УДК 004

Использование технологии flexbox для адаптивной верстки

Круглик Роман Игоревич
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
В статье рассматривается технология flexbox. Приведены несколько основных свойств для адаптивной вёрстки с примерами кода.
Ключевые слова: Адаптивная вёрстка, CSS, flexbox.

Use flexbox technology for responsive page-proofs

Kruglik Roman Igorevich
Sholom-Aleichem Priamursky State University
Student

Abstract
In article considered technology flexbox. Several basic properties for adaptive page-proofs with code examples are given.
Keywords: Adaptive layout, CSS, flexbox.

Современный front-end разработчик активно должен уметь применять на практике различные инструменты, позволяющие автоматизировать процесс верстки макетов и программирования клиентской составляющей проекта. Для этого уже существует множество фреймворков, как больших, так и малых, системы сборки, пакетные менеджеры, целая куча пакетов для задач любого уровня, препроцессоры, шаблонизаторы, которые созданы упростить и повысить производительность работы специалиста в данной области.

Новичкам бывает сложно разобраться в изобилии инструментов для вёрстки, так как тут нужно постоянно развиваться и знать основы. Многие по этой причине продолжают верстать «по-дедовски». Если раньше верстали на таблицах, а потом перешли на дивы, то сейчас flexbox обширно используется в сочетании с сетками или же готовыми css-фреймворкомами.

Исследования в области различных инструментов для ускоренной адаптивной верстки не заканчиваются по сей день. В статье Р.Р. Биктимиров, А.Б. Джемалетдинов [1] предлагают варианты использования HTML и CSS для front-end разработки при конструировании веб-сайтов, также описаны требования к использованию инструментов, которые помогут, в первую очередь, обеспечить конкурентоспособность front-end разработки веб-сайта. Т.Н. Филимоненкова, А.С. Дунаевский [2] анализируют лучшие подходы к написанию адаптивного сайта для разных разрешений и экранов, а также

самые популярные средства для ее реализации. Изложены возможности сервисов и преимущества каждого из них. А.С. Фадеева [3] рассматривает стратегии адаптации сайта к мобильным устройствам, выявлены их преимущества и недостатки. В статье Ю.С. Дмитриева [4] обоснована необходимость использования адаптивных веб-интерфейсов на основании статистики количества пользователей, пользующихся сетью Интернет с мобильных устройств (более 50%). Цель - создание методики, позволяющей привести неадаптированный веб-сайт к виду адаптированного, при этом улучшив его показатели скорости загрузки. Проведен анализ нескольких подходов к разработке адаптивных сайтов, объяснен отказ от использования отдельной мобильной версии в качестве альтернативы адаптивному дизайну.

Целью исследования является анализ возможностей инструмента для быстрой адаптивной вёрстки.

Одна из преимуществ технологии flexbox это редактирование только css кода для позиционирования элементов по всей странице. Структура html документа будет выглядеть так (см. рис. 1).

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <link rel="stylesheet" href="../app/style.css">
</head>
<body>
<div class="flex-container">
    <div class="flex-element">1</div>
    <div class="flex-element">2</div>
    <div class="flex-element">3</div>
    <div class="flex-element">4</div>
    <div class="flex-element">5</div>
</div>
</body>
</html>
```

Рисунок 1. Html документ

И CSS стили (см. рис. 2).

```
html
{
    background-color: #efeeff;
}
.flex-element
{
    display: block;
    background-color: #c6c9ff;
    margin: 0;
    padding: 10px;
    border: 1px black solid;
}
.flex-container
{
    margin-top: 10%;
    background-color: white;
    padding: 10px;
}
```

Рисунок 2. CSS

Сейчас имеется контейнер, в котором находятся 5 блочных элементов, выстроенных в том порядке, в котором записаны в коде. (см. рис. 3).

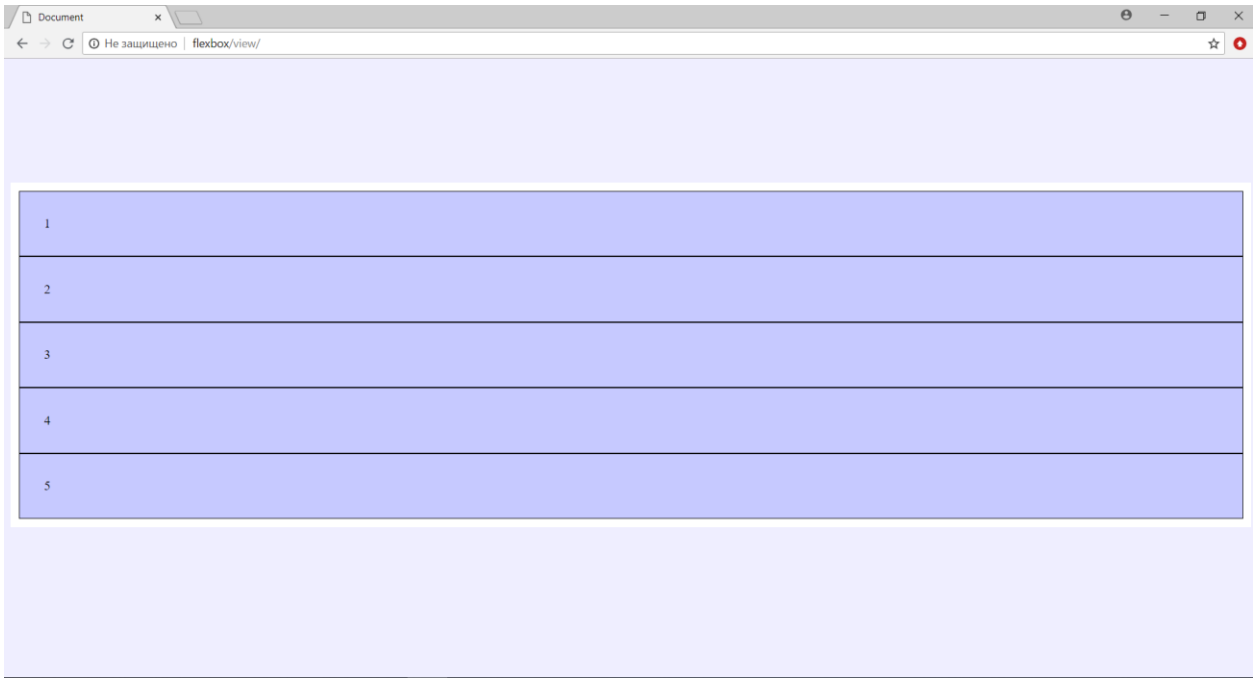

Рисунок 3. Начальный вид

Элементы пытаются занять всю ширину экрана. Почти в каждом макете присутствуют подобные ситуации, когда нужно разместить блок элементов на различных расстояниях и местах. Чтобы элементы выстроились по горизонтали раннее все пользовались свойством float, но есть огромное

количество минусов в дальнейшем использовании. Запишем главному контейнеру свойство display:flex и все входящие в него блоки станут flex элементами, которые автоматически выстроятся по горизонтали, но так же будут пытаться занять всю ширину экрана. Так как они выстраиваются по основной оси, то только главный контейнер белого цвета будет вытянут на всю страницу (см. рис. 4).

Рисунок 4. Flex элементы

Следующие свойство flex-direction, может принимать 4 параметра:

1. row (по горизонтали);
2. row-reverse (по горизонтали, но элементы выстраиваются справа налево);
3. column (по вертикали);
4. column-reverse (по вертикали, но элементы выстраиваются снизу вверх).

Выстроим элементы по поперечной оси, снизу вверх используя 4 параметр (см. рис. 5).

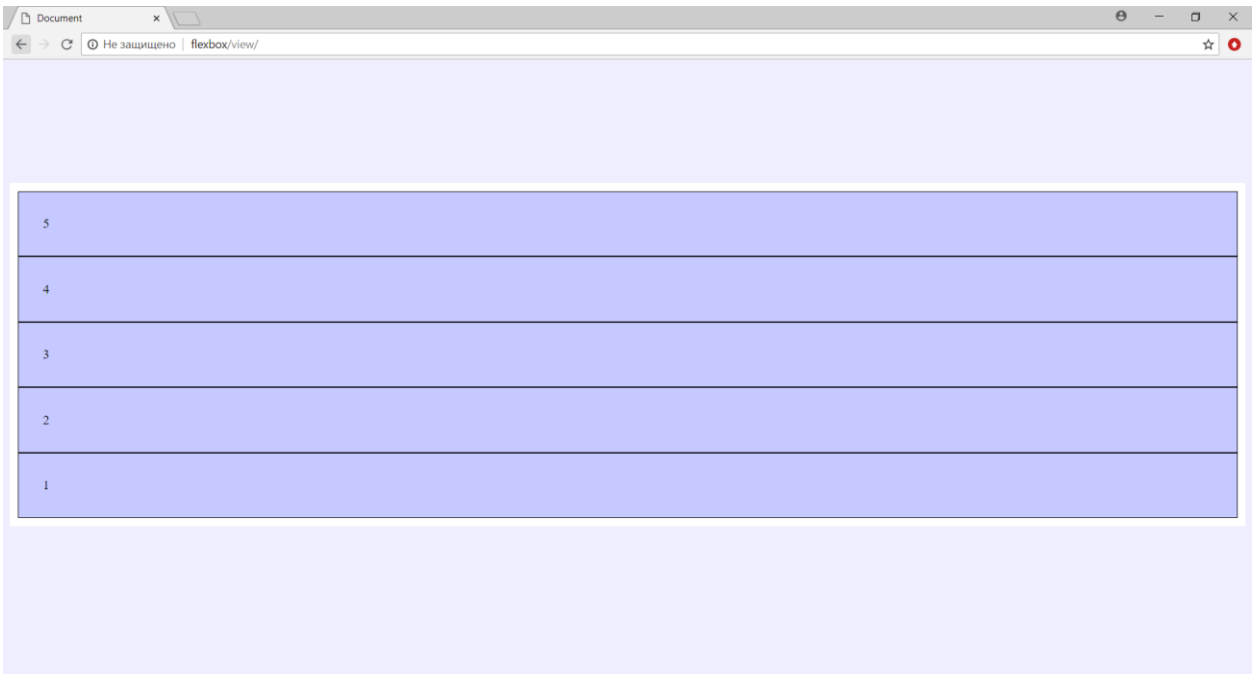

Рисунок 5. Flex-direction

Следующее свойство создаёт адаптивность для созданных блоков. flex-wrap имеет 3 параметра:

1. nowrap (по умолчанию, отключает свойство);
2. wrap (переносит элементы, чтобы все элементы были видны);
3. wrap-reverse (переносит элементы, но сверху вниз).

Используем параметр wrap, чтобы при мелких разрешениях все блоки были видны (см. рис. 6).

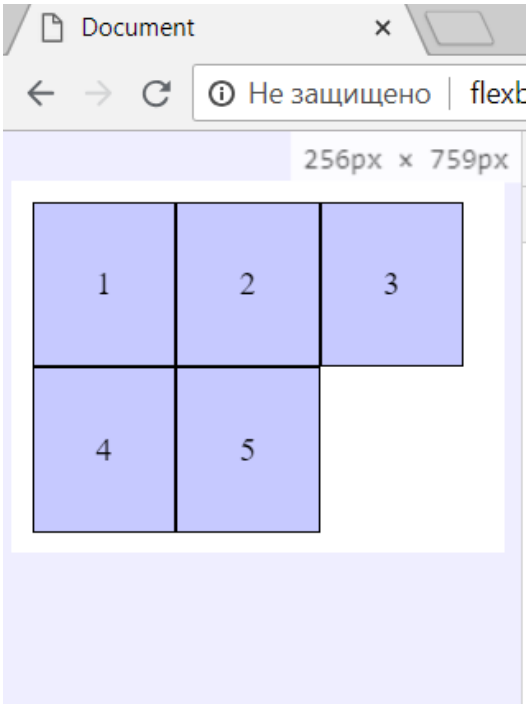

Рисунок 6. Flex-wrap

Теперь все элементы страницы полностью адаптивны. Ещё одно не мало важное свойство justify-content:

1. flex-start (по умолчанию);
2. flex-end (элементы перемещаются в конец оси);
3. center (по центру);
4. space-between (элементы выстраиваются так, что между ними становится одинаковое расстояние, но 1 и последний элементы располагаются по краям контейнера);
5. space-around (элементы равномерно распределяются по всей строке, но пустое пространство перед первым и после последнего элементов равно половине пространства между двумя соседними элементами.);
6. space-evenly (расстояние между любыми двумя соседними элементами, а также перед первым и после последнего, одинаковые).

Возьмём 5 параметр (см. рис. 7).

Рисунок 7. Justify-content

Свойство justify-content двигало элементы по основной оси, но есть ещё поперечная ось, которая идет сверху вниз. Добавим к стилям главного контейнера высоту в 200px и возьмём свойство align-items:

1. stretch (по умолчанию);
2. flex-start (выравниваются к началу поперечной оси);
3. flex-end (выравниваются к концу оси);
4. center (выравниваются по центру);
5. baseline (выравниваются к базовой границе).

Используем свойство для выравнивания по центру поперечной оси (см. рис. 8).

Рисунок 8. Align-items

А если нужно будет первый элемент выровнять к верхней части, второй элемент к нижней части и третий по центру. Для этого понадобится свойство align-self:

1. flex-start (к верхней части);
2. flex-end (к нижней части);
3. center (по центру);

4 и 5 элементы оставим без изменений (см. рис. 9).

Рисунок 9. Align-self

Данный инструмент может помочь разработчику в трудных ситуациях там, где другие не эффективны. Исходя из выше сказанного можно сделать вывод, что данная технология удобна и проста в использовании. Не нужно скачивать библиотеки или месяцами учится пользоваться новым инструментом.

Библиографический список

1. Биктимиров Р.Р., Джемалетдинов А.Б. Современные инструменты front-end разработки // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2016. № 3 (13). С. 63-69.
2. Филимоненкова Т.Н., Дунаевский А.С. Технологии адаптивной верстки сайтов // world science: problems and innovations. 2017. С. 78-81.
3. Фадеева А.С. Адаптивный дизайн или мобильная версия сайта:

преимущества и недостатки // Актуальные проблемы авиации и космонавтики. 2017. Т. 3. № 13. С. 1106-1107.
4. Дмитриева Ю.С. Методика адаптивной модернизации веб-интерфейсов // перспективы развития науки в современном мире 2017. С. 142-148.