



Урок 8

Углубленное изучение CSS3. Препроцессоры

Flexbox. Препроцессор Sass

[Flexbox](#)

[Свойства flex-контейнера](#)

[justify-content](#)

[align-items](#)

[flex-direction](#)

[flex-wrap](#)

[Свойства flex-элементов](#)

[align-self](#)

[Препроцессор SASS](#)

[Установка](#)

[Препроцессинг](#)

[Переменные](#)

[Математические операторы](#)

[Вложенности](#)

[Фрагментирование](#)

[Примеси](#)

[Наследование](#)

[Домашнее задание](#)

[Дополнительные материалы](#)

[Используемая литература](#)

Flexbox

CSS flexbox (Flexible Box Layout Module) – это система компоновки элементов. Она состоит из flex-контейнера – родительского контейнера и flex-элементов – дочерних блоков. Дочерние элементы могут выстраиваться в строку или столбик, а оставшееся свободное пространство распределяется между ними различными способами.

Благодаря flexbox элементы ведут себя предсказуемо на всех типах устройств и при различных размерах экрана. Модель гибкой разметки имеет преимущества перед блочной разметкой за счет отсутствия плавающих элементов (свойство float) и отсутствия схлопывания внешних отступов margin.

Дочерние flex-элементы можно располагать в любом направлении, при этом они имеют гибкие размеры, позволяющие им приспосабливаться к размерам экрана. Первоначальный порядок отображения элементов зависит от их порядка в исходном коде. Изменить направление и порядок отображения элементов можно за счет свойств flex-direction и order.

Flex-контейнер ведет себя как блочный элемент в случае display: flex; или как строчный для display: inline-flex;. В зависимости от размера экрана, элементы в контейнере расширяются для заполнения свободного пространства или уменьшаются для избежания переполнения контейнера.

В модели flexbox для внутренних блоков не работают такие css-свойства, как float, clear, vertical-align. На flex-контейнер не оказывают влияние свойства column-, задающие колонки в тексте.

По умолчанию flex-элементы располагаются по горизонтали – вдоль главной оси main axis, а поперечная ось cross axis пересекает главную. Задавая направление контента, можно поменять местами направления главной и поперечной осей. Если задать flex-flow: row; или flex-flow: row-reverse;, главная ось будет расположена горизонтально, а поперечная – вертикально. Если задать flex-flow: column; или flex-flow: column-reverse;, то главная ось будет расположена вертикально, а поперечная – горизонтально.

Свойства flex-контейнера

```
display: flex; //для родительского элемента
```

После установки данных значений свойства каждый дочерний элемент автоматически становится flex-элементом, выстраиваясь в ряд (вдоль главной оси) колонками одинаковой высоты, равной высоте блока-контейнера. При этом блочные и строчные дочерние элементы ведут себя одинаково, т. е. ширина блоков равна ширине их содержимого с учетом внутренних полей и рамок элемента.

justify-content

Свойство выравнивает flex-элементы по ширине flex-контейнера, распределяя оставшееся свободное пространство, незанятое flex-элементами. Для выравнивания элементов по вертикали используется свойство align-content.

- **flex-start** – значение по умолчанию. Flex-элементы позиционируются от начала flex-контейнера;
- **flex-end** – flex-элементы позиционируются относительно правой границы flex-контейнера;
- **center** – flex-элементы выравниваются по центру flex-контейнера;

- **space-between** – flex-элементы выравниваются по главной оси, свободное место между ними распределяется следующим образом: первый блок располагается в начале flex-контейнера, последний блок – в конце, все остальные блоки равномерно распределены в оставшемся пространстве, а свободное пространство равномерно распределяется между элементами;
- **space-around** – flex-элементы выравниваются по главной оси, а свободное место делится поровну, добавляя отступы справа и слева.

align-items

Свойство выравнивает flex-элементы, в том числе и анонимные, по перпендикулярной оси (по высоте). Не наследуется.

- **stretch** – значение по умолчанию. Flex-элементы растягиваются, занимая все пространство по высоте;
- **flex-start** – flex-элементы выравниваются по левому краю flex-контейнера относительно верхнего края блока-контейнера;
- **flex-end** – flex-элементы выравниваются по левому краю flex-контейнера относительно нижнего края блока-контейнера;
- **center** – flex-элементы выравниваются по центру flex-контейнера;
- **baseline** – flex-элементы выравниваются по базовой линии.

flex-direction

Свойство задает направление главной оси flex-контейнера и тем самым определяет, каким образом flex-элементы укладываются во flex-контейнере. Они могут располагаться в двух главных направлениях – горизонтально, как строки и вертикально, как колонки. Главная ось по умолчанию идет слева направо. Поперечная – сверху вниз.

- **row** – значение по умолчанию, слева направо (в rtl справа налево). Flex-элементы выстраиваются в строку. Начало (main-start) и конец (main-end) направления главной оси соответствуют началу (inline-start) и концу (inline-end) инлайн оси (inline-axis);
- **row-reverse** – направление справа налево. Flex-элементы выстраиваются в строку относительно правого края контейнера;
- **column** – направление сверху вниз. Flex-элементы выстраиваются в колонку;
- **column-reverse** – flex-элементы выстраиваются в колонку в обратном порядке, снизу вверх.

flex-wrap

Свойство управляет тем, как flex-контейнер будет выкладывать flex-элементы – в одну строку или в несколько, и направлением, в котором будут укладываться новые строки. По умолчанию flex-элементы выстраиваются в одну строку. При переполнении контейнера их содержимое будет выходить за границы flex-элементов. Не наследуется.

- **nowrap** – значение по умолчанию. Flex-элементы не переносятся, а располагаются в одну линию слева направо (в rtl – справа налево);
- **wrap** – flex-элементы переносятся, располагаясь в несколько горизонтальных рядов (если не помещаются в один ряд) слева направо (в rtl – справа налево);
- **wrap-reverse** – flex-элементы переносятся, располагаясь в обратном порядке, слева направо, при этом перенос происходит снизу вверх.

Свойства flex-элементов

order

Свойство определяет порядок, в котором flex-элементы отображаются внутри flex-контейнера. По умолчанию для всех flex-элементов задан порядок `order: 0`; и они следуют друг за другом по мере добавления во flex-контейнер. Самый первый flex-элемент по умолчанию расположен слева. Чтобы поставить любой flex-элемент в начало строки, ему нужно назначить `order: -1`; в конец строки – `order: 1`;

align-self

Свойство отвечает за выравнивание отдельно взятого flex-элемента по высоте flex-контейнера. Переопределяет выравнивание, заданное `align-items`.

- **auto** – значение по умолчанию. Flex-элемент использует выравнивание, указанное в свойстве `align-items` flex-контейнера;
- **flex-start** – flex-элемент выравнивается по верхнему краю flex-контейнера относительно левой границы;
- **flex-end** – flex-элемент выравнивается по нижнему краю flex-контейнера относительно левой границы.
- **center** – flex-элемент выравнивается по высоте посередине flex-контейнера относительно левой границы.
- **baseline** – flex-элемент выравнивается по базовой линии flex-контейнера относительно левой границы.
- **stretch** – flex-элемент растягивается на всю высоту flex-контейнера, при этом учитываются поля и отступы.

Препроцессор Sass

Установка

Чтобы начать работу, нужно установить gem Sass:

```
gem install sass
```

Если вы работаете под Windows, сначала вам необходимо установить [Ruby](#). Для запуска Sass из командной строки используйте команду 1:

```
sass input.scss output.css
```

Также вы можете указать Sass следить за файлом и автоматически его компилировать в CSS при любом изменении:

```
sass --watch input.scss:output.css
```

Если у вас в папке имеется несколько файлов Sass, то вы можете указать Sass следить за всей папкой:

```
sass --watch app/sass:public/stylesheets
```

Используйте команду `sass --help` для получения полной документации.

Препроцессинг

Писать CSS само по себе весело, но когда таблица стилей становится огромной, то ее становится сложно обслуживать. В таком случае нам поможет препроцессор. Sass позволяет использовать функции, недоступные в самом CSS, например переменные, вложенности, миксины, наследование и другие приятные вещи, возвращающие удобство написания CSS.

Как только вы начинаете пользоваться Sass, препроцессор обрабатывает ваш Sass-файл и сохраняет его как простой CSS-файл, который вы сможете использовать на любом сайте.

Переменные

Рассматривайте переменные как способ хранения информации, которую вы хотите использовать на протяжении написания всех стилей проекта. Вы можете хранить в них цвета, стеки шрифтов или любые другие значения CSS. Чтобы создать переменную в Sass, нужно использовать символ `$`.

Математические операторы

Использовать математику в CSS очень полезно. Sass поддерживает несколько стандартных математических операторов, таких как `+`, `-`, `*`, `/` и `%`.

Вложенности

При написании HTML вы, наверное, заметили, что в нем царит четкая иерархия. В случае с CSS это не так. Однако Sass позволит вам вкладывать селекторы CSS друг в друга таким же образом, как и элементы HTML. Но помните, что слишком много уровней вложенности сделает ваш документ менее читаемым. Это считается плохой практикой.

Фрагментирование

Вы можете создавать фрагменты Sass-файла, содержащие небольшие отрывки CSS, которые можно будет использовать в других Sass-файлах. Это отличный способ сделать ваш CSS модульным и облегчить его обслуживание. Фрагмент — это Sass-файл, имя которого начинается с нижнего подчеркивания, например, `_partial.scss`. Нижнее подчеркивание в имени Sass-файла говорит компилятору о том, что это только фрагмент, и он не должен компилироваться в CSS. Фрагменты Sass подключаются при помощи директивы `@import`.

Примеси

Некоторые вещи в CSS писать весьма утомительно, особенно в CSS3, где зачастую требуется использовать большое количество вендорных префиксов. Миксины позволяют создавать группы деклараций CSS, которые вам нужно использовать по нескольку раз на сайте. Хорошо использовать миксины для вендорных префиксов.

Для создания миксина используйте директиву `@mixin` + название этого миксина. Мы назвали наш миксин `border-radius`. Также в миксине мы используем переменную `$radius` внутри скобок, тем самым позволяя себе передавать в переменную все, что захотим. После того, как вы создали миксин, вы можете его использовать в качестве параметра CSS, начиная вызов с `@include` и имени миксина.

Наследование

Это одна из самых полезных функций Sass. Используя директиву `@extend`, можно наследовать наборы свойств CSS от одного селектора другому. Это позволяет держать ваш Sass-файл в «чистоте».

Практическое задание

1. Завершить работу над проектом.
2. Разместить все ваши работы на хостинге.
3. Применить новый препроцессор Sass.
4. Прислать ссылку на ваш размещённый проект.

Используемая литература

1. <http://www.wisdomweb.ru/>.
2. <http://html5book.ru/css3-flexbox/>.
3. <http://sass-scss.ru/guide/>.
4. Гоше Х. HTML5. Для профессионалов. СПб.: Питер, 2013. – 496 с.
5. Хоган Б. HTML5 и CSS3. Веб-разработка по стандартам нового поколения. СПб.: Питер, 2012.
6. Макфарланд Д. Большая книга CSS3. СПб.: Питер, 2014.