План занятия

1. Задача
2. Material Design
3. Размеры
4. Компоненты
5. Цвета
6. Стили
7. Итоги

ЗАДАЧА

ЗАДАЧА

Мы научились отображать коллекцию элементов и осуществлять с ними CRUD-операции:

- создание постов
- получение постов (уже сделали)
- обновление (сделали частично - лайк)
- удаление

Самое время заняться визуальным оформлением нашего приложения.

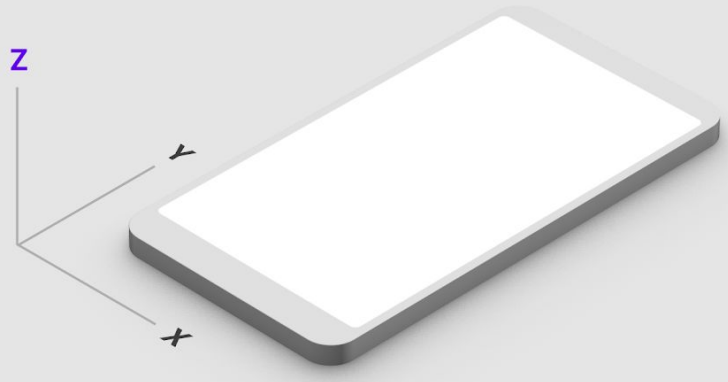
MATERIAL DESIGN

MATERIAL DESIGN

Material Design (или просто Material) — система руководств, готовых компонентов и инструментов, предназначенных для создания дизайна пользовательских интерфейсов (UI).

СИСТЕМА КООРДИНАТ

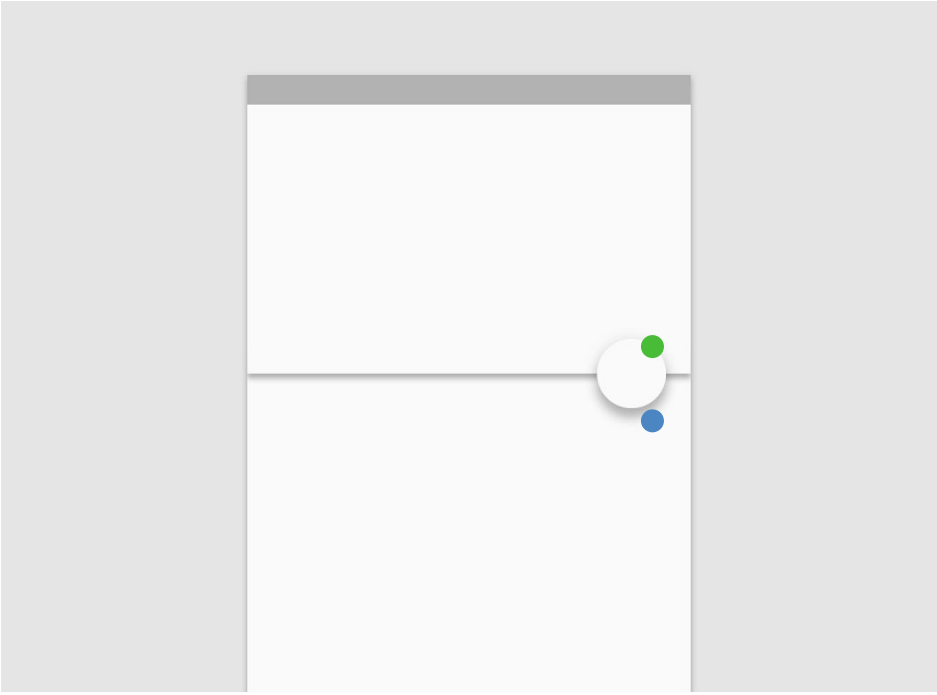
В рамках интерфейса у нас есть 3D система координат:



Благодаря координате Z, мы можем создавать у пользователя ощущение наложения компонентов.

КЛЮЧЕВАЯ ИДЕЯ

Проводится аналогия с реальным миром, в котором объекты могут располагаться относительно друг друга на разных или одинаковых уровнях, но не пересекать друг друга.

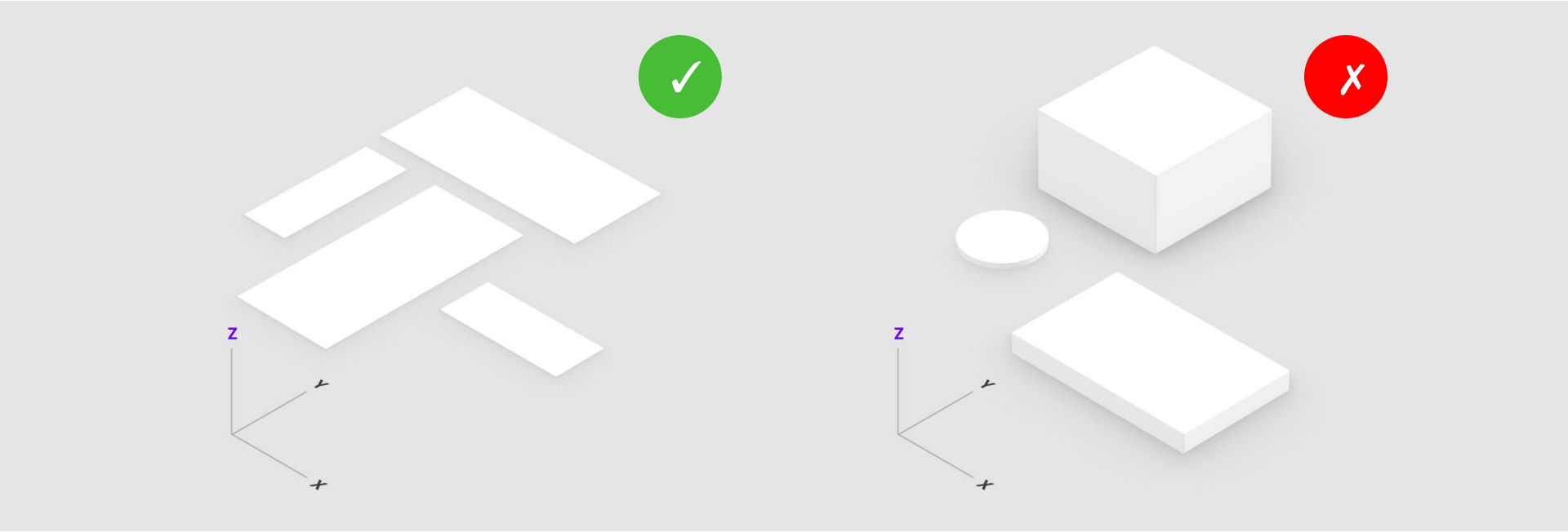


Уровень элемента визуально определяется по:

- наложению элементов (тот, что выше, закрывает тот, что ниже)
- размеру тени, отбрасываемой элементом

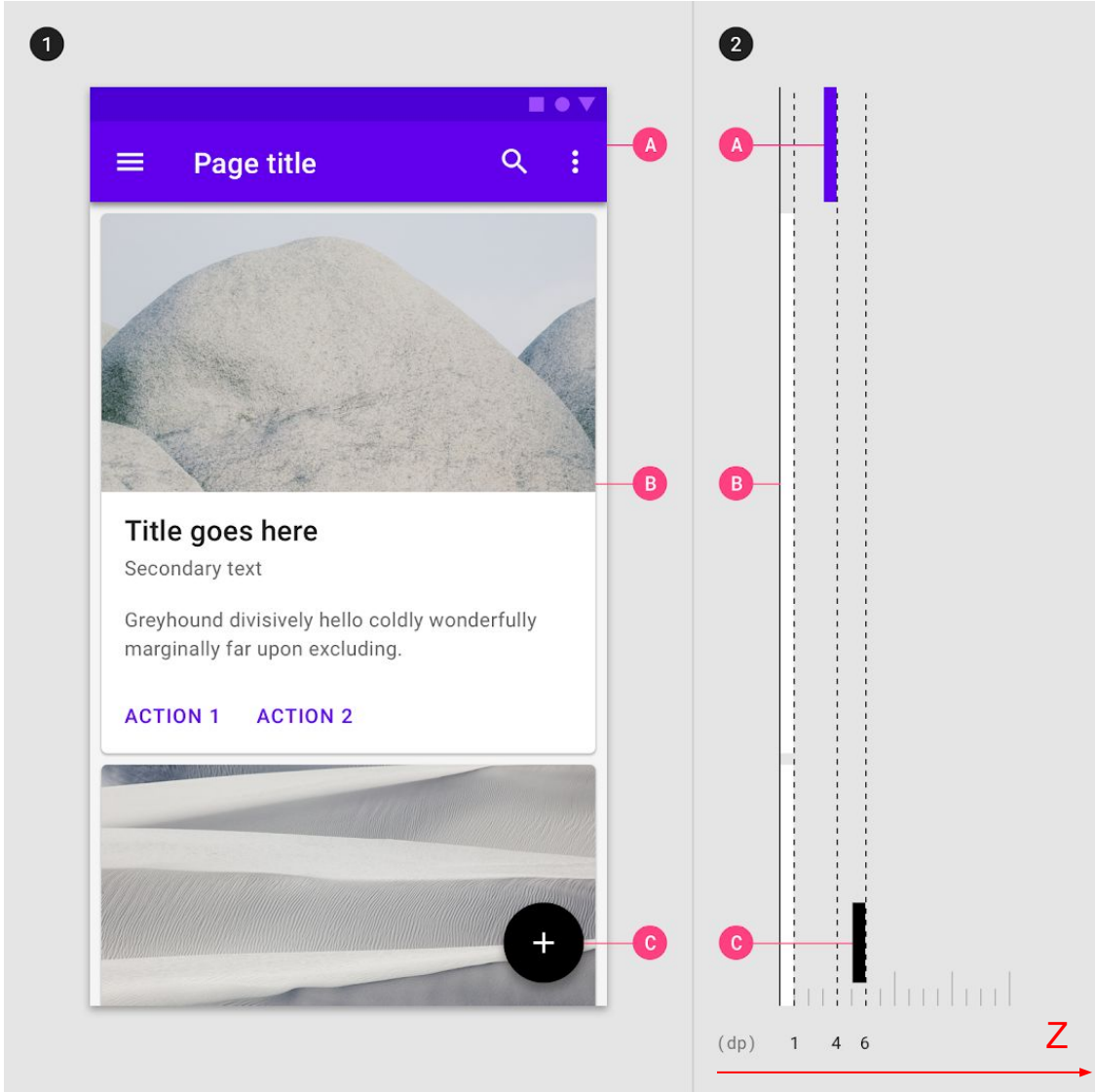
SURFACES

Элементы представляют из себя плоские поверхности (по аналогии с листами бумаги или тканью):



КЛЮЧЕВЫЕ ЭЛЕМЕНТЫ

- Surfaces — поверхности.
- Elevation — «уровень» поверхности по оси Z.
- Shadow — тень, отбрасываемая поверхностью.



MDC

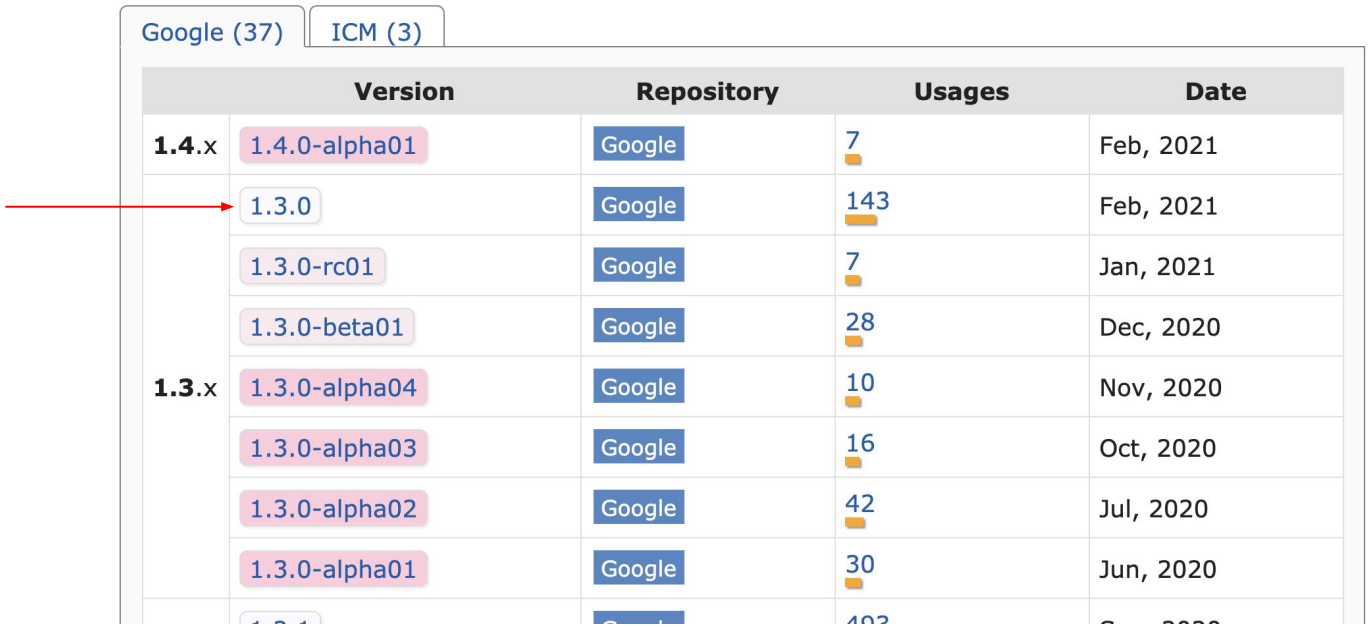
Для удобной реализации Material в Android предоставляется библиотека Material Design Components (MDC), содержащая реализации готовых компонентов.

ПОДКЛЮЧЕНИЕ

build.gradle:

implementation "com.google.android.material:material:$mdc_version"

На момент написания лекции версия: 1.3.0



Google (37) ICM (3)

	Version	Repository	Usages	Date
1.4.x	1.4.0-alpha01	Google	7	Feb, 2021
	1.3.0	Google	143	Feb, 2021
	1.3.0-rc01	Google	7	Jan, 2021
	1.3.0-beta01	Google	28	Dec, 2020
1.3.x	1.3.0-alpha04	Google	10	Nov, 2020
	1.3.0-alpha03	Google	16	Oct, 2020
	1.3.0-alpha02	Google	42	Jul, 2020
	1.3.0-alpha01	Google	30	Jun, 2020

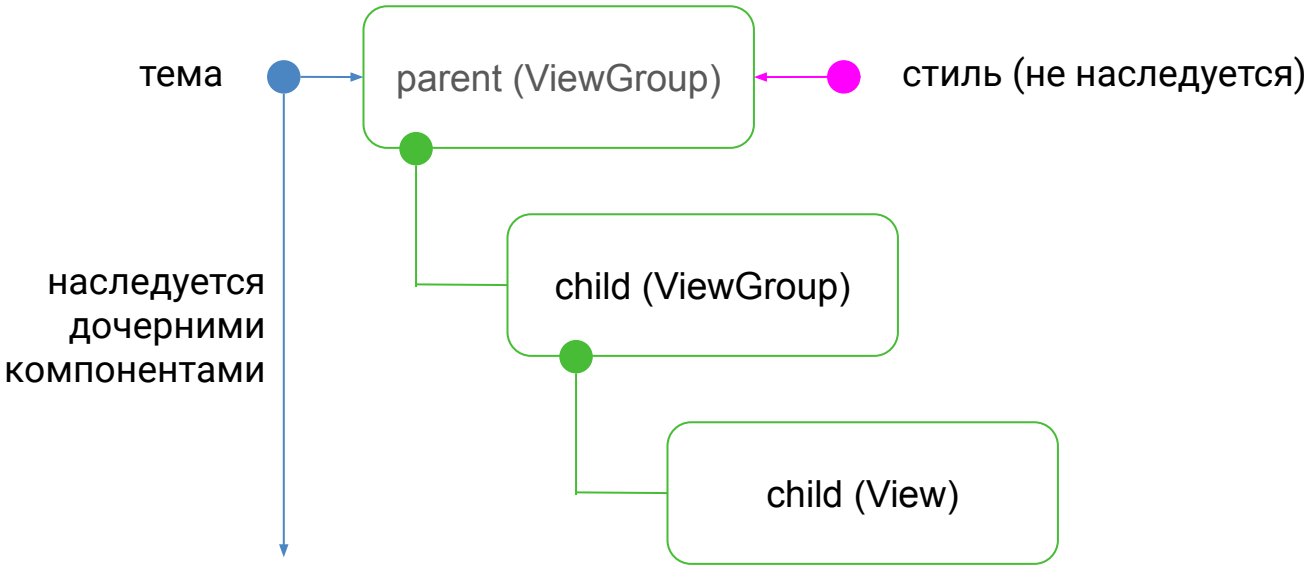
СТИЛИ И ТЕМЫ

Стиль — набор правил оформления (цвет, шрифт и т.д.), которые определяют внешний вид одного компонента. Т.е. мы определяем стиль и применяем его к каждому конкретному компоненту.

Тема — коллекция стилей, которые применяются к компоненту, Activity или всему приложению.

Ключевое отличие

UI организован в иерархическое дерево:



ТЕМЫ

Поскольку мы не хотим с нуля писать тему, мы можем отнаследоваться от уже существующей и переопределять/доопределять только то, что нам нужно.

Для этих целей нам предлагается целый набор тем:

- `Theme.MaterialComponents`
- `Theme.MaterialComponents.NoActionBar`
- `Theme.MaterialComponents.Light`
- `Theme.MaterialComponents.Light.NoActionBar`
- `Theme.MaterialComponents.Light.DarkActionBar`
- `Theme.MaterialComponents.DayNight`
- `Theme.MaterialComponents.DayNight.NoActionBar`
- `Theme.MaterialComponents.DayNight.DarkActionBar`

ТЕМЫ

У нас уже определена базовая тема в файле styles.xml:

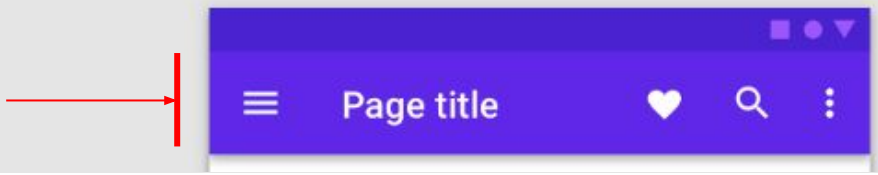
```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
```

ТЕМЫ

Меняем на DayNight:

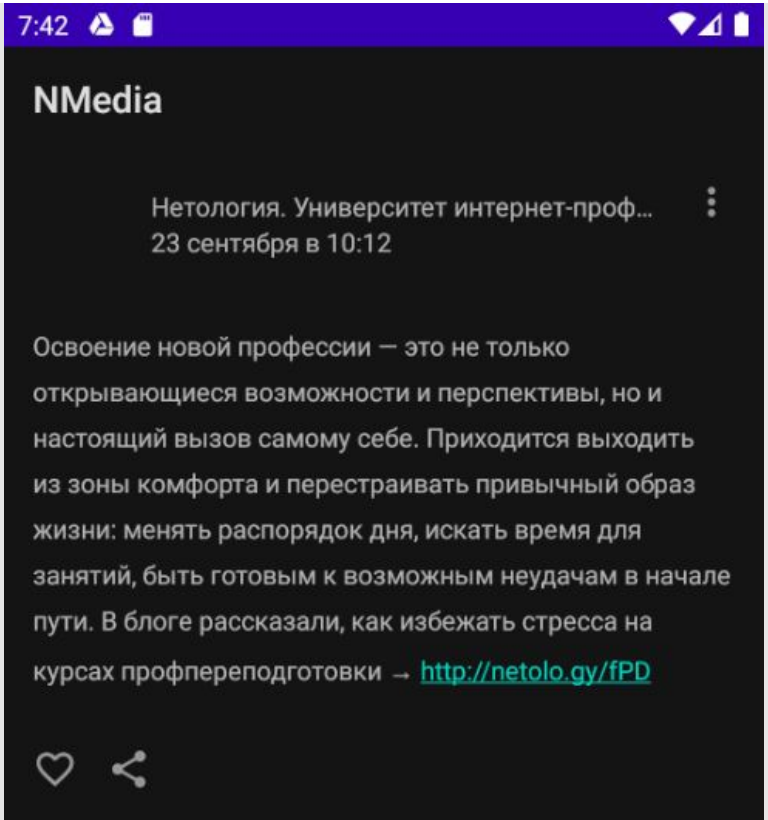
```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
```

ActionBar — это компонент в верхней части экрана:



DAYNIGHT

С первого взгляда ничего не изменилось (после запуска приложения). Но это не совсем так. Если мы включим на устройстве Dark Mode (тёмный режим), то и наше приложение изменится:



Т.е. мы автоматически получили поддержку [тёмной темы]

РАЗМЕРЫ

РАЗМЕРЫ

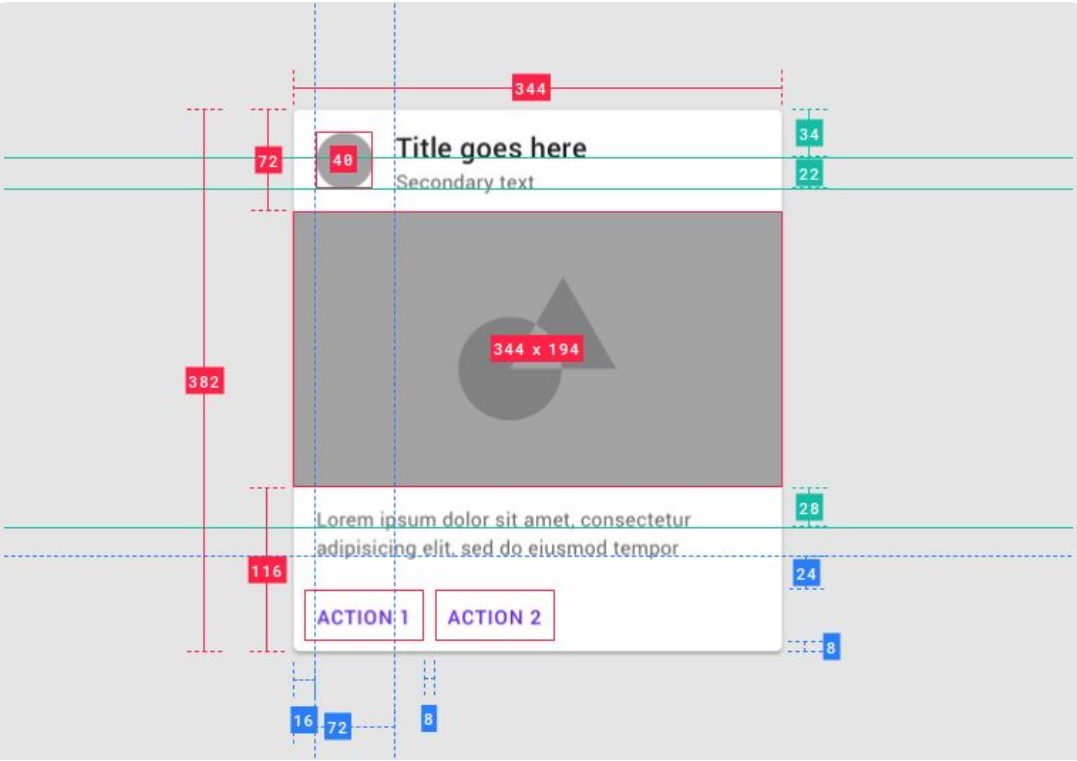
В рамках MD рекомендуется использовать размеры, кратные 4dp (как для расстояний между элементами, так и для размеров самих элементов):

РАЗМЕРЫ

Сама сетка элементов рекомендуется кратной 8dp, но для иконок и текста допустимо 4dp:

РАЗМЕРЫ

Для большинства типовых компонентов приводятся рекомендации* по размерам:



Примечание*: это действительно рекомендации. Вполне допустимо делать аватар или другие элементы больше или меньше.

КОМПОНЕНТЫ

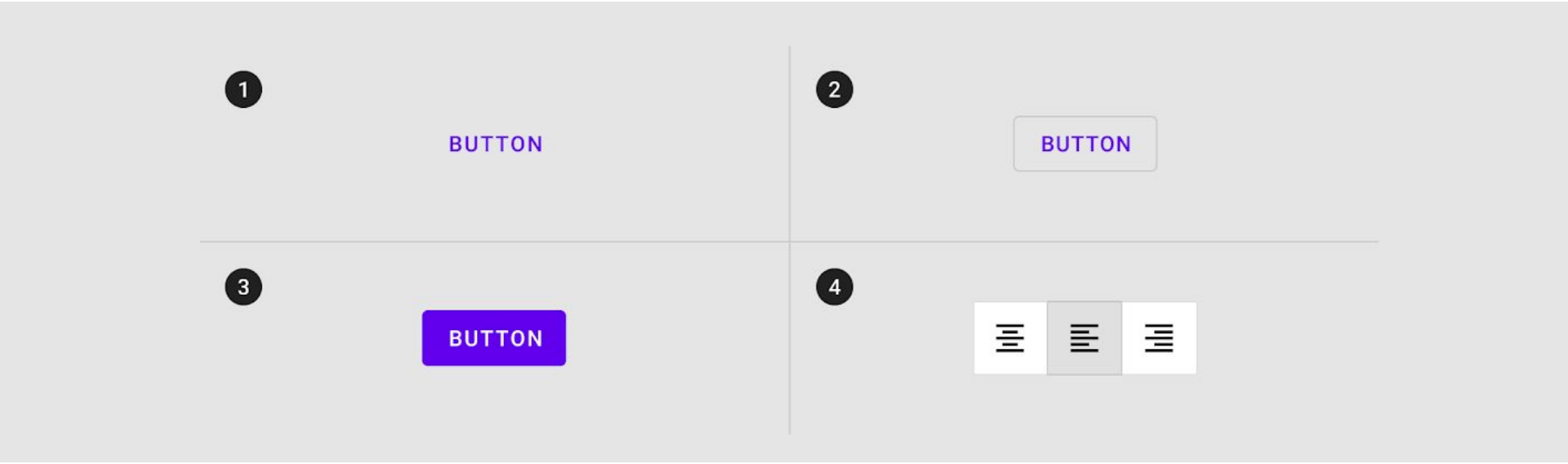
КОМПОНЕНТЫ

Дальнейшая работа строится в следующем режиме:

1. Выбираем из каталога компонентов подходящий компонент
2. Добавляем/изменяем стили (при необходимости)
3. Периодически сверяемся с material.io по рекомендуемым размерам и отступам
4. Пишем собственные компоненты (если готовых не существует) — будем проходить позже

BUTTONS

Начнём с кнопок. Их в MD 4 варианта:



1. Text Button
2. Outlined Button
3. Contained Button
4. Toggle Button

TOGGLE BUTTON

Первым в глаза бросается Toggle Button, поскольку он неплохо подходит для кнопки лайка.

Логически можно выделить два варианта его использования:

1. С текстом
2. Только иконка

Важно: стоит понимать, что Toggle Button — это термин, реализация может осуществляться различными способами (например, с помощью класса CheckBox).

TOGGLE BUTTON

```
<com.google.android.material.checkbox.MaterialCheckBox
    android:id="@+id/like"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/footer"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:button="@drawable/ic_like_24dp"
    android:contentDescription="Like" />
```

Но теперь setImageResource не работает. Поскольку у CheckBox нет такого метода.

Конечно, можно найти соответствующий метод, но хотелось бы больше декларативности.

STATE LIST

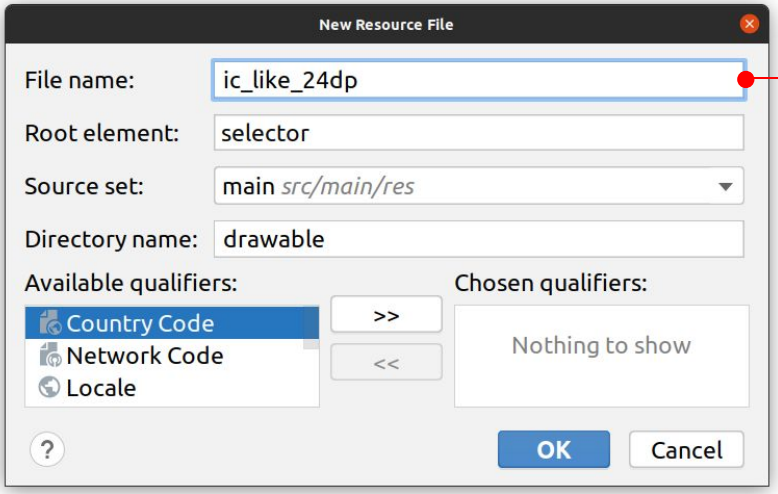
У CheckBox'а есть, как минимум, два состояния:

- флажок установлен (checked)
- флажок снят (unchecked)

Было бы здорово просто назначить на эти состояния разные изображения и заниматься переключением состояний, а Android бы делал смену изображения за нас.

STATE LIST

Android предлагает специальную концепцию State List: мы создаём специальный ресурс, в котором описываем свойства компонента при различных состояниях:




```xml
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- бывший ic_liked -->
    <item
        android:drawable="@drawable/ic_like_filled_24dp"
        android:state_checked="true"
        />
    <!-- бывший ic_like -->
    <item
        android:drawable="@drawable/ic_like_outlined_24dp"
        android:state_checked="false"
        />
    <!-- fallback для всех остальных состояний -->
    <item
        android:drawable="@drawable/ic_like_outlined_24dp"
        />
</selector>
```

STATE LIST

Ключевые идеи:

- выбирается первый подходящий state при проходе сверху вниз (поэтому наверху — самые специфичные, последним — по умолчанию)*
- можно устанавливать требования сразу на несколько состояний, например (state_checked и state_pressed)
- ресурс со state list'ом назначается вместо изображения

STATE LIST

Исходя из правил предыдущей страницы, мы можем сократить количество state'ов то двух:

```xml
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- бывший ic_liked -->
    <item
        android:drawable="@drawable/ic_like_filled_24dp"
        android:state_checked="true"
        />
    <!-- fallback для всех остальных состояний -->
    <item
        android:drawable="@drawable/ic_like_outlined_24dp"
        />
</selector>
```

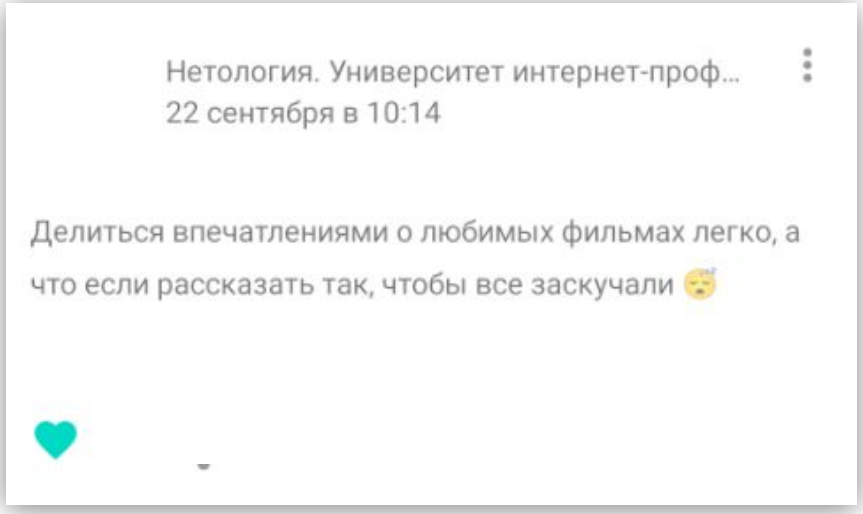
ПРИМЕНЯЕМ

```
<com.google.android.material.checkbox.MaterialCheckBox
    android:id="@+id/like"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/footer"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:button="@drawable/ic_like_24dp"
    android:contentDescription="Like" />
```

```
// в адаптере
like.isChecked = post.likedByMe
вместо
like.setImageResource(
    if (post.likedByMe) R.drawable.ic_liked_24dp else R.drawable.ic_like_24dp
)
```

РЕЗУЛЬТАТ

Вроде всё хорошо, но наша иконка была красной, а видим мы зелёную:



Почему так?

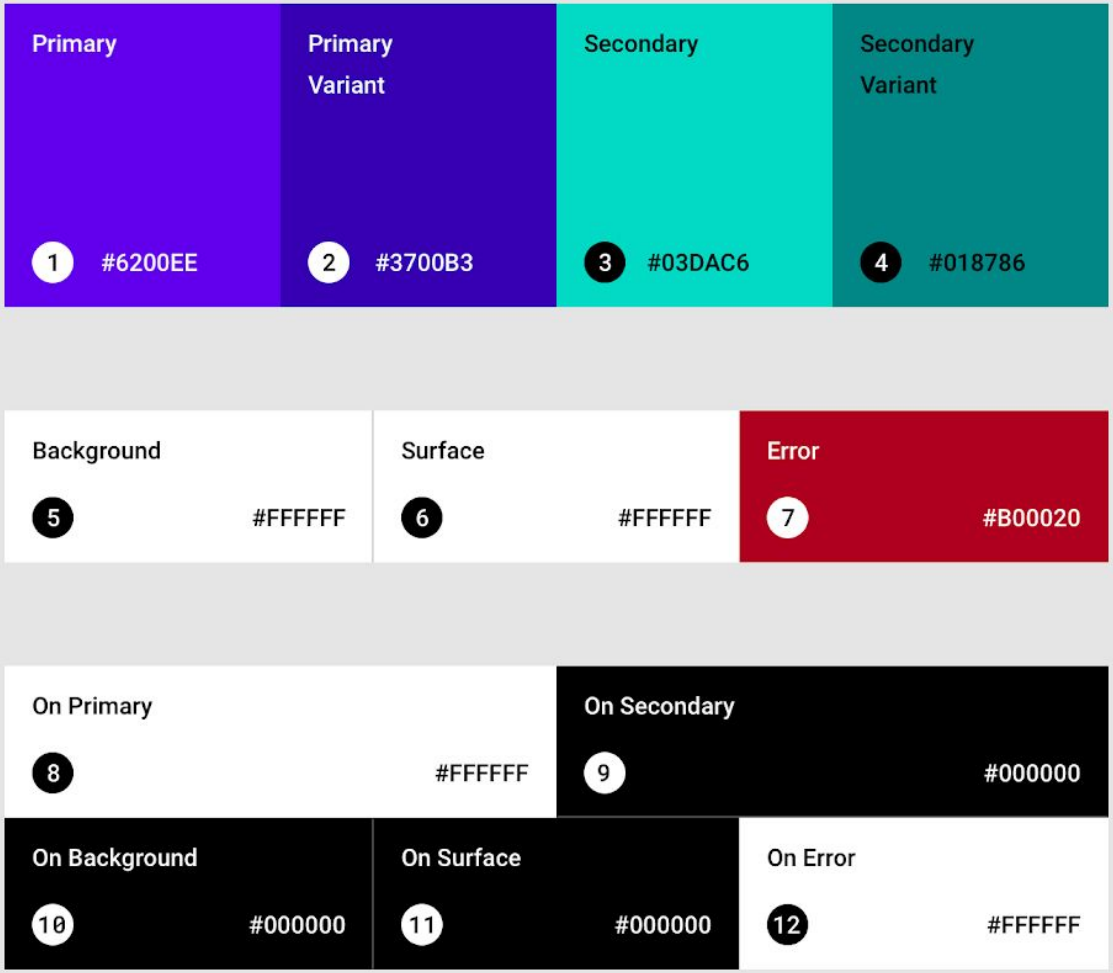
ЦВЕТА

ЦВЕТА

MD определяет для вашего приложения ряд цветов, которые будут использоваться для ключевых элементов (они уже прописаны в нашей теме):

```
<item name="colorPrimary">@color/colorPrimary</item>
<item name="colorPrimaryDark">@color/colorPrimaryDark</item>
<item name="colorAccent">@color/colorAccent</item>
```

```xml
<item name="colorPrimary">@color/colorPrimary</item>
<item name="colorPrimaryDark">@color/colorPrimaryDark</item>
<item name="colorAccent">@color/colorAccent</item>

<color name="colorPrimary">#6200EE</color>
<color name="colorPrimaryDark">#3700B3</color>
<color name="colorAccent">#03DAC5</color>
```



ЦВЕТА

Т.е. мы можем переопределить какой-то цвет в теме, и изменения автоматически применятся ко всем компонентам, которые используют этот цвет.

Но нам же нужно поменять только для лайков...

СТИЛИ

СТИЛИ

Здесь на выручку нам приходят стили. Мы можем создать стиль, который можно применить к конкретному компоненту, не затрагивая все остальные.

При этом механизмы наследования по-прежнему работают (т.е. мы можем отнаследоваться от определённого стиля и заменить только нужные свойства).

СТИЛИ

Определить, какой стиль использует компонент, можно во вкладке Attributes:

style	@style/Widget.MaterialComponents.CompoundButton.CheckBox

Либо на странице конкретного компонента:

Styles

	Style
Default style	`Widget.MaterialComponents.CompoundButton.CheckBox`

Default style theme attribute: `?attr/checkboxStyle`

See the full list of styles and attrs.

СТИЛИ

Определяем свой стиль:

```
<style name="Widget.AppTheme.LikeCheckBox"
    parent="Widget.MaterialComponents.CompoundButton.CheckBox">
    <item name="buttonTint">#ff0000</item>
</style>
```

цвет нужно вынести в ресурсы

С помощью F4, можно пройтись по родительскому стилю:

```
<style name="Widget.MaterialComponents.CompoundButton.CheckBox"
  parent="Widget.AppCompat.CompoundButton.CheckBox">
  <item name="enforceMaterialTheme">true</item>
  <item name="useMaterialThemeColors">true</item>
  <item name="android:minWidth">?attr/minTouchTargetSize</item>
  <item name="android:minHeight">?attr/minTouchTargetSize</item>
</style>
```

?ATTR

?attr — это атрибуты, указанные в теме, позволяющие вам на них ссылаться в своих стилях.

Можете воспринимать их как имена констант, которые определяет тема.

ИСПОЛЬЗУЕМ

```
<com.google.android.material.checkbox.MaterialCheckBox
    android:id="@+id/like"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/footer"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:button="@drawable/ic_like_24dp"
    android:contentDescription="Like"
    style="@style/Widget.AppTheme.LikeCheckBox"
    />
```

ЗАПУСКАЕМ

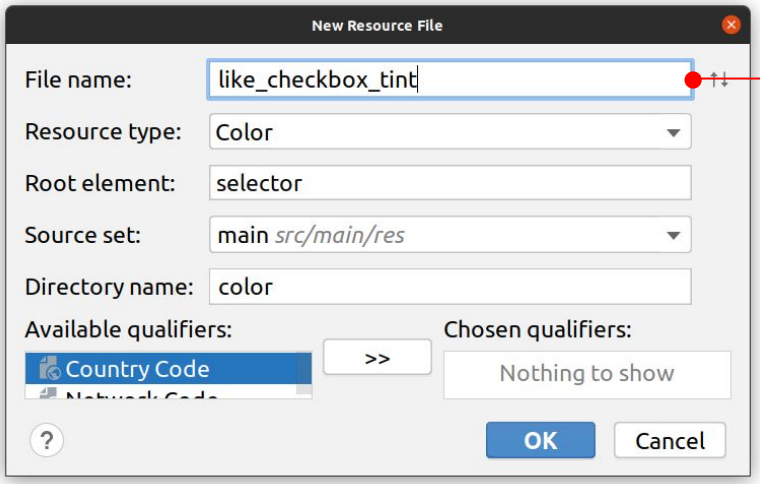
Получаем не совсем то, что нужно нам: 

buttonTint поменял цвет для всех состояний.

? *Есть ли у вас идеи, что с этим можно сделать?*

COLOR

Это нас немного не устраивает. Поэтому мы можем переопределить цвет для разных состояний:




```xml
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="#ff0000" android:state_checked="true"/>
    <item android:color="?attr/colorControlNormal" />
</selector>
```

Обратите внимание: в качестве цвета мы определяем именно ?attr/colorControlNormal. Посмотреть описание атрибутов вы можете по F4. А самые популярные собраны в статье от разработчиков.

CHECKBOX

Checkbox достаточно тяжело и не всегда очевидно кастомизируется, поэтому достаточно часто его просто заменяют на Button:

```
<com.google.android.material.button.MaterialButton
    android:id="@+id/like"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checkable="true"
    android:contentDescription="@string/description_post_like"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/footer"
    app:icon="@drawable/ic_like_24dp"
    tools:checked="@sample/posts.json/data/likedByMe"
    tools:text="@sample/posts.json/data/likes"
    style="@style/Widget.AppTheme.LikeCheckBox"
    />
```

CHECKBOX

И конечно же, стили:

```
<style name="Widget.AppTheme.LikeCheckBox" parent="Widget.MaterialComponents.Button.TextButton.Icon">
    <item name="iconTint">@color/like_checkbox_tint</item>
    <item name="android:minWidth">@dimen/icon_button_min_size</item>
    <item name="android:minHeight">@dimen/icon_button_min_size</item>
</style>
```

Освоение новой профессии — это не только открывающиеся возможности и перспективы, но и настоящий вызов самому себе. Приходится выходить из зоны комфорта и перестраивать привычный образ жизни: менять распорядок дня, искать время для занятий, быть готовым к возможным неудачам в начале пути. В блоге рассказали, как избежать стресса на курсах профпереподготовки → http://netolo.gy/fPD

РЕЗУЛЬТАТ

Вроде всё работает, но если вы внимательно приглядитесь, то увидите, что при каждом лайке текст «бликает». Может показаться, что это только кажется, но нет, вам не кажется :-)

Это связано с анимированием обновления элемента. Что с этим делать, мы поговорим на лекциях по анимации.

ИТОГИ

ИТОГИ

Сегодня мы обсудили вопросы стилизации приложения. Это достаточно большой вопрос — наша задача была в стилизации конкретного элемента.

В рамках курса мы будем постепенно раскрывать вопросы стилизации подробнее, в частности, вопрос стилизации всех компонентов одного класса мы решим при обсуждении создания кастомных компонентов.