# Less

Less — это препроцессор CSS, позволяющий использовать переменные, функции, циклы и другие технологии для упрощения работы со стилями. Препроцессор в данном случае означает, что мы имеем дело с динамическим языком стилей, который преобразуется в CSS. Таким образом, на выходе получаем стандартный стилевой файл.

Less преследует благородные цели — облегчить работу разработчикам сайтов дав им подходы и приёмы, которые в CSS на данный момент отсутствуют, но характерны для языков программирования. Например, те же переменные позволяют задать какое-либо значение, а затем использовать его многократно, подставляя лишь имя переменной.

Разработал Less Алексис Селье на языке Ruby, а затем под влиянием нарастающей популярности Node.js переписал код на JavaScript. Характерной особенностью Less стал синтаксис основанный на CSS. Это сразу же снизило порог вхождения и изучения новой технологии, к тому же не возникает никаких проблем с редакторами кода, они прекрасно поддерживают подсветку синтаксиса. В дальнейшем Less оказал своё влияние на препроцессор Sass, в котором появился новый похожий синтаксис. Но в целом, вопрос, кто на кого оказал влияние открыт, потому что многие полезные вещи были заимствованы друг у друга. Так что в настоящий момент и Less и Sass похожи как братья, различаясь лишь в деталях.

## См. также

Официальный сайт Less

http://lesscss.org

## Автор

**Влад Мержевич**
Веб-разработчик, автор нескольких книг, посвящённых созданию сайтов, HTML и CSS. Кандидат технических наук.

## *WebReference.ru*

Руководство взято с сайта webref.ru.

# Комментарии в Less

Комментарии делают ваш код удобнее и понятнее, особенно если вы работаете в команде или периодически возвращаетесь к редактированию. К тому же число самих комментариев никак не оказывает влияние на объём конечного файла CSS, потому что комментарии при компиляции удаляются и видны только в исходном документе. Так что не бойтесь комментировать много и часто, это повышает читаемость.

Less поддерживает два типа комментариев: многострочные и однострочные.

## Многострочные комментарии

Чтобы закомментировать блок кода поместите его между /* и */.

```
/*
  style.less
  Версия 1.0
*/
```

## Однострочные комментарии

Однострочные комментарии начинаются с // и всё, что идёт после этих символов Less игнорирует. Так что эта форма иногда применяется для быстрого отключения отдельных свойств.

```
@size: 200px; // Ширина и высота элемента
```

## Вложенные комментарии

Внутри одного типа комментария допустимо вставлять другой тип, как показано ниже.

```
/*
  // style.less
  // Версия 1.0
*/
// Используйте /* */ для комментирования
```

Однако одинаковый тип комментария внутри другого не допустим. Ниже показан пример, который приведёт к ошибке.

```
/*
/* Так делать нельзя! */
*/
```

## Специальные комментарии

Любые комментарии в итоговом CSS-файле удаляются, но иногда комментарий нужно оставить, например, информацию об авторских правах или важное пояснение. Для таких комментариев применяется специальная форма /*! !*/.

```
/*! Этот комментарий будет сохранён. !*/
/* А этот нет. */
```

Впрочем, некоторые компиляторы сохраняют даже обычные многострочные комментарии и удаляют их лишь при явно заданной минимизации кода. Но специальные комментарии сохраняются всегда, независимо от настроек компилятора.

# Переменные в Less

Переменные нужны для организации и упрощения написания кода и позволяют предварительно задать популярные значения, а затем повторно использовать их в любом месте кода просто написав имя переменной. Это удобно, потому что нам при необходимости не придётся отыскивать и править множество значений, достаточно изменить его в одном месте и значения везде заменятся автоматически.

Переменные начинаются с символа @, затем идёт имя переменной, оно может включать в себя латинские символы, символ подчёркивания (_) и дефис (-). После имени пишем двоеточие и любое значение допустимое в CSS, либо имя другой переменной. Всё это похоже на синтаксис CSS, так что должно быть знакомо.

```
@dark-color: #333;
  @light-color: #f0f0f0;

.creature {
  background-color: @dark-color;
  color: @light-color;
}
```

Теперь в любом месте кода можем вызвать переменную подставляя её имя. В CSS вместо переменной будет написано её значение. Вот что в итоге получится.

```
.creature {
  background-color: #333;
  color: #f0f0f0;
}
```

Переменная не является величиной постоянной, это не константа, так что переменную можно переопределять, присваивая ей при необходимости новое значение.

```
@color: red;
p { color: @color; }
@color: blue;
div { color: @color; }
```

Значение действует с момента инициализации переменной до её переназначения. В CSS получим следующий код:

```
p {
  color: #0000ff;
}
div {
  color: #0000ff;
}
```

На деле переменные не ограничиваются подстановкой значений, переменные можно добавлять в качестве имён селекторов, путей и др. В этом случае синтаксис вызова переменной немного изменится и обращаться к ней надо как @{var}. Ниже показано, как это делается.

```
@dark-color: #333;
@light-color: #f0f0f0;
@path-img: "../img";
@block: block;

.@{block} {
  background: url("@{path-img}/pic.png") no-repeat;
}
.@{block}-title {
  background:@dark-color;
  color: @light-color;
}
.@{block}-content {
```

```
  padding: 10px;
}
```

Вот что мы получим на выходе.

```
.block {
  background: url("../img/pic.png") no-repeat;
}
.block-title {
  background: #333333;
  color: #f0f0f0;
}
.block-content {
  padding: 10px;
}
```

## Вычисления

Переменные могут участвовать в математических выражениях и их можно складывать, вычитать, умножать и делить между собой или с другими значениями. Разумеется, правильный результат будет только в том случае, если данные сочетаются, так что складывать проценты с пикселями не получится. Ниже используется деление, сложение и вычитание переменных.

```
@num-columns: 3;
@bg-color: #3ac;
@column-width: 100% / @num-columns;
@size: 100% - 20px;
@bg-new: @bg-color + 30;
```

В результате вычисления мы получим следующие значения:

```
@num-columns: 3;
@bg-color: #3ac;
@column-width: 33.33333333%;
@size: 80%;
@bg-new: #51c8ea;
```

Как видите, Less при отнимании пикселей из процентов не выдаёт ошибку, а просто заменяет одни единицы на другие. В итоге он вычитает 20% от 100%, а никак не пиксели. Так что учитывайте этот момент в своих вычислениях.

## Ленивая загрузка

Паша дал Маше два яблока, а Петя ей три. Сколько яблок стало у Маши? Нет, не пять, потому что мы не знаем, сколько яблок было у Маши первоначально. Мораль: обнуляйте переменные. Этот программистский анекдот справедлив при работе с языками вроде JavaScript, но не с Less. Потому что Less применяет паттерн Lazy Loading (ленивая загрузка). Принцип такой — если в коде встречается переменная, значение которой выше не определено, то работа с ней откладывается, пока не встретится инициализация переменной. Это несколько усложняет работу с кодом, потому что его уже нельзя читать сверху вниз, но зато снижает число возможных ошибок.

В этом примере мы вначале присваиваем свойству color значение переменной, которое на этот момент не известно. Затем уже ниже определяем саму переменную и её значение.

```
body { color: @bg-color; }
@bg-color: #3ac;
```

## См. также

Документация по переменным Less

http://lesscss.org/features/#variables-feature

# Примеси

Примеси (англ. mixin) позволяют добавлять существующие стилевые правила к другим селекторам. Это сокращает код, поскольку нам уже нет смысла повторять один и тот же фрагмент кода несколько раз, достаточно сослаться на него в нужном месте. Примеси в этом плане похожи на функции в языках программирования.

Примеси в Less это обычные селекторы классов или идентификаторы. В примере ниже мы создаём два класса — left-mixin и right-mixin для выравнивания элементов, соотвественно, по левому или правому краю веб-страницы. Если нам требуется повторить свойства у другого селектора, то мы пишем имя класса с точкой впереди — .left-mixin или .left-mixin(). Для идентификатора вызов будет как #left-mixin. Круглые скобки пишутся по желанию, они в данный момент не играют роли.

```
.left-mixin {
  float: left;
  margin: 0 10px 10px 0;
}
.right-mixin {
  float: right;
  margin: 0 0 10px 10px;
}
.img-left {
  .left-mixin;
  background: #f0f0f0;
  padding: 10px;
}
.img-right {
  .righ-mixin;
  background: #f0f0f0;
  padding: 10px;
}
```

На выходе мы получим такой стиль:

```
.left {
  float: left;
  margin: 0 10px 10px 0;
}
.right {
  float: right;
  margin: 0 0 10px 10px;
}
.img-left {
  float: left;
  margin: 0 10px 10px 0;
  background: #f0f0f0;
  padding: 10px;
}
.img-right {
  float: right;
  margin: 0 0 10px 10px;
  background: #f0f0f0;
  padding: 10px;
}
```

Заметьте, что повторяющиеся свойства не группируются по селекторам и в коде остались наши классы left-mixin и right-mixin. Если они сами по себе не требуются и нужны лишь для вызова в других местах, то их можно убрать из CSS, добавив к имени класса круглые скобки. В остальном никаких изменений нет.

```
.left-mixin() {
  float: left;
  margin: 0 10px 10px 0;
```

```
}
.img-left {
  .left-mixin;
  background: #f0f0f0;
  padding: 10px;
}
```

В CSS останется только следующее:

```
.img-left {
  float: left;
  margin: 0 10px 10px 0;
  background: #f0f0f0;
  padding: 10px;
}
```

Скобки в примесях нужны ещё и для передачи в них переменных, как показано ниже.

```
.animation(@name, @time, @time-fun) {
  animation: @name @time @time-fun;
  -webkit-animation: @name @time @time-fun;
}
body {
  background: url(images/city.png) repeat-x 0 100% fixed,
  linear-gradient(to top, #5080b1, #004e8c) fixed;
  .animation(city, 30s, infinite);
}
```

Мы используем примесь с именем animation и передаём ей три переменные, которые подставляются в качестве параметров свойств animation и -webkit-animation. На выходе получим следующий код CSS:

```
body {
  background: url(images/city.png) repeat-x 0 100% fixed,
  linear-gradient(to top, #5080b1, #004e8c) fixed;
  animation: city 30s infinite;
  -webkit-animation: city 30s infinite;
  }
```

Таким образом, примеси удобно использовать для свойств, которые работают в браузерах с префиксами, вроде animation. Это позволяет избежать ошибок при копировании значений, т. к. они заменяются компилятором автоматически.

## См. также

Руководство по примесям

http://lesscss.org/features/#mixins-feature

# Вложения в Less

CSS не зря расшифровывается как каскадные таблицы стилей, потому что одни элементы могут находиться внутри других и к ним последовательно применяются стилевые правила. Для наглядного представления структуры элементов в Less применяются вложения, которые затем преобразуются во вложенные селекторы. Например, в коде HTML у нас имеется следующая структура элементов, схематично показанная ниже.

```
<aside>
  <div class="block">
    <ul>
      <li>...</li>
      <li>...</li>
    </ul>
  </div>
</aside>
```

Чтобы сохранить структуру в CSS мы вкладываем одни правила внутрь фигурных скобок родителя.

```
aside {
  .block {
    margin-bottom: 30px;
    ul {
      margin: 0;
      padding: 0;
      font-size: 14px;
      li {
        border-top: 1px solid #f4f4f4;
        list-style: none;
        color: #666;
        padding: 3px 0;
        position: relative;
        &:first-child {
          border-top: none;
        }
      }
    }
  }
}
```

Для добавления псевдоклассов и псевдоэлементов применяется амперсанд (символ &), он говорит Less что надо взять и подставить вместо него родительский селектор, в данном случае li. В итоге получим такой стиль.

```
aside .block {
  margin-bottom: 30px;
}
aside .block ul {
  margin: 0;
  padding: 0;
  font-size: 14px;
}
aside .block ul li {
  border-top: 1px solid #f4f4f4;
  list-style: none;
  color: #666;
  padding: 3px 0;
  position: relative;
}
aside .block ul li:first-child {
  border-top: none;
}
```

Less полностью сохраняет структуру вложения и показывает её в именах селекторов. Для сложных комбинаций вроде родственных селекторов (A+B, A>B, A~B) применяется амперсанд, а после него пишется соответствующий знак.

В качестве примера рассмотрим поле для ввода чисел. Если введённое пользователем число меньше заранее установленного минимального или больше максимального значения, то выводим сообщение об ошибке. Для этого применяется псевдокласс :out-of-range и псевдоэлемент ::after. Код HTML для этого будет следующий:

```
<p><input type="number" id="dec" min="1" max="10" value="1">
<label for="dec"></label></p>
```

CSS:

```
input[type="number"]:out-of-range {
  background: #f9c3d2;
}
input[type="number"]:out-of-range + label::after {
  content: 'Неверное число';
  color: #ec008c;
  margin-left: 0.5em;
}
```

Поскольку у нас используется псевдокласс и псевдоэлемент, то активно применяется амперсанд и вложения одних стилевых правил в другие.

```
input[type="number"] {
  &:out-of-range {
    background: #f9c3d2;
    & + label {
      &::after {
        content: 'Неверное число';
        color: #ec008c;
        margin-left: 0.5em;
      }
    }
  }
}
```

# Компиляция Less

Существует несколько способов, как из Less-документа получить на выходе CSS-файл, уже понимаемый всеми браузерами:

- запуск в браузере;
- компиляция через Node.js;
- использование сторонних программ.

Самым простым вариантом является компиляция непосредственно в браузере, это основная фишка Less, которая отличает его от других препроцессоров CSS.

## Подключение Less

Скачайте Less с официального сайта по этой ссылке:

http://lesscss.org/#download-options

Или с сайта GitHub:

https://github.com/less/less.js/archive/master.zip

Затем подключите less.js или сжатый less.min.js на свою страницу через <script>.

```
<script src="less.js"></script>
```

Также можно использовать CDN (Content Delivery Network, сеть доставки контента), тогда не придётся самостоятельно скачивать файл. Подключение в этом случае происходит следующим образом.

```
<script src="//cdnjs.cloudflare.com/ajax/libs/less.js/2.3.1/less.min.js"></
script>
```

Теперь добавляем свой файл в формате Less на веб-страницу через элемент <link>. Обратите внимание на значение атрибута rel, оно отличается от традиционного. Кроме того, важна последовательность строк — вначале стиль, потом скрипт, это принципиально.

```
<link rel="stylesheet/less" href="styles.less">
<script src="less.js"></script>
```

Компиляция непосредственно в браузере занимает порой продолжительное время, поэтому рекомендуется только в отладочных целях. На готовом сайте для ускорения производительности применяйте уже скомпилированный CSS или выполняйте компиляцию на стороне сервера.

При редактировании кода окно браузера для просмотра изменений потребуется обновить вручную. Чтобы Less сам следил за изменениями есть два способа.

## Метод watch

Добавьте параметр env со значением development, а затем вызовите метод less.watch() как показано ниже.

```
<script>less = { env: 'development' };</script>
<script src="less.js"></script>
<script>less.watch()</script>
```

## Применение #!watch

В адресной строке браузера в конец URL добавьте #!watch. К примеру, адрес документа index.html,

тогда вызывайте его как index.html#!watch

Учтите, что компиляция с помощью Less.js происходит только под управлением веб-сервера и не работает для локальных файлов.

# Программа WinLess

WinLess — один из самых простых и популярных компиляторов Less под Windows, к тому же совершенно бесплатный для использования. Скачайте программу по этому адресу и запустите её после установки.

http://winless.org/online-less-compiler

## Компиляция

Для начала добавьте в программу папку с Less-файлами. Для этого перетащите папку напрямую в окно «Less files» или выберите папку через кнопку «Add folder» (рис. 1).

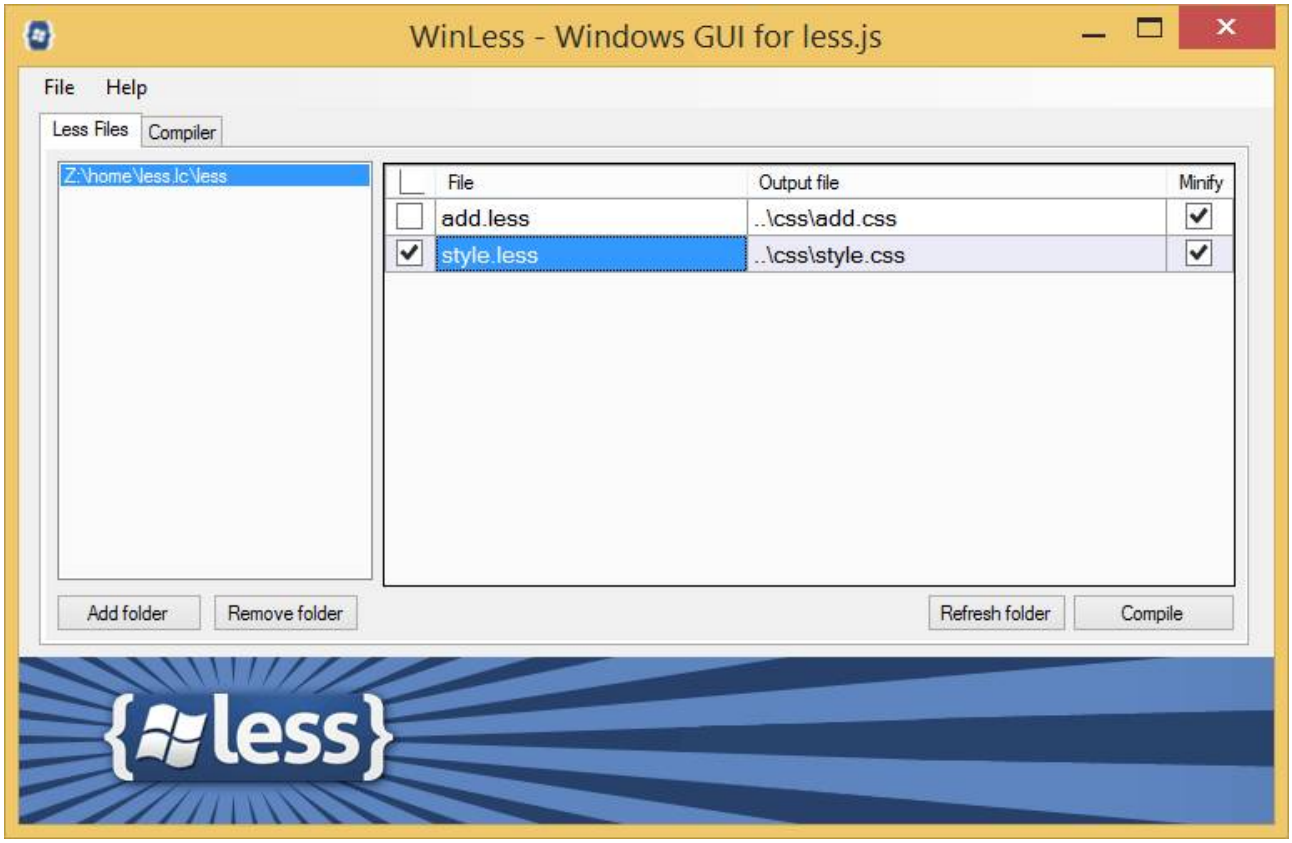


Рис. 1. Вид программы WinLess

По умолчанию готовые файлы сохраняются в папку с именем CSS. Если такой папки ещё не существует, то она будет создана и в неё добавлены новые стилевые файлы. Код можно сделать компактным, поставив галочку в пункте Minify, при этом все комментарии удаляются, а сам код записывается в одну строку. В итоге работать с конечным документом становится проблематично, зато уменьшается итоговый размер файлов.

Сама компиляция происходит двумя путями.

1. Добавляете папку с Less-файлами, отмечаете галочками те из них, которые следует преобразовать в CSS, указываете, минимизировать их или нет, после чего нажимаете кнопку Compile. При первой компиляции создаётся новая папка CSS и в неё сохраняются готовые файлы. В дальнейшем происходит только обновление их содержимого. Выходную папку при желании можно поменять, щёлкнув правой кнопкой мыши по файлу и выбрав в списке пункт «Select output file» (рис. 2).

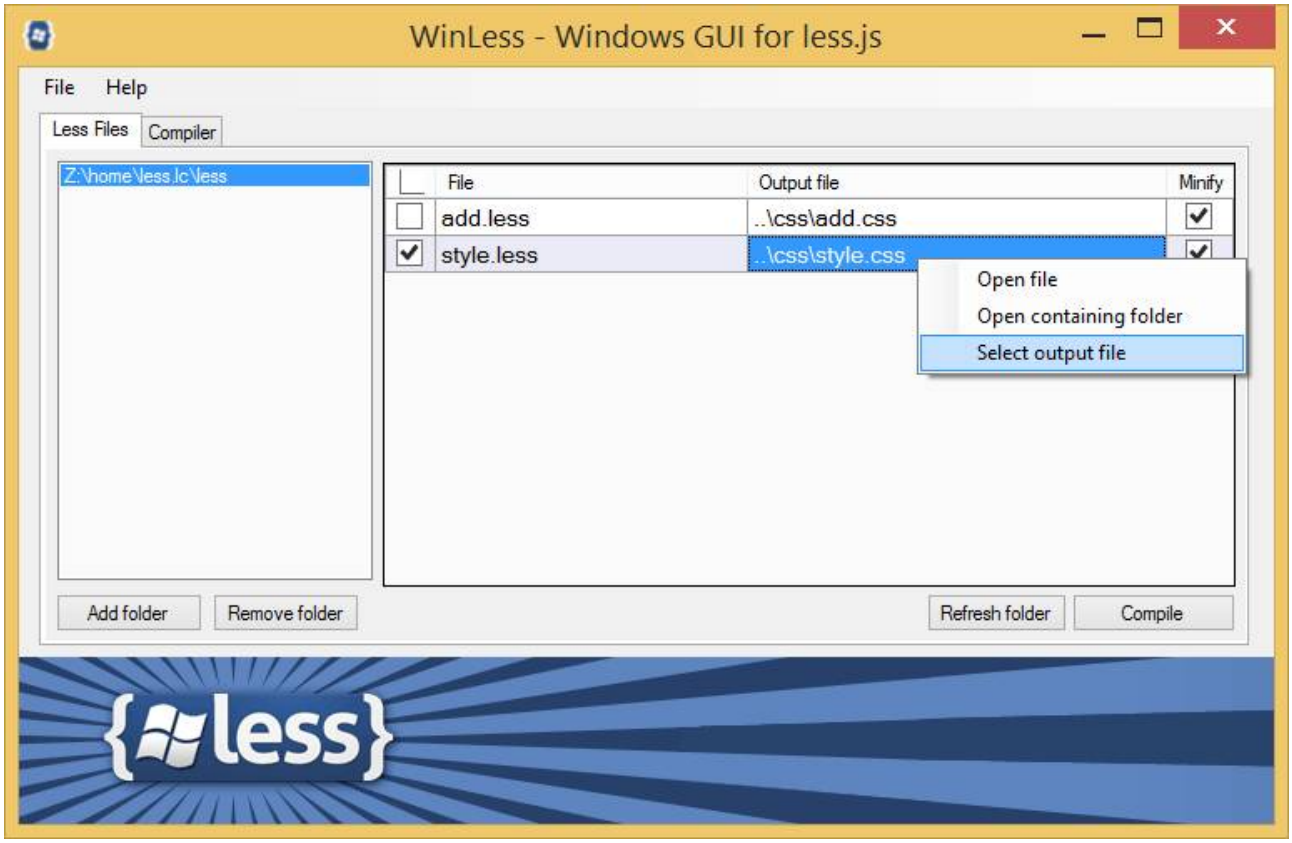


Рис. 2. Выбор файла для сохранения

2. После запуска программы компиляция происходит в автоматически фоновом режиме при сохранении Less-файлов. Это полезно при написании нового кода и работе над ним. Какие файлы компилировать и куда их сохранять определяется настройками WinLess. При возникновении ошибок программа выведет сообщение об их наличии.

## Настройки программы

Для вывода окна настроек выберите в меню пункт File > Settings (рис. 3).

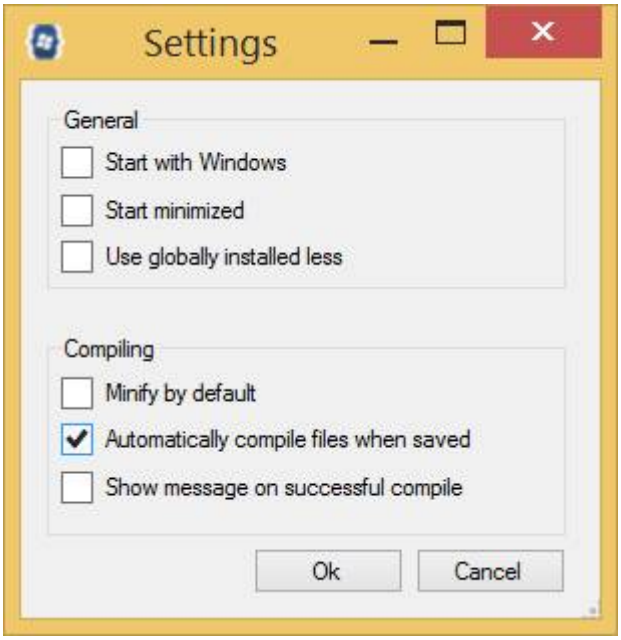


Рис. 3. Окно настроек WinLess

Start with Windows — запускать программу вместе с Windows. Опцию имеет смысл включить при активной работе с Less-файлами, когда их надо компилировать много и часто.

Start minimized — при запуске программа незаметно висит в трее Windows и тихо делает своё дело.

Use globally installed less — вместо компилятора в WinLess применяется компилятор, установленный через Node.js. Если такого нет, то будет показана ошибка.

Minify by default — по умолчанию все компилированные CSS-файлы оказываются компактными — из них удаляются пробелы, комментарии, табуляция и переносы строк.

Automatically compile files when saved — WinLess отслеживает добавленные в программу Less-документы и при их изменении в стороннем редакторе компилирует автоматически.

Show message on successful compile — после успешной компиляции появляется успешное сообщение об этом.

# Платформа Codepen

Codepen — это популярная онлайновая платформа для редактирования и хранения кода на HTML, CSS и JavaScript с просмотром готового результата в браузере. Полученным кодом можно делиться и видоизменять его при необходимости. Окно браузера делится на несколько рабочих областей, в которых отображается результат, а также код на HTML и CSS (рис. 1).

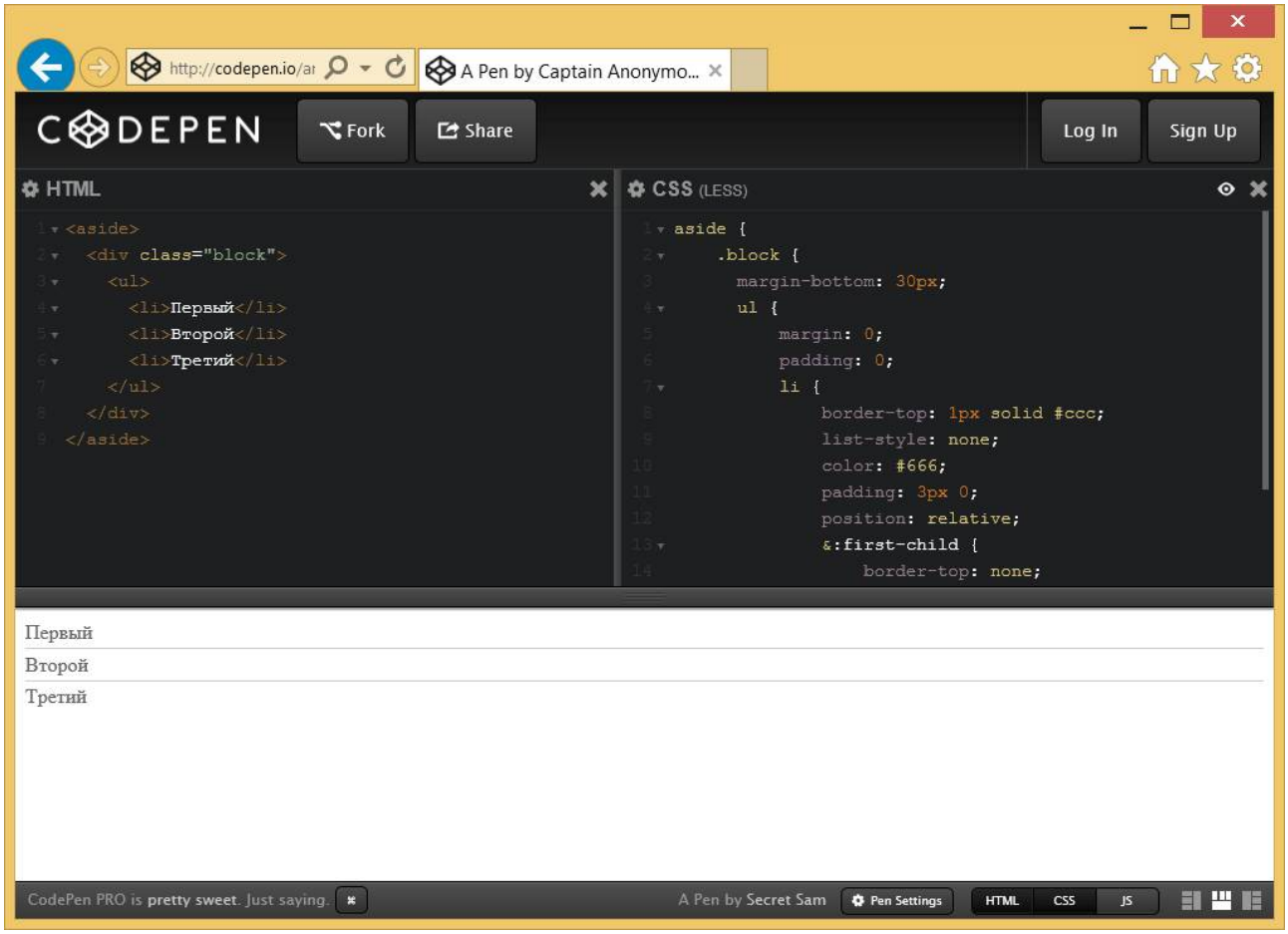


Рис. 1. Вид Codepen

Codepen не требует регистрации, но это рекомендуется сделать, если вам часто приходится пользоваться данной платформой. Это добавляет возможность управлять набором разного кода и возвращаться к его редактированию. Начать с чистого листа можно по этой ссылке:

http://codepen.io/pen

Или перейти к готовому коду по заранее известной ссылке, вроде этой:

http://codepen.io/anon/pen/BybpEg

В соответствующей рабочей области пишется код HTML и CSS, при этом служебные элементы вроде <head>, <body> и тому подобное добавлять не нужно. Сам результат написанного кода выводится в отдельной рабочей области, она обновляется автоматически при наборе кода.

Поскольку нас интересует не сам CSS, то для начала надо щёлкнуть по шестерёнке возле заголовка и в открывшемся списке задать Less. Тип выбранного препроцессора подсвечивается зелёным цветом и отображается в заголовке рабочей области (рис. 2).

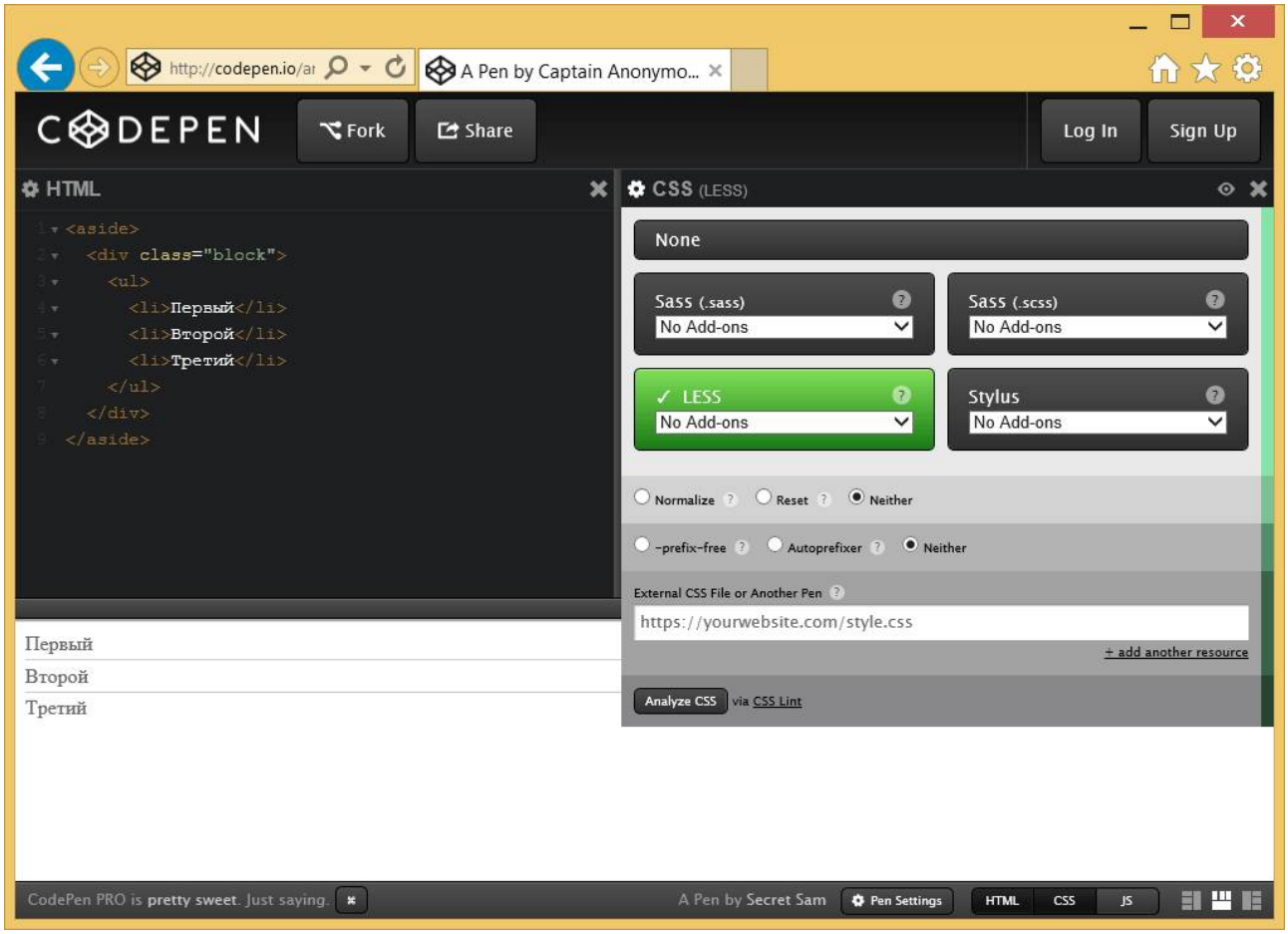


Рис. 2. Выбор Less

Кроме того, там же можно указать следующие настройки.

- Normalize — стилевая библиотека для приведения некоторых стилевых свойств к единому значению и обнуления других свойств. Иногда бывает необходима для получения одинакового результата в разных браузерах с учётом современных стандартов.
- Reset — стилевая библиотека от Эрика Мейера, преследует ту же цель что и Normalize.
- -prefix-free — небольшой скрипт, который автоматически добавляет необходимые вендорные префиксы (вроде -webkit и -moz) к стилевым свойствам.
- Autoprefixer — альтернативный скрипт добавляющий префиксы к свойствам, это позволяет писать чистый современный код CSS не задумываясь о поддержке в браузерах.
- External CSS File or Another Pen — подключение внешнего стилевого файла; актуально при использовании сторонних библиотек вроде Bootstrap.

Переключение между Less и готовым CSS делается при щелчке по иконке глаза, показанной на рис. 3. Итоговый CSS править нельзя, но допустимо выделить его и скопировать.

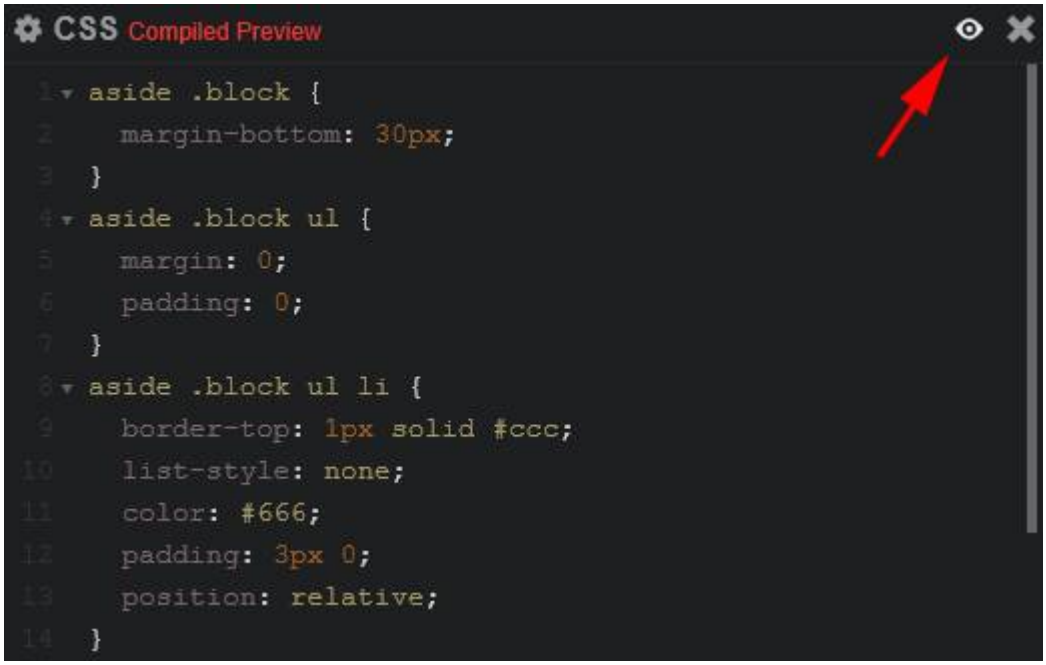


Рис. 3. Иконка для переключения вида

Кодом можно поделиться, предварительно сохранив его и скопировав адрес в браузере. Адрес всегда уникален и однозначно ведёт к нашему коду. Так что эту ссылку можно писать на форумах и в комментариях для демонстрации кода и результата. Кроме того в Codepen есть специальная кнопка Share, щёлчок по которой открывает специальное меню (рис. 4).

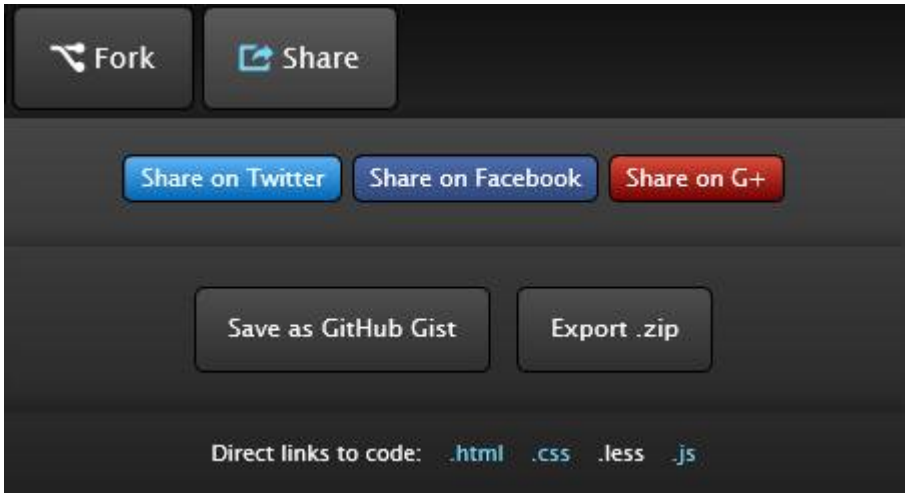


Рис. 4. Меню Share

Через Share можно поделиться кодом в социальных сетях, скачать полный архив и получить прямую ссылку на код HTML, CSS или Less.

# Программа Crunch

Crunch — простой редактор и компилятор Less сделанный на Adobe Air, за счёт чего работает в любых операционных системах. В редакторе имеется подсветка синтаксиса, возможность работать одновременно с несколькими файлами и группирование блоков. Скачать программу можно на официальном сайте по этой ссылке.

http://crunchapp.net

В программе доступны две вкладки. Для начала работы следует открыть папку содержащую Less-файлы через кнопку Open Website... (рис. 1).

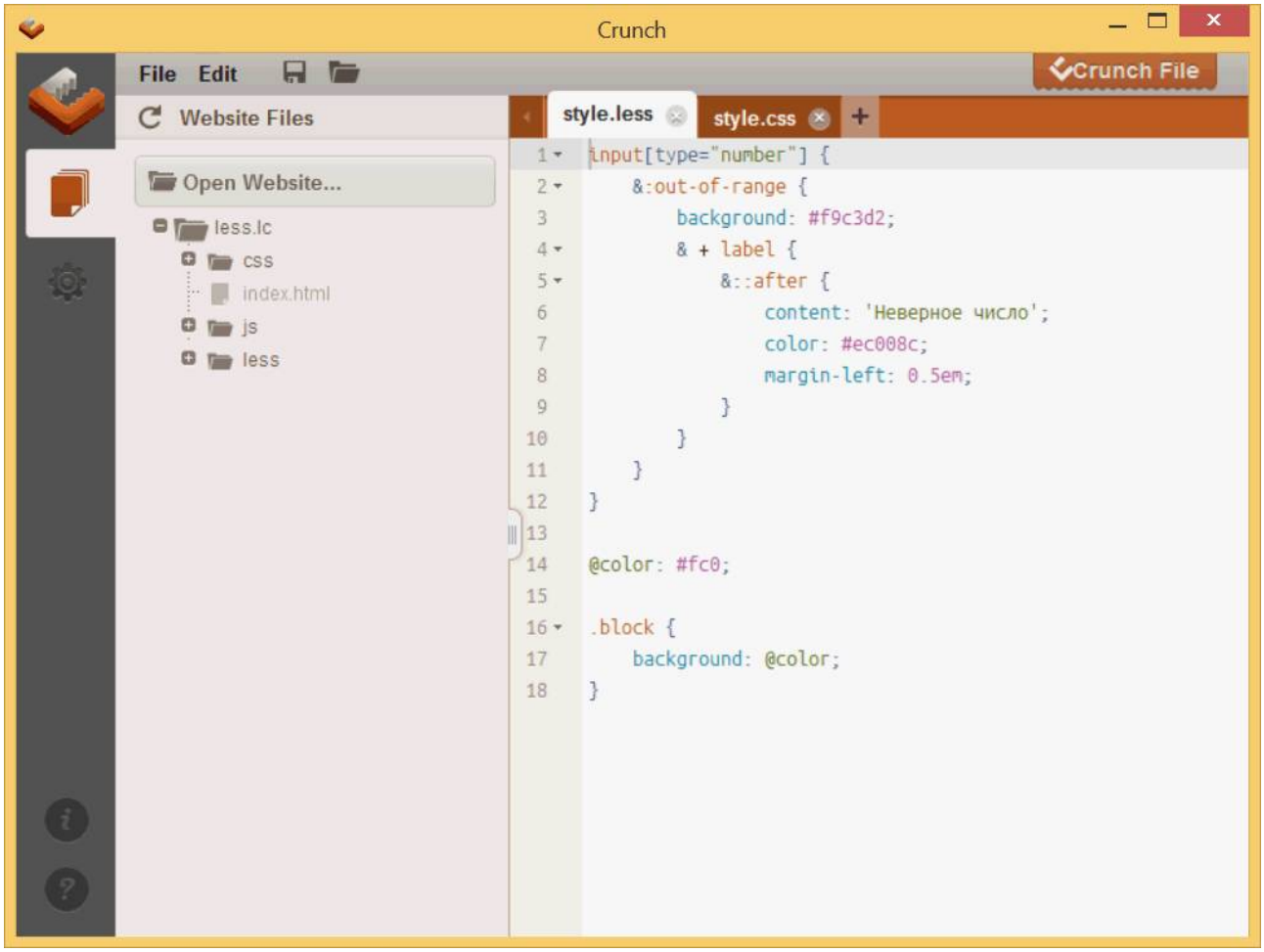


Рис. 1. Вкладка для работы с файлами

Как правило, исходные документы хранятся в папке с именем «less», а уже скомпилированные стилевые файлы — в папке «css». Самому создавать эту папку не требуется, она будет сделана автоматически при компиляции. Структура открытых файлов сохраняется и после закрытия и повторного запуска Crunch, что довольно удобно при работе с одним проектом.

Следующая вкладка Preferences содержит настройки программы и компилятора Less (рис. 2).

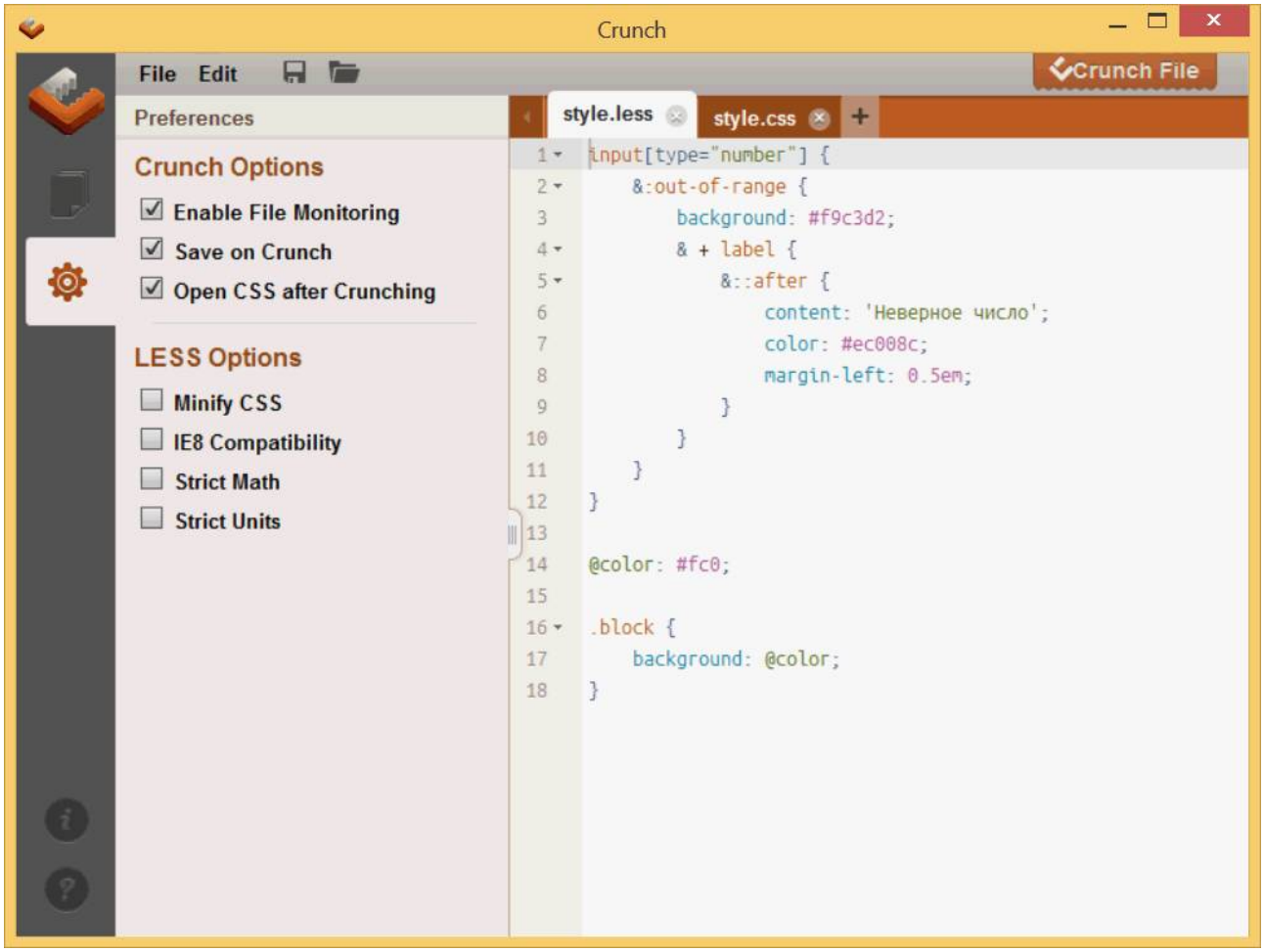


Рис. 2. Вкладка настроек

Все настройки делятся на два блока — настройки самой программы и настройки Less, которые влияют на получаемый результат.

**Enable File Monitoring**. Идёт отслеживание открытых во вкладках файлов и при их изменении внешней программой Crunch сообщит об этом.

**Save on Crunch**. Открытый файл сохраняется при компиляции.

**Open CSS after Crunching**. Вновь созданный стилевой файл открывается в новой вкладке для редактирования.

**Minify CSS**. Выходной файл сохраняется без пробелов в виде единой строки.

**IE8 Compatibility**. Режим совместимости с браузером Internet Explorer 8. В большинстве случаев не оказывает какое-либо влияние на результат.

**Strict Math**. При включении этого пункта математические выражения не вычисляются, а записываются в исходном виде. Например, 10px + 10px останется исходным, а никак не значением 20px полученным в результате сложения.

**Strict Units**. При выборе этого пункта операции с разными единицами измерения приводят к ошибке. Если галочку убрать, то можно проценты соединять с пикселями, но в этом случае считается, что это одни и те же единицы. Выражение 10px + 10% даёт 20px, поскольку за основу берётся первая единица, т. е. пиксели, и проценты заменяются на них.

Сама компиляция происходит при нажатии на кнопку «Crunch File» в правом верхнем углу окна. Если включена опция «Open CSS after Crunching», то в новой вкладке откроется скомпилированный CSS. Или документ обновится, если вкладка уже открыта.