весь запланированный функционал был проверен, то эту информацию необходимо предоставить в начале письма. Следует извиниться за задержку и указать адекватные причины произошедшего. Также в самом начале письма следует указывать, если были какие-то внешние факторы, препятствующие проверке какой-то части функционала.

Если во время тестирования не произошло никаких форс-мажорных обстоятельств, то достаточно обычного вежливого приветствия и далее уже переход к следующим пунктам.

Общая информация (Common Information)

В данной части отчета описывается, какие виды тестов проводились. Зачастую указываются модули, которые тестировались или функционал. Стоит удостовериться, не забыта ли какая-то часть функционала, особенно это актуально, когда нужно собрать итоговый отчет, соединив в себе данные о разных видах тестов и функционале.

Тестовое окружение (Test Platform)

Как правило, в этой части указываются:

* Название проекта.
* Номер сборки.
* Ссылка на проект (сборку). Необходимо убедиться, что зайдя по этой
* ссылке вы действительно попадаете на проект или можете установить приложение.

При указании данных в этой части отчета нужно быть очень внимательным, т.к. неправильная ссылка на сборку или неверный номер сборки не дают достоверной информации всем заинтересованным людям, а также затрудняют работу человеку, собирающему финальный отчет.

Рекомендации QA (QA Recommendations)

Данная часть отчета является наиболее важной, т.к. здесь отражается общее состояние сборки. Здесь показывается аналитическая работа тестировщика, его рекомендации по улучшению функционала, наиболее слабые места и наиболее критичные дефекты, динамика изменения качества проекта.

В этом разделе должна быть информация о следующем:

* Указан функционал (часть функционала), который заблокирован для проверки. Даны пояснения почему этот функционал не проверен (указаны наиболее критичные дефекты).
* Произведен анализ качества проверенного функционала. Следует указать, улучшилось оно или ухудшилось по сравнению с предыдущей версией, какое качество на сегодняшний момент, какие факторы повлияли на выставление именно такого качества сборки.
* Если качество сборки ухудшилось, то обязательно должны быть указаны регрессионные места.
* Наиболее нестабильные части функционала следует выделить и указать причину, по которой они таковыми являются.
* Даны рекомендации по тому функционалу и дефектам, скорейшее исправление которых является наиболее приоритетным.
* Список наиболее критичных для сборки дефектов, с указанием названия и их критичности.
* Для отчета уровня Smoke обязательно указать весь нестабильный функционал. Если сборка является релизной или предрелизной, то любое ухудшение качества является критичным и важно об этом сообщить менеджеру как можно раньше.

Помимо всего вышеуказанного для релизных и предрелизных сборок в отчете о качестве продукта важно указывать следующее:

* Дана информация о всех проблемах, характерных сборке. Проведен анализ, насколько оставшиеся проблемы являются критичными для конечного пользователя.
* Указаны дефекты, которые следует исправить, чтобы качество конечной сборки было выше.

Детализированная информация (Detailed Information)

В данной части отчета описывается более подробная информация о проверенных частях функционала, устанавливается качество каждой проверенной части функционала(модуля) в отдельности. В зависимости от типа проводимых тестов, эта часть отчета будет отличаться.

Smoke

При оценке качества функционала на уровне Smoke теста, оно может быть либо Приемлемым, либо Неприемлемым. Качество сборки зависит от нескольких факторов:

* Если это релизная или предрелизная сборка, то для выставления Приемлемого качества на уровне Smoke не должно быть найдено функциональных дефектов.
* Наличие нового функционала. Новый функционал, который впервые поставляется на тестирование, не должен содержать дефектов уровня Smoke для выставления Приемлемого качества всей сборки.
* Чтобы установить сборке Приемлемое качество, не должно быть дефектов уровня Smoke у того функционала, по которому планируется проводить полные тесты.
* Все наиболее важные части функционала отрабатывают корректно, тогда качество всего функционала на уровне Smoke может быть оценено, как Примлемое.

В части о детализированной информации качества сборки следует более подробно описать проблемы, которые были найдены во время теста.

DV

В этой части отчета указывается качество о проведении валидации дефектов.

Здесь должна быть следующая информация:

* Общее количество всех дефектов, поступивших на проверку.
* Количество неисправленных дефектов и их процент от общего количества.
* Список дефектов, которые не были проверены и причины, по которым этого не было сделано.
* Наглядная таблица с неисправленными дефектами.

По вышеуказанным результатам выставляется качество теста. Если процент неисправленных дефектов < 10%, то качество Приемлемое, если > 10%, то качество Неприемлемое.

NFT

При проведении полного теста нового функционала качество отдельно проверенного функционала может быть: Высокое, Среднее, Низкое.

В отчете следует отдельно указывать информацию о качестве каждой части нового функционала. В этой части отчета должна быть следующая информация:

* Дана общая оценка реализации нового функционала (сгруппированная по качеству).
* Подробная (детальная) информация о качестве каждой из частей новой функциональности.
* Проведен анализ каждой из новых функций в отдельности.
* Даны ясные пояснения о выставлении соответствующего качества.
* Даны рекомендации по улучшению качества (какие проблемы следует исправить).
* Показана таблица с новыми функциями (название), их качеством, статусом фуннкции из CQ.

AT, MAT, Regression

Если проводились тесты указанных уровней, то в первую очередь при написании отчета нужно анализировать динамику изменения качества проверенной функциональности в сравнении с более ранними версиями сборки. Также как и у предыдущего вида тестов, качество этих может быть: Высокое, Среднее, Низкое.

Для указанных видов тестов в данной части отчета должна быть описана информация следующего характера:

* Дана сравнительная характеристика каждой из частей функционала в сравнении с предыдущими версиями сборки.
* Подробная (детальная) информация о качестве каждой из частей проверенной функциональности.
* Даны ясные пояснения о выставлении соответствующего качества каждой функции в отдельности.
* Даны рекомендации по улучшению качества (какие проблемы следует исправить).

Окончание содержимого

В завершении содержимое отчета должно включать в себя информацию следующего характера:

* Ссылка на тест-план.
* Ссылка на документ feature matrix (если таковой имеется).
* Ссылка на документ со статистикой (если таковой имеется).
* Общее количество всех новых дефектов.
* Подпись высылающего отчет.

Данные ссылки должны быть корректными, необходимо проверить достоверную ли информацию получает пользователь, открывший ссылку. Следует обращать особое внимание на подпись, удостоверьтесь, что указана именно ваша подпись либо какая-то универсальная для определенного проекта подпись.

## GUI

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. Сравнить его с требованиями и/или с макетом и все. Или не все? А как насчет верстки?

**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.

Все? — Нет :) У верстки есть еще множество параметров и элементов, которые мы очень часто забываем проверить.

Сравнение с макетом — метод наложения готового эталонного макета (обычно psd-файл) на приложение в экране браузера, все несовпадения можно рассматривать как ошибки (для этого есть хороший инструмент [Pixel Perfect](http://www.welldonecode.com/perfectpixel/)).

Измерение размеров элемента — если это имеет значение, то померять размеры элемента и сравнить их со спецификацией можно с помощью, например [Page Ruler](https://blarg.co.uk/tools/page-ruler).

Правильность шрифтов (название, размер, цвет) — [WhatFont](https://chrome.google.com/webstore/detail/whatfont/jabopobgcpjmedljpbcaablpmlmfcogm).

Цвета интерфейса — [ColorZilla](http://www.colorzilla.com/chrome/screenshots.html).  
  
Контент — проверить на наличие орфографических и грамматических ошибок ([SpellChecker](https://chrome.google.com/webstore/detail/spell-checker-for-chrome/jfpdnkkdgghlpdgldicfgnnnkhdfhocg)).

Появление курсора — довольно часто мы забываем проверить, появляется ли вообще и как выглядит курсор в полях ввода, на кликабельных элементах.

Фавикон — такая маленькая незначительная вещица, но может изрядно подпортить впечатление пользователя (в моей практике были случаи, когда разработчики или дизайнеры шаблона оставляли фавикон с логотипом своей компании на сайте у заказчика).

Обозначение возможности переноса элементов.

Кодировка (UTF8...).

Стандарты HTML/CSS — достаточно неплохие решения для быстрой проверки предлагает [W3C](https://validator.w3.org/).

Заголовки по всему приложению должны быть приведены к одному стандарту ([пример](https://neilpatel.com/wp-content/uploads/2017/03/image32-2.png)).

Title страницы — о нем мы тоже часто забываем, также как и разработчики :)

Back button — достаточно часто встречается ошибка при переходе на какую-то страницу и нажатии на браузерную кнопку Back, предыдущая страница крашится или возврат на нее вовсе не осуществляется.

Масштабируемость — особенно это важно при тестировании на смартфонах и планшетах. Где пользователь часто меняет масштаб экрана ([Window Resizer](https://chrome.google.com/webstore/detail/window-resizer/kkelicaakdanhinjdeammmilcgefonfh)), а также режим адаптивного дизайна (например в FireFox Developer Edition).

Кроссбраузерность — одна и та же страница может выглядеть по-разному в разных браузерах ([пример](http://software-testing.ru/images/stories/library/layout-testing/layout-testing66.png)).

Проверяем Scroll.

Браузерные расширения, которые могут влиять на внешний вид приложения (например, AdBlock) — пробуем включить и отключить.

Проверить контент при отключенных (режим [WebDeveloper](https://chrome.google.com/webstore/detail/web-developer/bfbameneiokkgbdmiekhjnmfkcnldhhm)) изображениях, flash, JavaScript.

Все? — Нет :)

**Локализация** — что мы знаем об этом? Обычно наши знания сводятся к невнятным «ну, это язык», «кодировка», «раскладка», еще реже «геолокация». Что еще мы так часто забываем проверять в рамках тестирования локализации?

* Проверяем тестовый образец на правильность перевода — тут, конечно, хорошо бы подключить переводчика или носителя языка, но за неимением таких, берем тестовый образец и переводим через любой онлайн-переводчик (ну и все мы помним, как прекрасно и весело читать описание товаров на русском языке на AliExpress).
* Длина переведенных слов — количество символов в переведенном слове может быть гораздо больше ([пример](https://hsto.org/storage3/d1b/f24/090/d1bf24090668afe4cbf3024e9953419f.png)), что может привести к «расползанию» интерфейса при переводе.
* Сокращения/аббревиатуры — существуют правила, по которым их либо переводят, либо транслитерируют, либо оставляют как есть.
* Валюта.
* Параметры шрифта могут также значительно отличаться в зависимости от языка ввода.
* Проверить работу поиска во всех локализациях — к примеру, на AliExpress результаты поиска одного и того же слова «смартфон» дают разный результат по количеству найденных товаров, причем разница исчисляется десятками тысяч.
* Мета-информация (keywords/title/description) — столь незначительное для пользователя, невидимое, но такое важное для поисковых машин и продвижения сайта в гугле и других поисковиках.
* RTL (right to left languages) — языки c обратным написанием (арабский, иврит) имеют свои особенности: числа пишутся слева направо, значки и иконки отзеркаливаются, названия программ не переводятся, нет переносов, кнопки редактирования Backspace и Delete работают наоборот.

## Functional

От внешнего переходим к внутреннему — функциональному тестированию. Если в тестировании GUI мы проверяли наличие и внешний вид элементов, то в функциональном тестировании мы проверяем их **работоспособность и взаимодействие**.

Определить основные функции предмета или приложения достаточно просто — нужно понимать его назначение. Задайте себе вопрос — а для чего нужен карандаш? Занавеска? Интернет-магазин? Для чего нам на сайте нужна форма логина? Для чего нам кнопка «Купить»? И тогда все функции приложения открываются как на ладони.

Самый простой способ подготовиться к функциональному тестированию — это выписать список элементов вашего приложения и написать их целевое назначение («зачем?»).

Например:

|  |  |  |
| --- | --- | --- |
| Кнопка | Зачем? | После нажатия происходит какое-то действие. |
| Поле ввода | Для передачи какой-то информации и взаимодействия с приложением. |
| Поиск | Для того, чтобы пользователь мог быстро найти релевантную информацию. |
| Логин-форма | Чтобы пользователи могли иметь доступ к определенным функциям приложения (или наоборот, ограничить их доступ). |
| Календарь | Например, для выбора дат (билеты, бронирование и т. п.). |
| Дата и время | Например, расписание прибытия транспорта. |
| Сообщения об ошибках | Чтобы сообщить пользователю о том, что приложение работает некорректно, либо он делает некорректные действия. |
| Всплывающие окна и подсказки | Направить пользователя по нужному сценарию. |

У вас уже почти готов список тестовых сценариев. Зная целевое назначение любого элемента, мы можем легко описать все позитивные и негативные сценарии, необходимые для тестирования этого элемента.

Но и тут мы можем кое-что забыть. Часто забываемые проверки функциональных элементов приложения:

Кнопки:

* Enter должна срабатывать как submit;
* Tab должен переводить курсор на следующий элемент.

Поля ввода:

* trimming («убирание») пробелов в полях ввода;
* пустота/пробелы в поле ввода;
* все способы редактирования (Insert, Delete, Backspace, Ctrl+C/V/X/Z и т. д.);
* дроби ( 1.5 | 1,5 | ⅕).

Поиск:

* wildcard symbols (\* | ?);
* написание поискового запроса слитно | раздельно | через дефис должно вести к одному результату;
* ввод текста в другой раскладке.

Сообщения об ошибках:

* пробуем отключить в настройках браузера.

Календарь:

* 31 июня;
* 29 февраля + не высокосный год;
* прошлое/будущее (например, купить билет на уже прошедшее число).

Время:

* синхронизация с сервером (на сервере приложения может быть выставлено другое время, отличающееся от таймзоны пользователя);
* временные зоны.

E-mail:

* логин (63 символа) @ домен (253 символа (может быть ip)).

Всплывающие окна / подсказки:

* пробуем закрыть разными способами (нажатие на кнопку (если есть), на «крестик», клавишей ESC, просто нажатием в другую область экрана);
* рефреш страницы особенно в момент запроса на сервер (например, совершение транзакции по покупке) иногда может приводить к появлению ошибок.

## Usability

За внешним видом и функциональностью следует удобство (Usability). Не менее важная часть, так как от нее зависит, будет ли востребован ваш продукт вообще. О каких моментах нужно помнить при тестировании usability веб-приложения?

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

## Security

Тестирование безопасности:

* Начинаем всегда с составления [матрицы уровней доступа;](http://quality-lab.ru/wp-content/uploads/2017/04/matrica.png)
* 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.

## Performance

Производительность:

* Имитируем нагрузку пользователями ([JMeter](http://jmeter.apache.org/));
* Пробуем загрузить большие объемы данных, файлы, медиа;
* Нагружаем БД;
* Понижаем скорость инета ([NetLimiter](https://www.netlimiter.com/));
* Понижаем скорость передачи данных (Throttling);
* Тестируем восстановление системы после падений.

## Configuration

Конфигурационное тестирование. Тут все тоже просто:

* Берем у разработчиков/заказчика список софта и железа, на котором и с которым должно работать наше приложение.
* Думаем над тем, с чем еще взаимодействует приложение (например соцсети, почта, возможно, камера на телефоне и т. п.).
* 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.

## Памятка

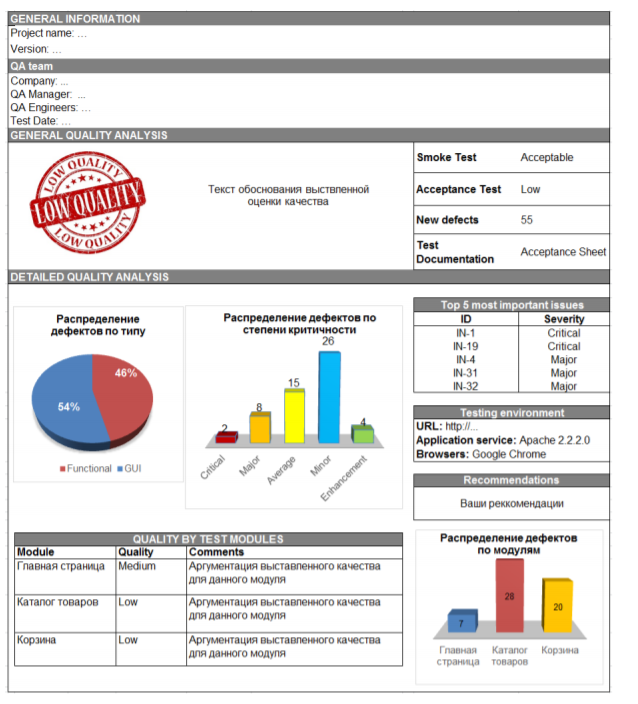
В завершение хочу поделиться с вами базовой памяткой по тестированию веб-приложений, которую вы можете взять за основу и дополнять.

|  |  |
| --- | --- |
| GUI | * макет * контент * кодировка * элементы (цвет, размер, расположение) * локализация * стандарты HTML/CSS * масштабируемость * курсор * заголовки * шрифты * фавикон * scroll * кроссбраузерность |
| Functional | * работа кнопок * имейл * регистрация/авторизация * поля ввода * время и дата * сообщения об ошибках * поиск * всплывающие окна/подсказки * формы заполнения * календари * взаимодействие всех модулей системы |
| Usability | * навигация * соответствие целям приложения * печать * логичность * локализация * help * информативность * совместимость с другими приложениями * ожидания конечного пользователя * скорость работы |
| Security | * матрица уровней доступа * протоколы передачи данных * конфиденциальность информации * протоколы криптования * доступность информации * авторизация |
| Performance | * нагрузка * имитация количества пользователей * БД нагрузка * стабильность * «тяжелый» медиа-контент * скорость выполнения запросов к БД * стресс * скорость интернета * корректные сообщения об ошибках * восстановление * объем загружаемых файлов * восстановление данных / системы |
| Configuration | * сторонний софт * «железо» * совместимость с другими браузерами * OS |

**ЗАДАНИЕ К ПРАКТИЧЕСКОЙ РАБОТЕ**

1. Изучить статью
2. \*Запустить ранее созданное приложение.
3. \*Составить итоговый отчет по результатам тестирования приложения.

Образец:



1. Оформить презентацию по представленному материалу
2. Ответить на контрольные вопросы

1. Какая структура итогового отчета о результатах тестирования?

Приветствие.

 Общая информация (Common Information).

 Тестовое окружение (Test Platform).

 Рекомендации QA (QA Recommendations).

 Детализированная информация (Detailed Information).

Окончание содержимого.

2. Что содержится в разделе Приветствие?

Свое письмо с отчетом необходимо начать с приветствия всех адресатов. Также в самом начале письма следует указывать, если были какие-то внешние факторы, препятствующие проверке какой-то части функционала.

3. Что содержится в разделе Общая информация?

В данной части отчета описывается, какие виды тестов проводились.

4. Что содержится в разделе Тестовое окружение?

Как правило, в этой части указываются:

Название проекта.

Номер сборки.

Ссылка на проект (сборку).

5. Что содержится в разделе Рекомендации QA?

Здесь отражается общее состояние сборки. Здесь показывается аналитическая работа тестировщика, его рекомендации по улучшению функционала, наиболее слабые места и наиболее критичные дефекты, динамика изменения качества проекта.

6. Что содержится в разделе Детализированная информация?

В данной части отчета описывается более подробная информация о проверенных частях функционала, устанавливается качество каждой проверенной части функционала(модуля) в отдельности.

7. Что содержится в разделе Окончание содержимого?

В завершении содержимое отчета должно включать в себя информацию следующего характера:

 Ссылка на тест-план.

 Ссылка на документ feature matrix (если таковой имеется).

 Ссылка на документ со статистикой (если таковой имеется).

 Общее количество всех новых дефектов.

 Подпись высылающего отчет.

Вывод: Мысоставили итоговый отчет о результатах тестирования

приложения.