15.08.2016

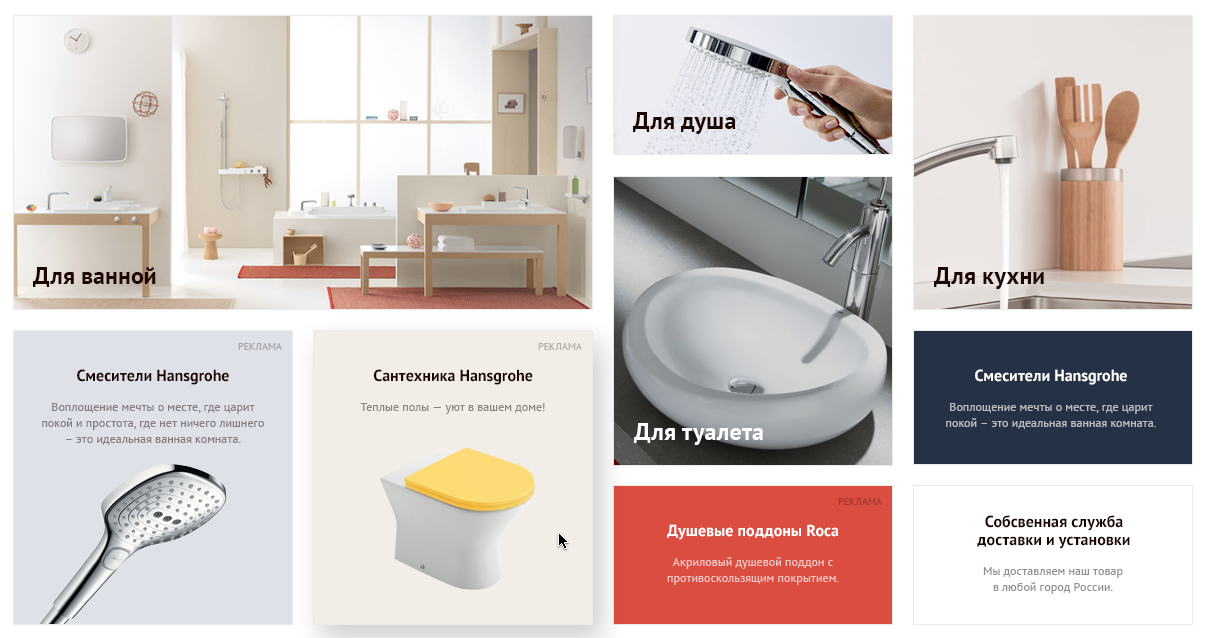
PIXOMATICA / Игорь

Тут обсуждаем верстку.

15.08.2016

Михаил

Промо-плитка



Рекламно-навигационный блок.

Каждый элемент “плитки” является изображением и является ссылкой в том же окне.

Ссылки могут вести:

* На категорию;
* На карточку товара;
* На страницу “ситуативки”;
* На сервисную страницу.

Блок включает скрытые элементы, отображающиеся по таймауту сменяя видимые.

Динамически сменяемыми являются только рекламные блоки.

Описание динамики:

* Одномоментно сменяется только 1 элемент.
* Смена первого элемента происходит с таймаутом 4000ms от загрузки страницы.
* Все последующие элементы сменяются с таймаутом 4000ms от предыдущей смены элемента.
* Эффект смены элементов: flip down, flip right. Время анимации 500ms. Выбор типа анимации- вручную при загрузке.

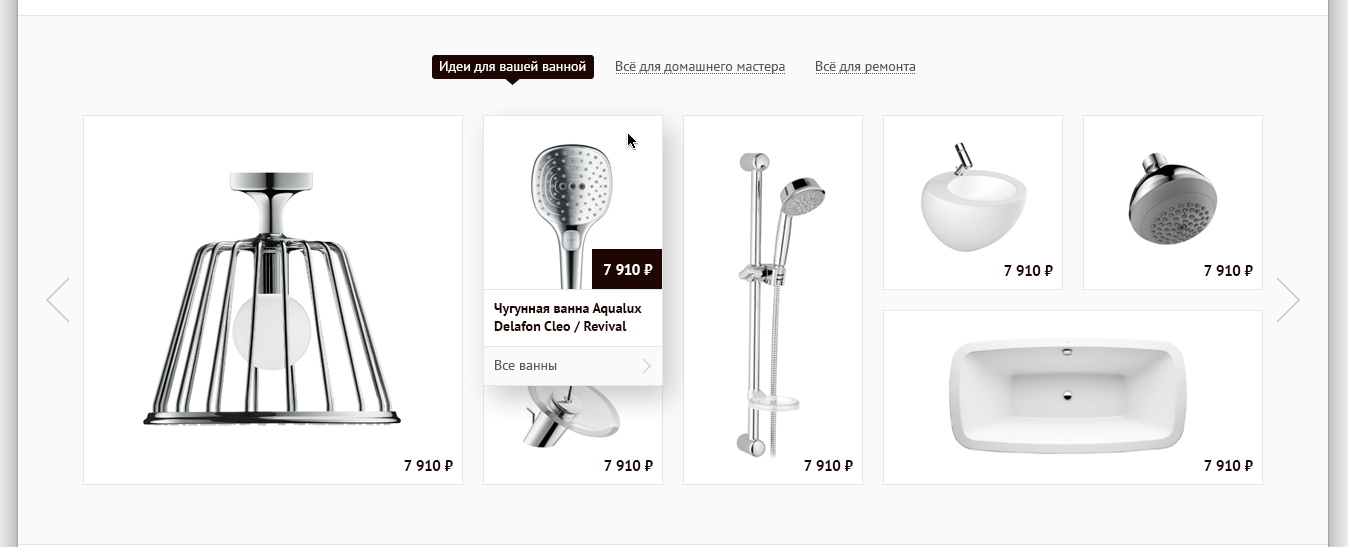
Пример реализации:

<https://media.giphy.com/media/3oEjHVfozNaRMWLaTu/giphy.gif>

Элементы имеют визуальные состояния:

* базовое;
* по наведению;

Блок “Товарная плитка”



Блок представляет собой слайдер без автопрокрутки.

Слайдер:

Слайдер включает: а) навигацию, б) слайды.

Слайды замкнуты в кольцо *(1-2-3-1-2-3-1)*

Навигация включает:

* Контролы влево/вправо;
* Текстовые якори над слайдером.

Контролы влево/вправо имеют визуальные состояния:

* базовое;
* по наведению;
* нажатию (mouse down).

Текстовые якори имеют визуальные состояния:

* базовое;
* по наведению;
* активное.

Слайд включает 1 набор анонсов товаров

**Типы наборов: общий, плиточный**

Каждый общий набор формируется вручную и включает:

* Promo наименование набора;
* Индивидуальную расстановку элементов в рамках общей сетки 6х2;
* Элементы набора

Каждый элемент общего набора слайда включает:

* Ссылку на товар (в том же окне);
* Ссылку на категорию (в том же окне).

Элементы (анонсы товаров) общего набора имеют визуальные состояния

* базовое;
* по наведению;

Ссылка на категорию (в рамках элемента) имеет визуальные состояния

* базовое;
* по наведению;

Каждый плиточный набор формируется вручную и включает:

* Promo наименование набора;
* Индивидуальную расстановку элементов коллекции плитки в рамках общей сетки 6х2;
* Ссылку на карточку коллекции (в том же окне);

Блок “Популярные производители”

Добавляем листалку “влево-справо”, обновляет все элементы сразу.

22.09.2016

PIXOMATICA / Игорь

Ответ на вопросы и предложения Павла в скайпе от 21.09.2016:

1. Рассмотреть возможность интеграции с одним из популярных css фреймворков, например Bootstrap, от туда можно забирать: гриды, дропдауны, тултипы, и тд. Фреймворк позволяет гибко настраивать параметры, определять состав компонентов

Bootstrap сам по себе сочетается с БЭМ как таковым никак, а с именованием классов по БЭМ просто очень плохо: использование классов из бутстрапа будет добавлять зависимость элементам от него. На мой взгляд в принципе желательно не использовать bootstrap нигде кроме прототипирования. Если же интеграция нужна, то можно сделать, но по крайней мере на главной странице нужды в бутстрапе не было никакой, и его использование наоборот усложнило бы ситуацию.

Оцениваемые трудозатраты: 0,5 дня, но целесообразность этого лично у нас под тремя вопросами (может, мы чего-то не знаем из жизни проекта).

2. Для иконок лучше собирать коллекцию из векторных шрифтов, например можно собрать свою коллекцию на сервисе http://fontello.com/

Сейчас векторные иконки собираются в спрайты при сборке проекта, что в свою очередь позволяет настроить их сборку как угодно, например, группируя по блокам, страницам, позволяет задавать им соответствующие префиксы. Чем лучше вручную их объединять в шрифт?

Оцениваемые трудозатраты: 0,5-2 дня, но целесообразность этого лично у нас под тремя вопросами (может, мы чего-то не знаем из жизни проекта). Понадобятся уточнения по требованиям.

3. js должен быть объектно ориентированным и разделён на слои:

JS не организовывался никаким образом из-за того что миллион вопросов в отсутствии ТЗ. Неизвестно каким образом будет внедряться данная страница, в уже существующую систему или в новую создаваемую систему? Будет ли там уже иметься подход к организации скриптов и самого js-кода? Есть там принятая архитектура фронта? Есть ли там процесс билда ресурсов? Билда блоков по БЭМ? Насколько там используется БЭМ, только для именования блоков или прямо весь стек технологий? Или только планируется его использование?

Ничего из этого не было известно, поэтому js сейчас организован наиболее прозрачным и простым способом, позволяющим его понять и как угодно его переделать и организовать в соответствии с потребностями (напоминаю, что главную мы делали в качестве теста).

3.1. Сторонние библиотеки, например jquery. Такие библиотеки должны лежать в разделе со статикой, а не со скриптами, так как некоторые из них содержат стили и картинки, а может быть даже шрифты, как пример font-awesome. Управлять набором библиотек и их версий требуется пакетным менеджером [bower](http://bower.io)

Введение bower для библиотек разумеется повлияет на структуру хранения подключаемых файлов в любом случае.

Оцениваемые трудозатраты: 1 день.

3.2. Уровень библиотек проекта, например классы корзин, оформления заказа,

веб-форм. Такие библиотеки должны описываться js классами и лежать в отдельной папочке, почитать об этом можно в документации к БЭМ (https://ru.bem.info/methodology/filesystem/). Данные классы должны быть модулями в Ecmascript 2015 стиле или как модули AMD.

(см. ниже)

3.3. Уровень БЭМ компонентов. Тут выполняется бизнес логика компонента, требуемые библиотеки подключаются асинхронно как модули Ecmascript 2015 или AMD

(см. ниже)

4. Сборщик статики gulp, для модульной сборки рекомендуется использовать webpack или r.js, оба имеют плагины под gulp, Про сборку зависимостей можно почитать тут https://ru.bem.info/methodology/build/ Стремиться к тому чтобы большая часть сборки подтягивалась как зависимости модулей.

(см. ниже)

5. На фронтенде соответственно подключается модульный загрузчик, который будет первым делом подключать собранный бандл и уметь подключать не вошедшие в бандл зависимост. Это или SystemJS или RequireJS

Имеет ли смысл организовывать имеющийся js не имея представления о том какой бэк будет на сайте, за что будет отвечать он, а за что фронт, и что будет требоваться от фронта помимо jquery-лапши независимо от того организована она в amd модули или нет? :-)

Миллион вопросов должны быть определены и решены до и для реализации реальной архитектуры фронта на сайте. Ничего этого разумеется сейчас не делалось и имеющийся js сейчас это не более чем очень примерный пример реализации части требуемого от главной страницы функционала. Крутить процесс билда и организации js-кода туда-сюда без приложения к реальному сайту с бэком и всем остальным выглядит пустой тратой времени ни на что.

Тем не менее! Оцениваемые трудозатраты: навскидку — 5-7 рабочих дней.

**Резюмируя.** Для того, чтобы все со всеми договорились, нужно хоть какое-то техзадание, согласованное на вашей стороне. И мы по нему будем жить. Мы просили его в начале, но из этого ничего не получилось, поэтому мы изначально верстали главную, чтобы пройти тестовое задание (об этом мы говорили на встрече в мае). Сейчас, когда мы стоим на пороге верстки каталога и остальных страниц, вопрос ТЗ становится как никогда актуальным.

**Протокол собрания по вёрстке от 27.09.2016г.**

На встрече рассматривались вопросы технической архитектуры выполняемой вёрстки.

В результате обсуждения были достигнуты следующие договоренности:

Поддержка браузерами

Opera 12.17+

Ie8+

Safari 7+

Chrome 44+

Yandex 16+

Firefox 39+

IE8-9 в режиме деградации, должен работать основной функционал сайта, grid колонки.

Старт интеграции с сайтом

Старт интеграции с сайтом планируется после утверждения вёрстки главной страницы.

Иконки

Иконки сайта рисуются в в svg формате и сборщиком проекта собираются в спрайты.

Использование компонентов css фреймворка Bootstrap

В вёрстке используются компоненты bootstrap по мере надобности. В первую очередь это grid система, а также js плагины дропдаунов, табов. Подключается bootstrap sass через bower, настраиваются параметры цветов, размеров. Изменения конфигурации не должны модифицировать подключенную библиотеку - должна сохраниться возможность обновить bootstrap до новой версии пакетным менеджером. Если так не получится сделать договорились собирать bootstrap через конструктор на сайте и заливать в проект уже готовую сборку.

В разговоре не была достигнута договоренность по версии bootstrap, так как не были сформулированы окончательные требования к версиям поддерживаемых браузеров. Вообще B4 официально не поддерживает ie8, но его есть возможность подключить с полифилами, если проблем с этим не возникнет, то я думаю лучше делать сразу на 4м, так как поддержка 3го скоро прекратится. <http://v4-alpha.getbootstrap.com/getting-started/browsers-devices/#supporting-internet-explorer-8>

Модульная(AMD) объектно-ориентированная архитектура вёрстки

Логика приложения пишется на js последней версии, языки с синтаксическим сахаром coffeescript или typescript не используются.

На этапе сборки проекта запускается транспайлер(Babel) переписывающий код в предыдущий стандарт ES5.

На сайт подключается модульный загрузчик SystemJS который который управляет асинхронным подключением зависимостей (AMD) реализованных в стандарте ES2015, на этапе сборки зависимости собираются в бандл(-ы) с помощью сборщика webpack, лучше создавать отдельные бандлы для разнотипных разделов.

Код пишется в объектно-ориентированном стиле и разделяется на слои:

- Слой уровня представления - является бэм компонентом, js должен быть тонким, отвечает за инициализацию и передачу управления уровням ниже

- Слой прикладных операций - на этом уровне определяются задачи которые должна решать данная задача и распределяет их между объектами, выражающими суть модели. Своего рода менеджер-управляющий который запускает, контролирует связывает объекты и сервисы. Этот уровень не нужно раздувать сильно. В нём не содержатся ни знания, ни бизнес правила, а только выполняется координирование задач и распределение работы между совокупностями объектов модели. Тут как и наши могут находиться сторонние библиотеки, например слайдер, модальное окно.

- Слой модели - на этом уровне описываются сущности объектами и сервисами приложения. Например объект корзины, фильтр или элемент фильтра типа диапазон значений.

- Слой инфраструктуры - классы отвечающие за обеспечение непосредственной технической поддержки слоёв выше, например класс работы с json форматами. Часто это сторонние библиотеки.

Сборка всех этих слоёв осуществляется модульным асинхронным загрузчиком.

Важно понимать что не на все задачи нужны все слои, например для подключения слайдера достаточно уровней представления и прикладных операций, как правило это плагин стороннего изготовителя, где разделение его реализации на слои нас не касается. В этом примере в БЕМ компоненте подключается плагин слайдера как зависимость и происходит его инициализация.

Пример который я приводил с перемещением блоков в зависимости от типа ширины экрана состоит из представления - отвечает за создание объекта класса для конкретного бем компонента, слоя прикладных операций - отвечает за подписку на событие и перемещение блока если есть где-то под него контейнер, слоя инфраструктуры - класс генерирующий события xsmall-large в зависимости от on-resize.

Для того же фильтра скорее всего понадобится уже полный набор слоёв.

Сторонние библиотеки, например jquery. Такие библиотеки должны лежать в разделе со статикой, а не со скриптами, так как некоторые из них содержат стили и картинки, а может быть даже шрифты, как пример font-awesome. Управлять набором библиотек и их версий требуется пакетным менеджером [bower](http://bower.io).

Если для задачи требуется сложный html шаблон, его можно положить в скрытый блок например с классом sample-item, а ему или родителю через data-attrs передать дамп данных или ajax-ом догрузить, если требуется дозагрузка после инициализации страницы. на этапе инициализации объекта слоя модели этот sample сделать detach из dom дерева и использовать его как прототип для создания объектов. Чтобы не было мерцания на загрузке страницы, например большой корзины, можно положить и sample и на бекенде уже собрать html с элементам. Тогда на стадии инициализации нужно это учитывать и по какому-то атрибуту dom элемента связать его с объектами модели данных построенными описанном ранее образом через data-attrs.

Обсуждалось также использование Vue.js для дата-биндинга этих сложных html шаблонов, тут важно предусмотреть что разметка {{}} не должна быть видна на этапе загрузки страницы.