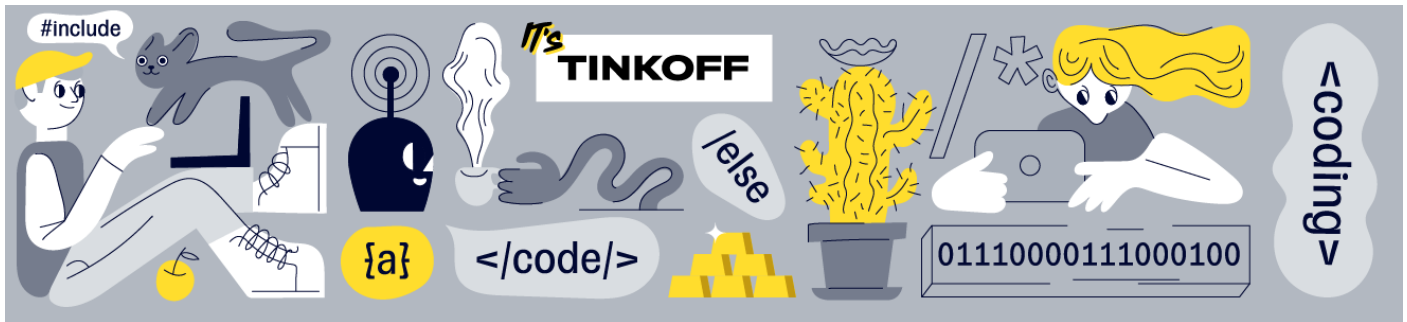


КАК СТАТЬ АВТОРОМ



Неделя тестировщиков

Обновляем рейтинг IT-бре...

**161.31**  
Рейтинг

## TINKOFF

IT's Tinkoff — просто о сложном



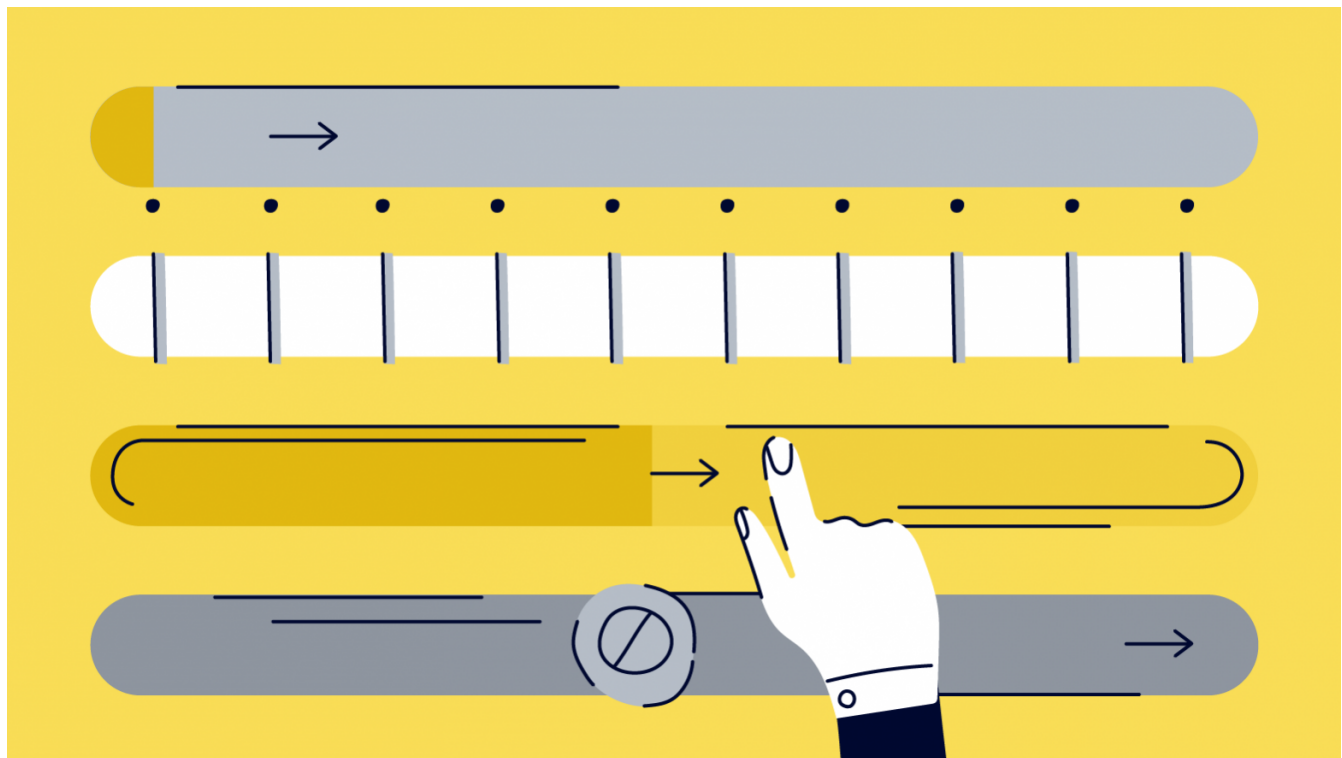
nsbarsukov 23 мая в 17:57

## Три слоя градиента одного слайдера

Блог компании TINKOFF, JavaScript\*, Accessibility\*, Angular\*, TypeScript\*

Мы в Тинькофф разрабатываем [библиотеку Taiga UI](#): в ней сотни полезных компонентов, директив и сервисов. При разработке нам важно уделять внимание вопросу поддерживаемости кода, поэтому мы стараемся не писать лишнего кода и искать решения, работающие на нативных рельсах браузеров.

В статье поговорим про одно из таких решений — написание [Angular-компонента Slider](#) с помощью встроенных инструментов браузеров и с минимальным использованием Javascript. Расскажу про доступность, интересное решение с многослойным градиентом, немного CSS-хитростей и даже чуть-чуть про Change Detection в Angular.



## Когда задача сложнее, чем может показаться

Нужно разработать элемент интерфейса, с которым пользователь сможет выбирать число из упорядоченного диапазона. Инпут состоит из двух частей:

- **бегунок, или thumb** — активный элемент инпута, который пользователь перемещает, уменьшая или увеличивая выбранное значение;
- **дорожка, или track** — пространство, вдоль которого двигают бегунок.



Встроенный Slider в Chrome

Изначально задача кажется простой. Следи, на каком расстоянии от начала дорожки произошло событие `MouseMove/TouchMove`, сдвигай туда бегунок и выдавай числовое значение пользователю.

Кажется, что на этом можно закончить и не посвящать этому целую статью. И тут в игру вступают множественные но, о которых не стоит забывать. Нам поможет документация W3C.

W3C — всемирная организация, которая разрабатывает стандарты для построения веб-

приложений. Ее цель — сделать веб доступным и понятным для каждого.

Звучит очень абстрактно, поэтому приведу пример. Представьте, что вы заходите на любой сайт и открываете там попап. Какую кнопку клавиатуры вам хочется интуитивно нажать, чтобы закрыть это модальное окно? Esc. Если разработчики сайта добросовестно подошли к своей работе, то, скорее всего, попап действительно закроется при нажатии на эту кнопку.

Это и есть стандарт. Распространением похожих стандартов и занимается W3C. Посмотрим, что нам говорит [W3C про Slider](#):

**Клавиатурная навигация.** Нажатие клавиш со стрелками вверх/вниз и вправо/влево должно увеличивать/уменьшать значение слайдера на один шаг. Нажатие клавиш Home/End должно перемещать бегунок в начало или конец слайдера. А еще есть рекомендации по поведению клавиш Page Up / Page Down: увеличение/уменьшение значения слайдера на величину большую, чем изменение на шаг при нажатии клавиш со стрелками. При создании своего слайдера придется написать обработчики под восемь кнопок, каждая из которых будет отвечать за определенный результат.

**Доступность.** Сайтом могут пользоваться люди с ограниченными физическими возможностями. Они обычно используют вспомогательные технологии. Например, пользователь может испытывать трудности в использовании клавиатуры и мыши. В таких случаях работают технологии, распознающие голосовые команды. Пользователь проговаривает компьютеру, что нужно сделать на экране. Для корректной работы таких вспомогательных технологий сайт и должен быть грамотно размечен.

Какие требования по доступности предъявляются к слайдеру:

- Бегунок слайдера обязан иметь `role="slider"` .
- Для слайдера должен быть задан `aria-valuenow` , равный текущему числовому значению слайдера.
- Также слайдер должен иметь свойства `aria-valuemin` / `aria-valuemax` , обозначающие минимально/максимально возможное значение слайдера.
- Если числовое значение слайдера недостаточно просто в понимании, то он дополнительно должен иметь `aria-valuetext` . Простой пример из документации: если вы по каким-то причинам решили использовать слайдер для выбора дней недели (от первого до седьмого), то обязательно сопроводите выбранное значение текстовым описанием.
- А еще стоит не забывать про `aria-label` .

Соблюдение всех требований к доступности должно прибавить дополнительных хлопот. А учитывая, что мы проектируем многократно переиспользуемый UI-Kit-компонент, то часть вышеперечисленных свойств придется получать как input-пропсы компонентов.

**У слайдера должна быть возможность конфигурирования шага слайдера.**

Например, разработчик может разрешить пользователю выбирать только целочисленные величины. А может разрешить выбирать значение с точностью до 0,0001.

Вот почему все не так просто, как могло показаться с первого взгляда. Но за вас все это может сделать браузер.

Существует встроенный слайдер `<input type="range" />`, который обеспечивает соблюдение всех вышеперечисленных стандартов. Думаю, никого не удивит, что такое встроенное решение не очень красиво, да еще и в каждом браузере выглядит по-разному. Поэтому покажу, как кастомизировать встроенный слайдер под свои нужды, чтобы он выглядел одинаково во всех современных браузерах.

Компонент спроектируем под Angular. Но JS-кода будет настолько мало, что такое решение вы без труда перенесете под ваш любимый фреймворк.

## Анатомия `<input type="range" />`

Если включить в настройках инструмента разработчика в браузере отображение shadow DOM, то можно обнаружить, что встроенный тег `<input type="range" />` внутри имеет еще теги, которые мы не добавляли. При этом внутренняя структура может отличаться в разных браузерах. Где-то будет плоская структура с тремя `div` - контейнерами, а где-то — дополнительная вложенность.

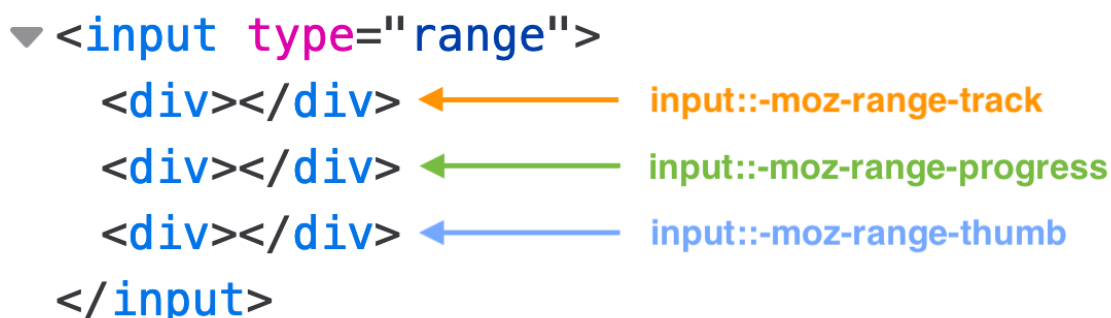
Каждый движок браузера по-разному подошел к написанию своего слайдера, поэтому все они имеют разную структуру. Но суть одинакова: всегда есть контейнер с дорожкой слайдера и контейнер с бегунком. Просто нужно знать, как обращаться к этим вложенными тегами.

Обратиться к HTML-тегам внутри `<input />`, используя CSS-селектор, опирающийся на структуру DOM, не получится (например, `input[type="range"] > div`). Вся разметка слайдера инкапсулирована внутри shadow DOM и недоступна извне. Но браузеры оставили API (псевдоклассы), как все-таки можно достучаться до нужных тегов.

Все браузеры, использующие WebKit/Blink-движок (например, Safari и Chrome), имеют такую анатомию:



А Mozilla дала больше свободы кастомизации: здесь помимо дорожки и бегунка появляется дополнительный тег, отвечающий за шкалу заполненности слайдера (progress).



Это все, что нужно знать, чтобы написать свой простенький слайдер. Однако если гибкости нужно больше, чем просто покрасить вложенные теги в разные цвета, то могут быть трудности. Расскажу про возможные проблемы и как с ними бороться.

## Пишем базовый вариант

Для начала создадим основу нашего компонента, опираясь на информацию, полученную ранее.

Зададим [less-переменные](#) для самодокументируемости кода:

```

@thumb-diameter: 1rem;
@thumb-color: orange;
  
```

```
@track-height: 0.125rem;  
@track-color: lightgray;  
@track-border-radius: 0.625rem;
```

Создадим [less-миксин](#), который отвечает за кастомизацию дорожки слайдера. Пока он будет простым, но в следующих главах мы его значительно расширим.

```
.customize-track() {  
  height: @track-height;  
  border-radius: @track-border-radius;  
  background-color: @track-color;  
}
```

Далее создаем миксин для стилизации бегунка слайдера:

```
.customize-thumb() {  
  /* Clear browser default customization */  
  appearance: none;  
  border: none;  
  /* _____ */  
  
  background-color: @thumb-color;  
  height: @thumb-diameter;  
  width: @thumb-diameter;  
  border-radius: 50%;  
  
  :not(:disabled)& {  
    cursor: ew-resize;  
  }  
  
  :focus-visible& {  
    box-shadow: 0 0 2px inset rgba(51, 51, 51, 0.64);  
  }  
}
```

Если не совсем понятно, как работает символ `&`, советую прочитать [раздел документации по less](#).

Наконец, создадим вспомогательный миксин для webkit-браузеров. Webkit-браузеры не умеют вертикально центрировать бегунок слайдера, если он больше/меньше высоты дорожки слайдера. Множественные способы вертикального центрирования контейнера здесь не помогут, но проблема решается так:

```
/* Vertically centers thumb on the track (webkit only) */  
.vertically-align-thumb() {  
  margin-top: (@track-height / 2) - (@thumb-diameter / 2);  
}
```

Теперь можно создавать Angular-компонент. Наш компонент будет без дополнительной верстки, но с атрибутивным селектором с нативным элементом

`<input type="range" />`. Использование атрибутивных компонентов — хорошая практика, когда мы расширяем поведение [нативных элементов](#). В нашей библиотеке Taiga UI можно найти много таких примеров. Об одном из них [я уже писал](#).

Работать компонент будет на OnPush-стратегии. Если вы с ней не знакомы или не пользуетесь ею, рекомендую статью моего коллеги:

#### OnPush — ваш новый Default

В Angular есть два режима change detection: Default и OnPush. В этой статье мы разберем, как можно с...

habr.com



Содержание Typescript-файла будет такое:

```
import {
  ChangeDetectionStrategy,
  Component,
} from '@angular/core';

@Component({
  selector: 'input[type=range][tuiSlider]',
  template: ``,
  styleUrls: ['./slider.style.less'],
  changeDetection: ChangeDetectionStrategy.OnPush,
})
export class SliderComponent {}
```

Селектор нашего компонента заставляет пользователя обязательно прописывать тип инпута (type="range"). Есть большой соблазн упростить жизнь пользователям и самим накидывать это статичное свойство через `host` в метаданных компонента. Но, к сожалению, это ломает логику встроенного [controlValueAccessor](#) от команды [Angular](#), который применяется через CSS-селектор `input[type=range]`.

В Less-файле активно применяем ранее созданные миксины на `:host`-селектор (напоминаю, в данном случае это и будет элемент `<input type="range" />`).

```
:host {
  /* Clear browser default customization */
  appearance: none;
  background-color: transparent;
  outline: none;
  /* _____ */

  display: block;
  cursor: pointer;
  width: 100%;
  height: @track-height;
  /* To catch click events nearby THIN input's track */
  padding: 0.4375rem 0;

  &:disabled {
    opacity: 0.56;
```



```
    cursor: auto;
}

&::-webkit-slider-runnable-track {
    .customize-track();
}

&::-moz-range-track {
    .customize-track();
}

&::-webkit-slider-thumb {
    .customize-thumb();
    .vertically-align-thumb();
}

&::-moz-range-thumb {
    .customize-thumb();
}

&::-moz-range-progress {
    background: @thumb-color;
    border-radius: @track-border-radius 0 0 @track-border-radius;
}
}
```

Готово! Получили базовое кроссбраузерное решение по кастомизации встроенного слайдера. Но проблемы на этом не закончились.

## Проблема кастомизации шкалы заполненности в Webkit

Mozilla позаботилась о желании разработчика кастомизировать шкалу заполненности индикатора — псевдокласс `moz-range-progress`. Но увы, такой функциональности нет для WebKit/Blink-браузеров. Проблему решают [линейные градиенты](#).

Ранее мы просто красили всю дорожку слайдера в серый цвет. А теперь будем красить часть шкалы в цвет заполненности, а оставшуюся часть оставлять серой.

Дополняем в класс компонента немного простой математики, которая посчитает, на сколько процентов заполнился слайдер, и запишет это значение [в CSS-переменную](#) `--slider-fill-percentage` :

```
// ...
export class SliderComponent {
  get min(): number {
    return Number(this.elementRef.nativeElement.min) || 0;
  }

  get max(): number {
    return Number(this.elementRef.nativeElement.max) || 100;
  }

  get value(): number {
    return Number(this.elementRef.nativeElement.value) || 0;
  }

  @HostBinding('style.--slider-fill-percentage.%')
  get valuePercentage(): number {
    return (100 * (this.value - this.min)) / (this.max - this.min) || 0;
  }

  constructor(
    @Inject(ElementRef)
    private readonly elementRef: ElementRef<HTMLInputElement>
  ) {}
}
```

Модернизируем прошлый миксин `customize-track` :

```
.customize-track(@progress-filling: false) {
  height: @track-height;
  border-radius: @track-border-radius;

  @filling-progress-gradient: linear-gradient(
    to right,
    @thumb-color 0 var(--slider-fill-percentage),
    @track-color var(--slider-fill-percentage) 100%
  );
}
```

```
& when (@progress-filling = true) {  
  background: @filling-progress-gradient;  
}  
  
& when (@progress-filling = false) {  
  background: @track-color;  
}  
}
```

Пробуем полученный компонент в действии:

```
<input type="range" tuiSlider value="30" />
```

Мы используем `changeDetectionStrategy.OnPush`, поэтому ожидаемо, что слайдер не сделает пересчет значения CSS-переменной `--slider-fill-percentage` в процессе перетаскивания бегунка слайдера. Нам нужно подсказать Angular, когда запускать проверку изменений. Добиться этого просто: проверка изменений должна происходить при срабатывании [InputEvent](#). Вешаем пустой обработчик события в метаданные компонента:

```
@Component({  
  selector: 'input[type=range][tuiSlider]',  
  template: '',  
  styleUrls: ['./slider.style.less'],  
  host: {  
    '(input)': '0',  
  },  
  changeDetection: ChangeDetectionStrategy.OnPush,  
})  
export class SliderComponent {  
  // ...  
}
```

Теперь при перетаскивании бегунка запускается пересчет значения CSS-переменной `--slider-fill-percentage`.

Одной проблемой меньше, и с этого момента наш слайдер полностью идентичен во

всех современных браузерах.

## Добавляем ticks через многослойный градиент

У нас есть рабочее решение. Представим, что захотели использовать слайдер со следующим сочетанием нативных атрибутов:

```
<input type="range" tuiSlider min="0" max="100" step="20"/>
```

Такая комбинация означает, что пользователь может сдвигать бегунок только на значение, кратное 20. То есть пользователь может выбрать только пять значений в диапазоне, или, другими словами, слайдер состоит из пяти сегментов. Популярная практика для такого кейса — выделение сегментов визуально через отметки/черточки на дорожке слайдера.



Чтобы этого добиться, можно было бы добавить дополнительные элементы/псевдоэлементы внутри инпута. Но такое поведение поддерживают только webkit-браузеры, потому что это не общепринятый стандарт. Еще вариант — попробовать реализовать отметки вновь через градиент, но в прошлой главе на background -дорожки уже накинули градиент (то есть место уже как будто занято).

Выход есть: градиенты можно накладывать друг на друга в несколько слоев. С помощью [repeating-linear-gradient](#) создадим новый слой и нанесем его поверх предыдущего.

Но для начала в код компонента добавим:

```
// ...
export class SliderComponent {
  @Input()
  segments = 1;

  //...

  @HostBinding('style.--slider-segment-width.%')
  get segmentWidth(): number {
    return 100 / Math.max(1, this.segments);
  }
}
```

```
}  
}
```

Мы добавили для компонента инпут-проптерти `segments`, с помощью которой пользователь может задать количество визуальных сегментов, которое он хочет выделить на дорожке слайдера. А новый геттер в компоненте выполняет простое вычисление — считает, сколько процентов от общей длины должен занимать каждый сегмент, а результат сохраняет в CSS-переменную `--slider-segment-width`.

Осталось модернизировать миксин `.customize-track()`. Чтобы на один и тот же фон накладывать несколько слоев градиента, достаточно перечислить их через запятую в свойстве `background-image`. Например:

```
@top-layer: linear-gradient(...)  
@middle-layer: repeating-linear-gradient(...)  
@bottom-layer: linear-gradient(...)  
  
background-image: @top-layer, @middle-layer, @bottom-layer;
```

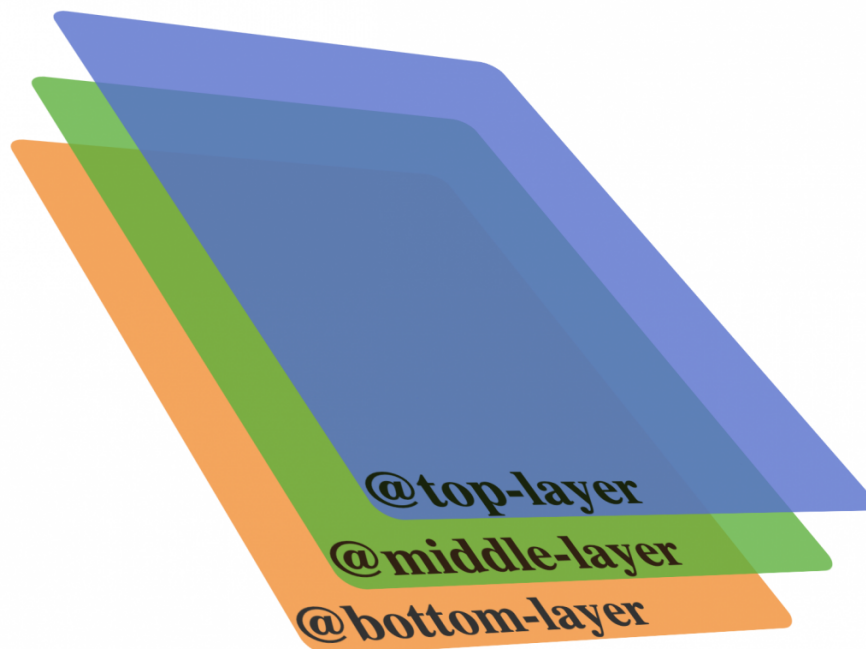


Иллюстрация примера, когда несколько слоев градиента накладываются друг на друга

Главное — не запутаться, в каком порядке слои накладываются друг на друга. Самый первый указанный градиент будет наложен поверх всех остальных, а самый последний станет нижним слоем фона. То есть наложение слоев друг на друга начинается с конца перечисления списка.

Все остальные свойства `background-*` имеют аналогичное поведение. Можно указать только один набор параметров для всех слоев, а можно перечислить параметры через запятую — в том же порядке, что и в свойстве `background-image`, задавая свои значения для каждого слоя.

Модернизированный код миксина `customize-track`:

```
@ticks-thickness: 0.25rem;
@tick-color: darkgrey;

.customize-track(@progress-filling: false) {
  height: @track-height;
  border-radius: @track-border-radius;

  @first-tick-offset: ((@thumb-diameter - @ticks-thickness) / 2);
  @ticks-background-size: calc(100% - @thumb-diameter);

  @hide-first-tick: linear-gradient(
    to right,
    @thumb-color 0px (@first-tick-offset + @ticks-thickness),
    transparent @ticks-thickness
  );

  @ticks-gradient: repeating-linear-gradient(
    to right,
    @tick-color 0 @ticks-thickness,
    transparent 0 var(--slider-segment-width)
  );

  @filling-progress-gradient: linear-gradient(
    to right,
    @thumb-color var(--slider-fill-percentage),
    transparent var(--slider-fill-percentage)
  );

  background-repeat: no-repeat;
  background-color: @track-color;

  & when (@progress-filling = true) {
    background-image:
      @hide-first-tick,
      @ticks-gradient,
      @filling-progress-gradient;
    background-position-x: 0, @first-tick-offset, 0;
```

```
    background-size: @ticks-background-size, @ticks-background-size, auto;
  }

  & when (@progress-filling = false) {
    background-image: @hide-first-tick, @ticks-gradient;
    background-position-x: 0, @first-tick-offset;
    background-size: @ticks-background-size;
  }
}
```

Важная особенность нативных слайдеров: при перемещении бегунка в самый старт/конец дорожки он касается конца дорожки не своим центром, а краешком. То есть бегунок слайдера никогда не выходит за визуальные границы дорожки слайдера. Поэтому самая первая отметка начинается не в самом начале дорожки, а со сдвигом, что позволяет ей быть в центре бегунка (когда тот находится в крайней левой позиции), а также корректно расставляет остальные отметки.

Помимо всех слоев у `background-image` мы также задали значение для `background-color`. Логика здесь следующая: `background-color` выступает в роли самого последнего, запасного, слоя. Вся поверхность дорожки, которая никак не закрывается ни одним из слоев, окрасится в цвет значения этого свойства.

Вот и все. Такое решение — хороший пример того, что порой задачу можно решить без дополнительных вложенных HTML-тегов.

## Вместо заключения

Весь код статьи я собрал в одном [StackBlitz-примере](#). В библиотеке [Taiga UI](#) мы используем точно такое же решение с небольшими улучшениями. [На витрине компонента Slider](#) можно посмотреть примеры его использования.

Не всегда нужно изобретать свои решения — иногда есть уже готовая нативная альтернатива, которую нужно чуть-чуть доработать.

Отказываясь от готового решения, вы расширяете свою кодовую базу, увеличивая сложность поддержки. Не зря одна современная мудрость гласит: лучший код — тот, который не написан.

Буду рад вопросам и идеям в комментариях.

**Теги:** [slider](#), [range](#), [слайдер](#), [angular](#), [a11y](#)

**Хабы:** [Блог компании TINKOFF](#), [JavaScript](#), [Accessibility](#), [Angular](#), [TypeScript](#)

◆ +15

🔖 30

💬 10 +1



## Редакторский дайджест

Присылаем лучшие статьи раз в месяц





**TINKOFF**

IT's Tinkoff — просто о сложном



16

-1

Карма   Рейтинг

**Барсуков Никита** @nsbarsukov

Frontend developer

Комментарии 10



**fedorro**

24.05.2022 в 01:37

Зашел в примеры PdfViewer вместо просмотра, на всех трех кнопках, скачивает файл taiga.pdf, Firefox 100.0.2.

Но компоненты симпатичные, спасибо за то что делитесь!



0

Ответить







**nsbarsukov**  
24.05.2022 в 15:34

Привет!

Пока не могу понять, о какой проблеме идет речь (у меня на том же браузере все корректно отрабатывает)

Можно тебя попросить завести нам issue на github?

<https://github.com/Tinkoff/taiga-ui/issues>

И написать все подробно (девайс, ОС, браузер + приложить запись экрана).

Мы обязательно посмотрим и поправим!



0

Ответить



**fedorro**  
24.05.2022 в 15:36

Да, сделаю, спасибо!



+1

Ответить



**fedorro**  
05.06.2022 в 22:03

Написал, отправил, и тут нашло озарение - действие по умолчанию для данного типа файла было установлено "Загрузка" Возможно это можно обойти ли предупредить пользователя, но если нет - то можно закрывать. В комментарии я это написал, может кому поможет кто с этим же столкнется.



0

Ответить



**Torvald3d**  
24.05.2022 в 16:04

Из-за того, что вы меняете курсор только под бегунком - при перемещении бегунка курсор постоянно меняется. Firefox

<https://skr.sh/vE6yNeEB5V7?a>



+2

Ответить



**nsbarsukov**  
24.05.2022 в 16:21

Хорошее замечание, спасибо!


Нам нужно на `:host`-элемент слайдера докинуть:

```
&:active {  
  cursor: ew-resize;  
}
```

Пойду исправлять)

0 Ответить



 **anotherjohndoe**  
24.05.2022 в 16:25

мы стараемся не писать лишнего кода и искать решения, работающие на нативных рельсах браузеров

К слову, не подскажите — чего не хватило в нативной стилизации скроллбара?

Отличная статья, спасибо!

0 Ответить



 **nsbarsukov**  
25.05.2022 в 01:01

На момент написания кастомного скроллбара у нативного была плохая кроссбраузерность (в частности, у Firefox все было плохо с этим).

Также через нативную стилизацию нельзя сделать полноценно скроллбар поверх контента.

Кажется, что сейчас ситуация не сильно улучшилась. Но я не проверял, нужно как-то провести исследование этого вопроса)

0 Ответить



 **alekssamos**  
24.05.2022 в 20:11

О, доступность, это важно, уважаю.

Но у вас есть (на вебе) одна проблема.

При переходе по таргету из рекламы в социальных сетях на оформление новой карты, открывается форма, где словно сделано так:

```
<input type="text" role="button">
```

Соответственно, программы экранного доступа считают поле ввода кнопкой и не могут перейти в режим редактирования, ничего написать или отредактировать невозможно.

0 Ответить



 **nsbarsukov**  
25.05.2022 в 14:30

Спасибо за замечание!

Изучим проблему и исправим)

 0 Ответить

Только полноправные пользователи могут оставлять комментарии. [Войдите](#), пожалуйста.

## ПОХОЖИЕ ПУБЛИКАЦИИ

17 ноября 2021 в 18:12

**Div на div'e не сидит и div'ом не погоняет: пишем семантически верные индикаторы загрузки на Angular**

 +21 7.1K 39 0

7 октября 2021 в 16:15

**Как мутировать код в Angular-схематиках и не поседеть**

 +15 2.5K 20 0

20 сентября 2021 в 16:24

**Концепция контроллеров компонента в Angular: часть вторая**

 +21 5.1K 47 6 +6

## ВАКАНСИИ КОМПАНИИ «TINKOFF»

Java Разработчик (Тинькофф Инвестиции)

TINKOFF · Можно удаленно

Старший Java Разработчик (Тинькофф Инвестиции)

TINKOFF · Можно удаленно

Системный аналитик

TINKOFF · Можно удаленно

Разработчик iOS

TINKOFF · Можно удаленно

Android разработчик

TINKOFF · Можно удаленно

Больше вакансий на Хабр Карьере

## ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

вчера в 11:16

### Bash отладчик с поддержкой произвольных точек останова

 +44

 4.1K

 74

 5 +5

вчера в 14:40

### Новые нули дзета-функции

 +31

 4K

 14

 10 +10

вчера в 21:20

### Личный опыт выгорания

 +26

 8.3K

 31

 25 +25

вчера в 17:00

### DIY квантовые вычисления: как я начал собирать квантовые схемы

 +20

 8.3K

 41

 25 +25

вчера в 21:00

### Антиматерия и бариогенезис. Три причины, почему нет антивещества, но есть мы

 +15

 2.8K

 21

 1 +1

## ИНФОРМАЦИЯ

Дата основания

18 ноября 2005

Местоположение

Россия

Сайт

[www.tinkoff.ru](http://www.tinkoff.ru)

Численность

свыше 10 000 человек

Дата регистрации

26 октября 2011

## НОВОСТИ

---

Тинькофф запустил льготное рефинансирование ипотеки для семей с детьми по ставке 4,99%

29 июня

Тинькофф Касса компенсировала бизнесу 70 млн рублей комиссий за использование СБП

29 июня

Тинькофф Кредит Брокер запустил технологию онлайн-идентификации Self ID для оформления рассрочек и кредитов

29 июня

Тинькофф Кредит Брокер запустил технологию Self ID для оформления рассрочек и кредитов онлайн

29 июня

5% для всех: Тинькофф Путешествия запустили программу кэшбэка на отели для клиентов любых банков

28 июня

Тинькофф Инвестиции наградили победителей олимпиады «Высшая проба»

28 июня

Тинькофф упрощает подключение клиентов к программе лояльности платежной системы «Мир»

27 июня

Тинькофф Инвестиции вносят изменения в условия хранения ряда валют на брокерских счетах и ИИС

24 июня

Тинькофф Бизнес запустил систему кадрового электронного документооборота для компаний

24 июня

Тинькофф Инвестиции снижают тарифы по маржинальной торговле

22 июня

## ССЫЛКИ

---

[Наши митапы](#)

[o.tinkoff.ru](https://o.tinkoff.ru)

[IT-вакансии](#)

[o.tinkoff.ru](https://o.tinkoff.ru)

[Карьера в Тинькофф](#)

[o.tinkoff.ru](https://o.tinkoff.ru)

## ПРИЛОЖЕНИЯ

---



### Тинькофф Банк в iPhone

С помощью бесплатного мобильного приложения вы можете совершать большинство операций без комиссии

[iOS](#)



### Тинькофф Банк в Android

С помощью бесплатного мобильного приложения вы можете совершать большинство операций без комиссии

[Android](#)

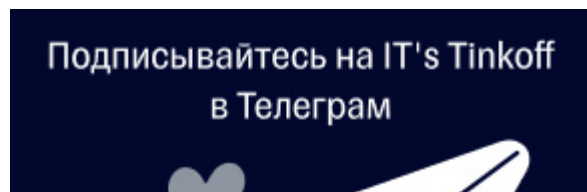


### Тинькофф Банк в Windows Phone

С помощью бесплатного мобильного приложения вы можете совершать большинство операций без комиссии

[Windows Phone](#)

#### ВИДЖЕТ



#### Ваш аккаунт

[Войти](#)

[Регистрация](#)

#### Разделы

[Публикации](#)

[Новости](#)

[Хабы](#)

[Компании](#)

[Авторы](#)

[Песочница](#)

#### Информация

[Устройство сайта](#)

[Для авторов](#)

[Для компаний](#)

[Документы](#)

[Соглашение](#)

[Конфиденциальность](#)

#### Услуги

[Корпоративный блог](#)

[Медийная реклама](#)

[Нативные проекты](#)

[Образовательные](#)

[программы](#)

[Мегапроекты](#)



[Настройка языка](#)

[Техническая поддержка](#)

[Вернуться на старую версию](#)

© 2006–2022, Habr

---

10 июня в 17:25

### Использование Gatling. Тестирование AMQP

 1.1K    0

---

3 июня в 18:27

### Как мы подходим к поддержке ML-моделей в синтезе речи

 1.3K    2 

---

25 мая в 15:06

### Как найти тему для выступления или статьи

 3.2K    1 