Эта статья будет интересна тем, кто хочет на самом базовом уровне познакомится с препроцессором LESS и некоторыми подходами к адаптации web-страниц для мобильных устройств. Форма материла - своего рода сборник самых ходовых рецептов, использованных при реализации проекта адаптации портала крупного ритейлера для мобильных устройств. Дизайн портала изначально был сделан с расчетом на мобильные устройства. Почти все блоки на странице реализованы в концепции виджетов. Как правило, по горизонтали расположено четыре виджета. Соответственно логично было бы использовать для планшетной версии портала два виджета по горизонтали, а для телефонной – один.

LESS некоторые плюшки

В данном проекте, был выбран препроцессор LESS. Использование препроцессора в проекте имело множество причин. Из основных:

- Была необходимость использовать повторно ряд селекторов, не загромождая при этом код бесконечным числом повторений.

Например:

```
body, html {
    &.phone  {
        font-size: 5vw;
    }
    &.tablet {
        font-size: 2vw;
    }
}
```

Если перевести в обычный CSS это будет выглядеть вот так:

```
body.phone, html.phone {
    font-size: 5vw;
}

body.tablet, html.tablet {
    font-size: 2vw;
}
```

Соответственно & это родительский селектор, вместо которого подставляется цепочка селекторов уровнем выше. В данном случае это body и html

- Структурировать дизайн в единый фреймворк, так чтобы не было необходимости дублировать css код элементо, в оформленных схожим образом.

- Иметь возможность при необходимости или требованию заказчика быстро изменить дизайн отдельных групп элементов.

Все группы оформления можно разбивать на отдельные файлы и подключать их по мере необходимости. Так же это упрощает внесение изменений.

Например, для оформления текста использован отдельный модуль typography.less, который содержит основные принципы типографики, использованные для мобильных версий сайта.

Небольшой пример из этого файла:

```
@font-scale-factor: 1.09;

.font-size(@size: 0) {
    font-size: pow(@font-scale-factor, @size)*1rem;
    line-height: pow(@font-scale-factor, @size+1)*1rem;
}

.line-height(@size: 0) {
    line-height: pow(@font-scale-factor, @size)*1rem;
}

.line-height-block(@size: 0, @lines: 1) {
    height: pow(@font-scale-factor, @size)*1rem*@lines;
}
```

таким образом в LESS задаются константы

```
@font-scale-factor: 1.09;
```

таким образом задаются примеси с параметрами (Parametric Mixins) и используются константы

```
.line-height-block(@size: 0, @lines: 1) {
    height: pow(@font-scale-factor, @size)*1rem*@lines;
}
```

Примеси по сути функция, на входе которой параметры, а на выходе css код, в который преобразуется эта примесь.

В данном миксине задается размер текста и количество строк, а на выходе получается высота блока в пикселах для нужного количества строк. Это очень удобно, если нужно обрезать текст новости внутри блока так, чтобы строки вне границ блока были не видны.

Миксины выполняют такую же роль, как классы, которые используются для оформления элемента, как например, это реализовано в bootstrap. Только вместо того, чтобы семантически захламлять HTML код кучей классов, которые объединяют элементы в бессмысленные группы (например, ширина колонки, или отступ колонки), используется препроцессор, который повторяет один и тот же код в файле стилей.

Размер HTML+CSS файлов при использовании препроцессоров в итоге становится больше, но при использовании gzip сжатии на веб сервере, это не играет такого большого значения, так как все избыточные повторы эффективно сжимаются.

Так как портал ритейлера базируется на платфрме Sharepoint, то очень удобно именно добавлять миксины в файлы стилей, нежели делать правки в коде серверной разметки, добавляя новые классы. Второй вариант иногда требует использования механизма установки пакетов для некоторых решений, что довольно неудобно.

Типографика

Переменная @font-scale-factor задана для того, чтобы можно было менять соотношение между размерами заголовков и основного текста. С помощью этой переменной задается линейка размеров текста:

1/(1.09*1.09), 1/1.09 1, 1*1.09, 1*1.09*1.09

то есть:

0.841, 0.917, 1, 1.09, 1.188. и т.д.

Соответственно, меняя переменную, можно добиться того, что соотношение размеров заголовков h1 h2 h3 и т.д. будет отличаться не так сильно друг от друга по величине, как по умолчанию установлено в файле стилей самого браузера.

Такое решение довольно удобно для мобильных устройств - заголовки меньшего размера лучше помещаются и читаются на маленьком экране. Размер заголовков задается через миксины. В данном случае это заголовки блоков новостей, видео и т.д.

```
.modul-h1() {
    .font-size(3);
    .line-height(4);
}

.modul-h2() {
  .font-size(2);
  .line-height(3);
}

.modul-h3() {
  .font-size(1);
```

```
  .line-height(2);
}
```

Так как мобильная версия добавлялась уже после того, как был реализован портал для стационарных устройств, довольно часто использовались миксины такого вида:

```
.modul-h2-p-t() {
    .phone &, .tablet & {
        .modul-h2();
    }
}
```

Данный миксин будет применятся только при наличии классов phone или tablet (они заданы для элемента body). Это довольно удобно, так как нет необходимости добавлять такую конструкцию только для того, чтобы добавить оформление заголовка.

```
.some-thing {
    .phone &, .tablet & {
        .modul-h2();
    }
}

.some-thing {
     .modul-h2-p-t();
}
```

Цветовые константы

Константы удобно использовать для того, чтобы выделить цветовое оформление элемента в отдельную сущность.

```
@table-color-dark-grey: #e5e5e5;
@table-color-grey: #f4f4f4;
@table-color-white: #fff;
```

В обычном css может быть куча мест, где используется цвет #fff (белый). И нет возможности без особых сложностей поменять цвет только для таблиц, сделав поиск цвета по файлу стилей, особенно если мы имеем несколько таблиц разных видов.

Корпоративные цвета также заданы константами, что упрощает использование этих цветов, если над проектом работает несколько человек. Нет необходимости снимать этот цвет из

дизайна или искать, где он использовался в прошлый раз. Вдобавок по названию сразу понятно, что цвет был использован именно тот, который подразумевался.

```
/* Цвета */
@x5-dark: #1e2129;
@x5-orange: #ff4900;
@x5-white: #fff;
@pe-darkest-green: #03362d;
@pe-dark-green: #076821;
@pe-green: #67b200;
@pe-white: #fff;
@ka-green: #4d9d34;
@ka-yellow: #f8a901;
@ka-white: #fff;
@py-red: #e71802;
@py-green: #28a62a;
@py-white: #fff;
```

Вот так задаётся фон подложек в зависимости от бренда (у каждого бренда в body задан уникальный класс):

```
.main-colors-bg() {
    background-color: @x5-orange;

    .perekrestok-page & {
        background-color: @pe-green;
    }

    .karusel-page & {
        background-color: @ka-yellow;
    }

    .pyaterochka-page  & {
        background-color: @py-red;
    }
}
```

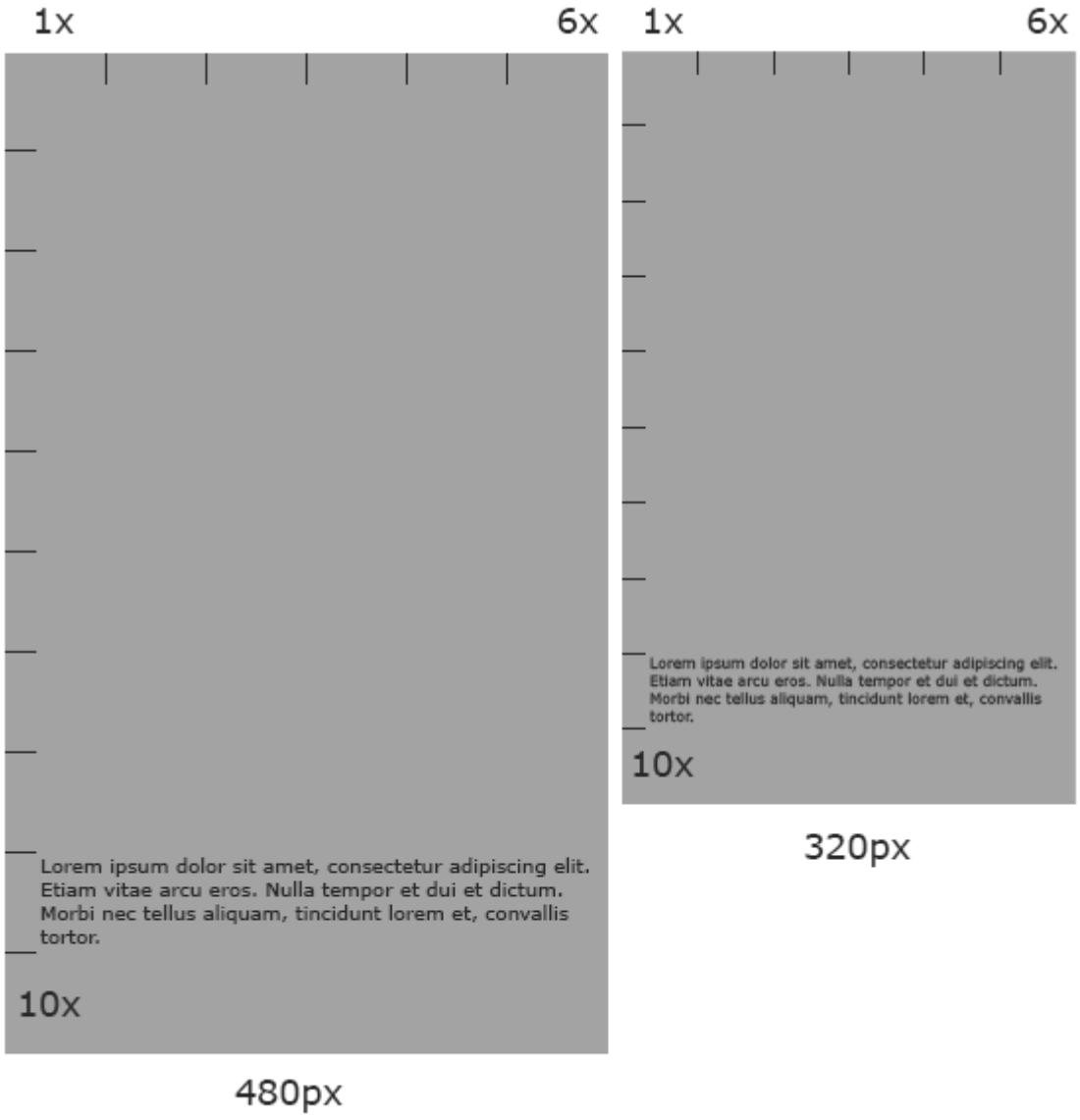
Реализация адаптации. Некоторые принципы.

Для дизайна мобильной версии была использована квадратная модульная сетка. Экран по горизонтали делится на 6 частей, таким образом получается ширина одной ячейки модульной сетки.

Все остальные размеры привязываются к ширине этой ячейки. Такая система была выбрана для того, чтобы можно было масштабировать все элементы под разную ширину экрана, при этом сохраняя пропорции элементов.

Например:

Есть блок новостей, по ширине 6 х размер модуля, и по высоте 10 х размер модуля. Соотношение сторон 6 к 10 будет сохраняться независимо от ширины экрана.



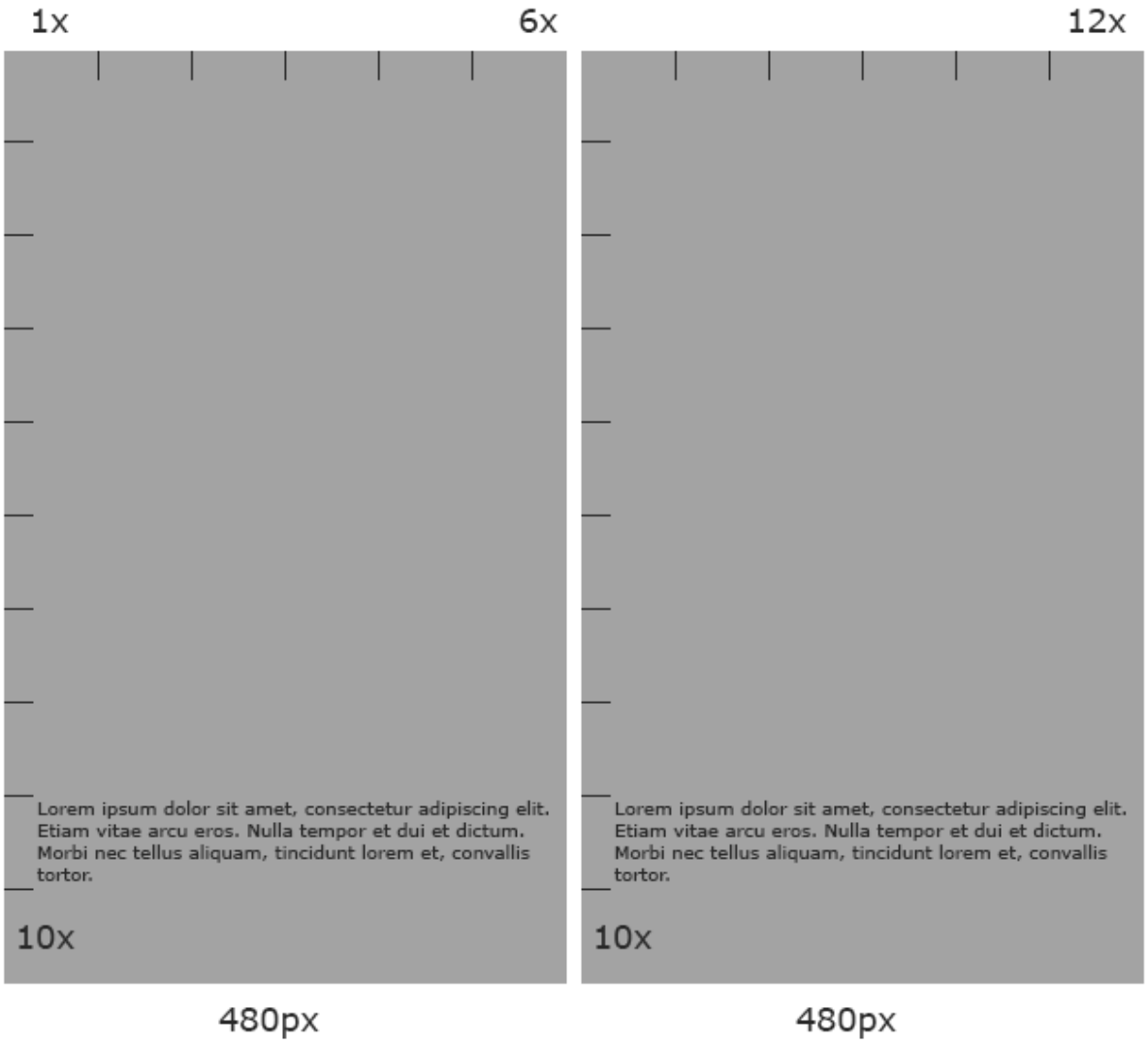
Примеры даны для ширины экрана в 480 пикселов и 320 пикселов.

С текстом аналогичная ситуация, его величина зависит от ширины экрана. Поэтому конечный размер блока текста будет занимать один и тот же процент площади, не зависимо от ширины экрана.

Для планшета также использована квадратная модульная сетка. Ширина экрана планшета делится на 12 частей, вместо 6, что позволяет на планшете с величиной экрана практически в два раза большей чем у телефона, отображать все элементы в два раза меньше (1/12 ширины экрана меньше 1/6 в 2 раза).

Например:

на планшете новости отображаются в 2 колонки вместо одной. Физически размер одной колонки на экране планшета примерно такой же как одной колонки на телефоне.



Деление экрана планшета на 12 частей. Одна часть на планшете в два раза меньше чем на телефоне. Физически и в пикселах одна часть на планшете практически равна такой же на телефоне.

Размеры одного элемента (1x) заданы как константы, так же как производные размеры.

```
/* Размер одного юнита квадратной модульной сетки */
@u: 100 / 6vw;
@u0125: @u * .125;
@u025: @u * .25;
@u05: @u * .5;
@u075: @u * .75;

@uu: 100 / 12vw;
@uu0125: @uu * .125;
@uu025: @uu * .25;
@uu05: @uu * .5;
@uu075: @uu * .75;
```

В данном случае @uu размер элемента модульной сетки для планшета (и сокращение от unit).

Унифицированные внутренние отступы для сайта и для блоков с контентом, заданы в юнитах.

```
@site-padding-phone: @u025;
@site-padding-tablet: @uu025;

@module-padding-phone: @u025;
@module-padding-tablet: @uu025;
```