Переполнение

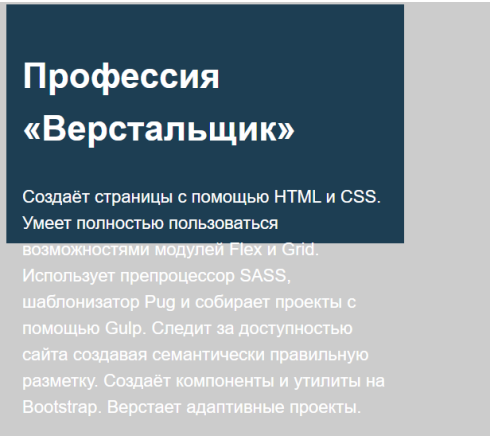
Переполнение контента внутри блока — распространённое явление при вёрстке контента. Переполнением считается ситуация, при которой контент внутри контейнера больше, чем размер самого контейнера. Распространённый случай такого поведения — использование контейнера с фиксированными значениями высоты и ширины.

Интересно: использование фиксированных значений высоты и ширины в большинстве случаев не является хорошей практикой. Так можно достаточно быстро сверстать блок по макету, но одновременно с этим отнимается возможность расширения функционала. Любой отход от изначального контента может привести к проблемам, связанным с выходом контента из контейнера. Используйте фиксированные значения высоты и ширины там, где это предполагается в дизайне или для создания специфичного функционала.

В качестве примера создадим блок с фиксированными значениями высоты и ширины. Внутри такого контейнера расположим текст так, чтобы он вышел за границы блока.

```
<section>
  <h1>Профессия «Верстальщик»</h1>
  <p>Создаёт страницы с помощью HTML и CSS. Умеет полностью пользоваться
возможностями модулей Flex и Grid. Использует препроцессор SASS, шаблонизатор
Pug и собирает проекты с помощью Gulp. Следит за доступностью сайта создавая
семантически правильную разметку. Создаёт компоненты и утилиты на Bootstrap.
Верстает адаптивные проекты.</p>
</section>
section {
  box-sizing: border-box;

  width: 500px;
  height: 300px;
  padding: 20px;

  color: #fff;
  font: 22px/1.5 sans-serif;

  background: #1d3e53;
}

h1 {
  font-size: 2em;
}
```

Результат



Большая часть описания профессии вышла за пределы блока. Браузеры в данном случае считают контент важнее, чем контейнер в котором он лежит и не скрывает текст. Это не лишено смысла, ведь главная часть любой страницы — её контент. Без него вся страница не будет иметь никакого смысла.

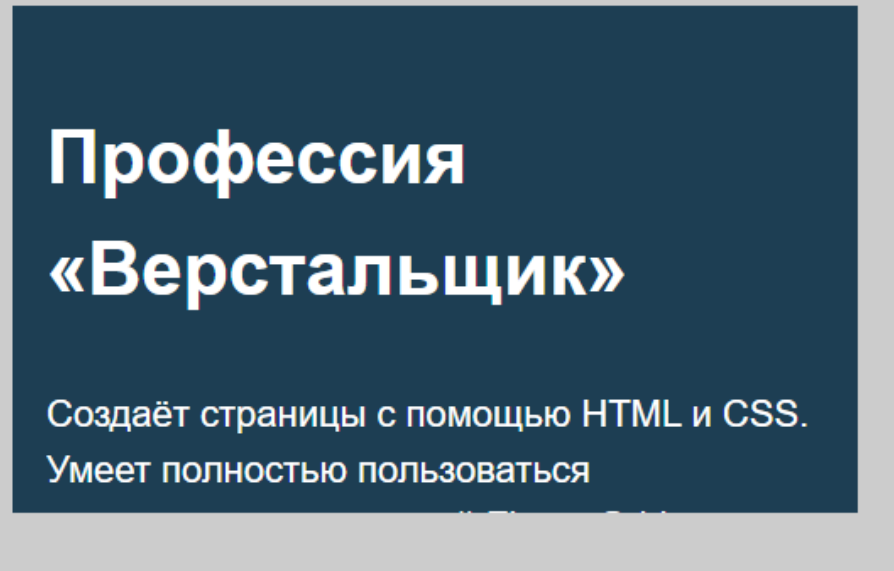
Такая ситуация называется переполнением и CSS позволяет управлять им. Для этого существует три свойства:

1. `overflow-x` — управление переполнением по горизонтали.
2. `overflow-y` — управление переполнением по вертикали.
3. `overflow` — сокращённая запись двух предыдущих свойств. Если указать внутри только одно значение, то оно применится к двум осям одновременно. Наиболее распространённый вариант использования.

По умолчанию свойство имеет значение `visible`, которое и указывает на то, что при переполнении контент должен отрисовываться вне своего родителя. В противовес `visible` есть значение `hidden`. Его задача обратна — скрыть контент, который выходит за пределы своего родителя. При этом доступ к такому контенту теряется. При использовании свойство `overflow` важно помнить, что это свойство не является наследуемым, поэтому его необходимо указывать у каждого блока, с которым происходит переполнение. В дальнейшем вы увидите примеры таких реализаций.

Распространённая ситуация при вёрстке блоков, которые должны находиться в HTML, но быть временно скрытыми. Например, при создании слайдеров, в которых все неактивные слайды находятся за пределами блока и скрыты с помощью свойства `overflow`.

Результат

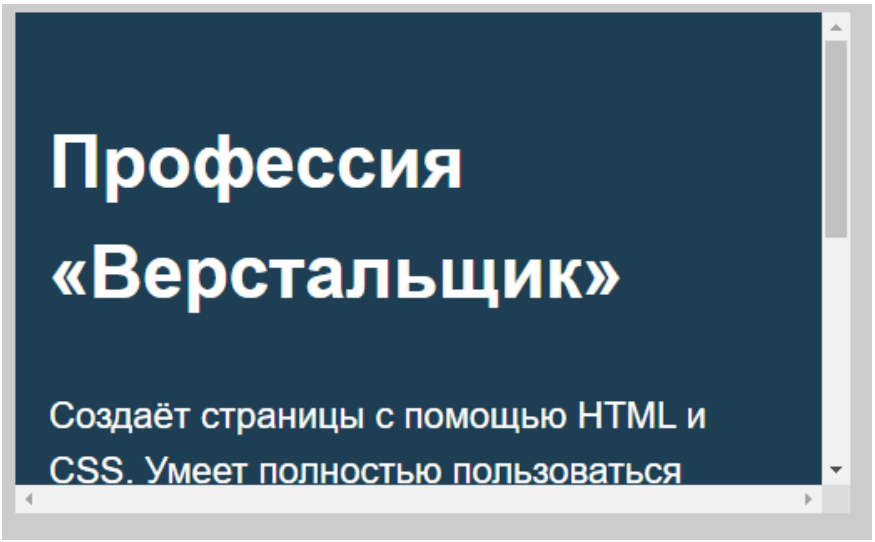


Хоть теперь вёрстка не «сломалась» от переполнения, но прочитать описание профессии невозможно. Не хватает какой-нибудь полосы прокрутки внутри блока. Свойство `overflow` позволяет добавить полосу прокрутки в такой блок. Для этого может использоваться два значения:

- `scroll`
- `auto`

В чём разница между ними? Посмотрим на примере взяв вначале значение `scroll`. Установим для секции новое значение свойства `overflow`.

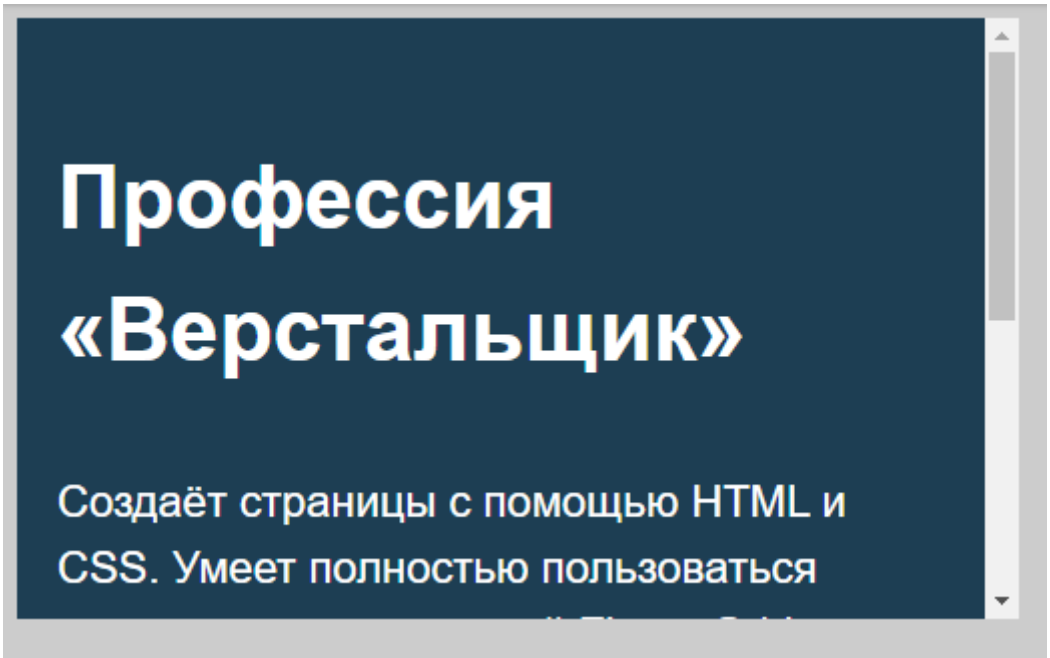
Результат



Теперь мы можем прокрутить контент внутри блока и наконец прочитать описание всей профессии. Но прокрутка появилась не только по вертикали, но и по горизонтали. При этом она недоступна, так как в этом направлении нет переполнения контента. Во-первых, это «портит» дизайн, а во-вторых отнимает место внутри блока. Если такое поведение явно не обозначено в макете, то

стоит добавить полосу прокрутки только для того направления, где возникает переполнение контента. Это возможно с помощью значения `auto`. В этом случае браузер следит за тем, где возникло переполнение и добавляет полосу прокрутку именно для этого направления.

Результат



Важно: используйте свойство `overflow` с осторожностью. Велик соблазн использовать его в случае быстрой вёрстки, когда при выходе макета за пределы экрана выставляют следующий CSS код:

```
body {
  overflow-x: hidden;
}
```

Это действительно решит проблему с горизонтальной прокруткой, но может и обрезать важную часть контента. Потратьте чуть больше времени, но локализуйте проблему и решите именно её. Это сделает вашу вёрстку понятнее и проще.

Переполнение текста

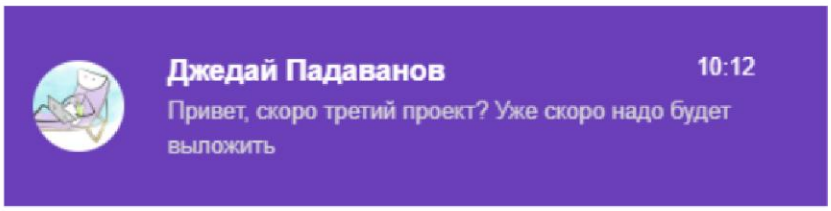
Не всегда требуется работать с переполнением только в рамках свойства overflow. Бывают ситуации, когда нужно точечно работать с переполнением контента, а не со всем блоком сразу. В качестве примера используем вёрстку превью сообщения чата

HTML

```html
<div class="contact">
  <img class="avatar"
src="https://assets.codepen.io/1425525/avatar_
2.jpg" alt="Джедай Падаванов">
  <div class="contact-body">
    <div class="contact-title">
      <div class="contact-name">Джедай
Падаванов</div>
      <div class="contact-time">10:12</div>
    </div>
    <p class="contact-message">Привет, скоро
третий проект? Уже скоро надо будет
выложить</p>
  </div>
</div>
```

CSS

```css
body {
  margin: 0;
  padding: 50px;
  font: 16px/1.5 sans-serif;
  color: #fff;
}

.avatar {
  width: 50px;
  border-radius: 50%;
}

.contact {
  display: flex;
  align-items: center;
  padding: 0 1rem;
  background-color: #673ab7;
  width: 420px;
}

.contact-body {
  padding: 1.5rem;
}

.contact-title {
  display: flex;
```



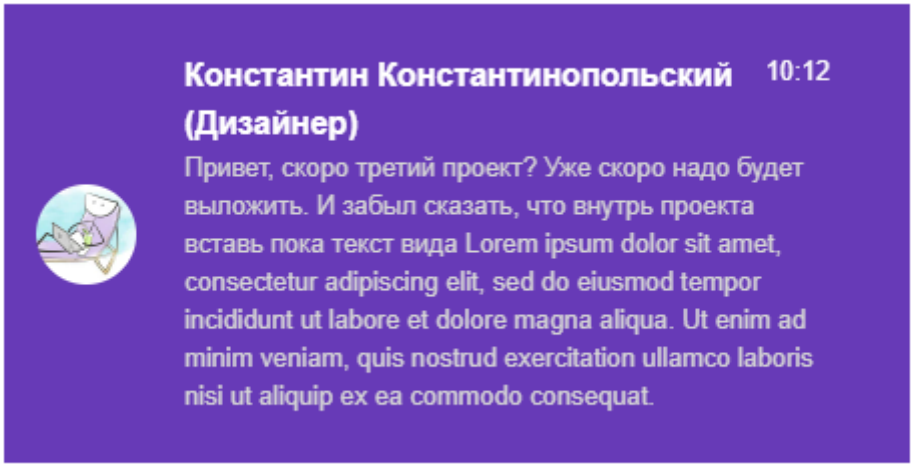
Перед продолжением изучите вёрстку этого примера. Вы можете обнаружить пару новых свойств, которые относятся к модулю *CSS Flexible Box Layout*. С этим модулем вы познакомитесь в курсе CSS: Flex.

Превью выглядит неплохо с текущим количеством текста, но если его станет больше, то вся вёрстка может развалиться.

Интересно: одна из задач хорошего верстальщика — предусмотреть различные варианты контента внутри блока. Попробуйте в полях, где должно быть имя выставлять длинные последовательности. Среди дизайнеров хорошей

практикой является тестирование макета на «Константине Константинопольском»

Увеличим количество контента внутри компонента с превью сообщения.



Не очень похоже на превью. Теперь это полноценное сообщение, которое отображается пользователю. Если таких сообщений будет десяток или сотня, то очень легко плюнуть на всё и уйти с сайта, чем листать такое количество контента. В идеальной ситуации стоит отобразить только по одной строчке от имени и сообщения. Это можно сделать с помощью свойства `white-space` со значением `nowrap`. Такая конструкция запретит перенос текста по строкам внутри блока. Если её добавить, то весь текст внутри блока с именем и блока с сообщением расположится в одну строку, что приведёт к переполнению, но решит задачу. Ведь мы уже умеем работать с переполнением. Установим это свойство для селекторов `.contact-name` и `.contact-message`.

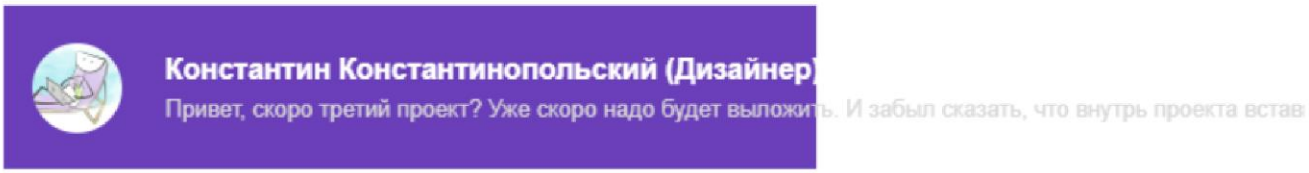
```
.contact-name {
  font-weight: 700;
  white-space: nowrap;
}

.contact-message {
  margin: 0;

  color: #d6d6d6;
  font-size: 80%;
  white-space: nowrap;
}
```

HTML

```
<div class="contact">
  <img class="avatar"
src="https://assets.codepen.io/1425525/avatar
_2.jpg" alt="Джедай Падаванов">
  <div class="contact-body">
    <div class="contact-title">
      <div class="contact-name">Константин
Константинопольский (Дизайнер)</div>
      <div class="contact-time">10:12</div>
    </div>
    <p class="contact-message">Привет, скоро
третий проект? Уже скоро надо будет выложить.
И забыл сказать, что внутрь проекта вставь
пока текст вида Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip
ex ea commodo consequat. </p>
  </div>
</div>
```

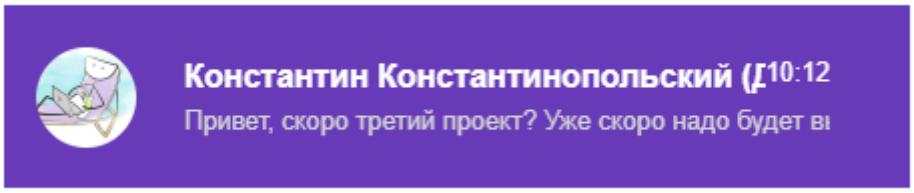
CSS

```
body {
  margin: 0;
  padding: 50px;
  font: 16px/1.5 sans-serif;
  color: #fff;
}

.avatar {
  width: 50px;
  border-radius: 50%;
}

.contact {
  display: flex;
  align-items: center;
  padding: 0 1rem;
  background-color: #673ab7;
  width: 420px;
}

.contact-body {
  padding: 1.5rem;
}

.contact-title {
  display: flex;
```



Не очень красиво получилось. Стоит обрезать контент, который не помещается в рамки контейнера. Добавим свойство `overflow-x` для селекторов, к которым было добавлено правило `white-space`. Помимо этого свойство необходимо добавить для всего контейнера, внутри которого и содержатся элементы с именем и сообщением

```css
.contact-body {
  padding: 1.5rem;
  overflow-x: hidden;
}

.contact-name {
  overflow-x: hidden;

  font-weight: 700;
  white-space: nowrap;
}

.contact-message {
  margin: 0;
  overflow-x: hidden;

  color: #d6d6d6;
  font-size: 80%;
  white-space: nowrap;
}
```



Почему понадобилось столько свойств `overflow`? Дело в отображении HTML. По своей сути браузер просто считывает вёрстку сверху вниз. Если взглянуть на этот компонент с точки зрения браузера, то получится следующая ситуация:

1. Отрисовываем блок `.contact-body` и ограничиваем его по ширине.
2. Отрисовываем блок `.contact-name`. Внутри него содержится длинный контент, который запрещено переносить согласно правилу `white-space`. Ширина блока больше, чем ширина родителя. По умолчанию отрисовываем контент за пределами контейнера.
3. Повторяем действия из пункта 2 для блока `.contact-message`.

Добавляя в каждый из трёх блоков свойство `overflow` браузер последовательно работает с переполнением контента. Если упустить свойство у блока `.contact-body`, то ширина блоков `.contact-name` и `.contact-message` не будет ограничена и использование `overflow` никак на них не повлияет.

Можно сказать, что вёрстка закончена, но сейчас отсутствуют отступы между участками текста по горизонтали. Например, имя пользователя и время

сообщения почти слиплись. Можно добавить отступ, но есть и другой путь — работа с переполнением контента внутри строки.

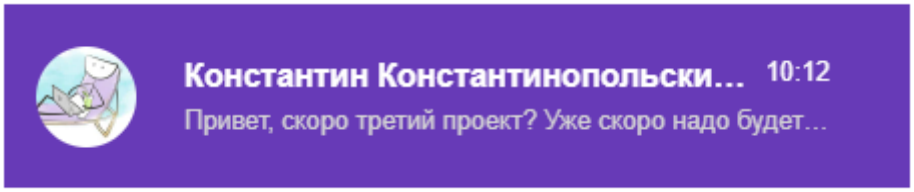
Для указания браузеру, что нужно делать при переполнении контента внутри строки используется правило `text-overflow`. Оно может принимать всего два значения:

- `clip` — значение по умолчанию. Текст «режется» в том месте, где достиг края блока. Именно это поведение можно заметить в примере выше.
- `ellipsis` — вместо грубого среза строки добавляется многоточие. Это визуально показывает пользователю, что строка не закончена.

Добавим многоточие в блоки с именем пользователя и текстом сообщения.

Важно: для работы свойства `text-overflow` необходимо наличие свойства `overflow` со значением, отличным от `visible`.

```
.contact-name {
  padding-right: 10px; /* добавим правый внутренний отступ для красоты */
  overflow-x: hidden;

  font-weight: 700;
  white-space: nowrap;
  text-overflow: ellipsis;
}

.contact-message {
  margin: 0;
  overflow-x: hidden;

  color: #d6d6d6;
  font-size: 80%;
  white-space: nowrap;
  text-overflow: ellipsis;
}
```



Задание

Используя новые свойства создайте блок с превью новостей. В нём будет три колонки шириной 500 пикселей. Блок содержит данные:

- Дата
- Название новости
- Краткое описание

Дата и название новости должно выводиться полностью с возможным переносом строк. Краткое описание должно выводиться в одну строку. Если места не хватает, то в конце описание ставится многоточие.