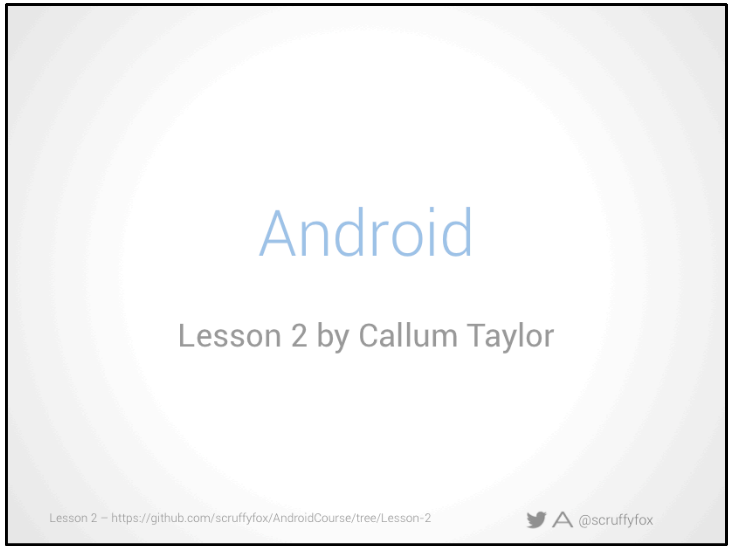
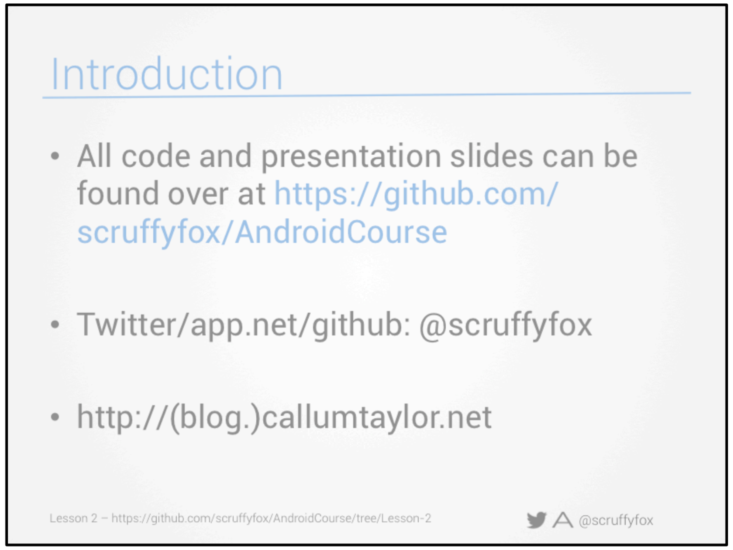


Android

Lesson 2 by Callum Taylor



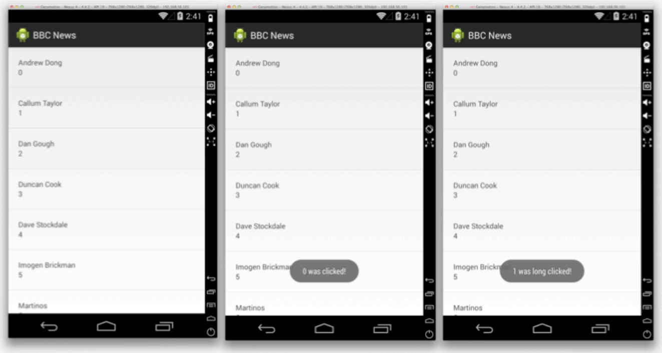
Introduction

- All code and presentation slides can be found over at https://github.com/scruffyfox/AndroidCourse
- Twitter/app.net/github: @scruffyfox
- http://(blog.)callumtaylor.net

Introduction

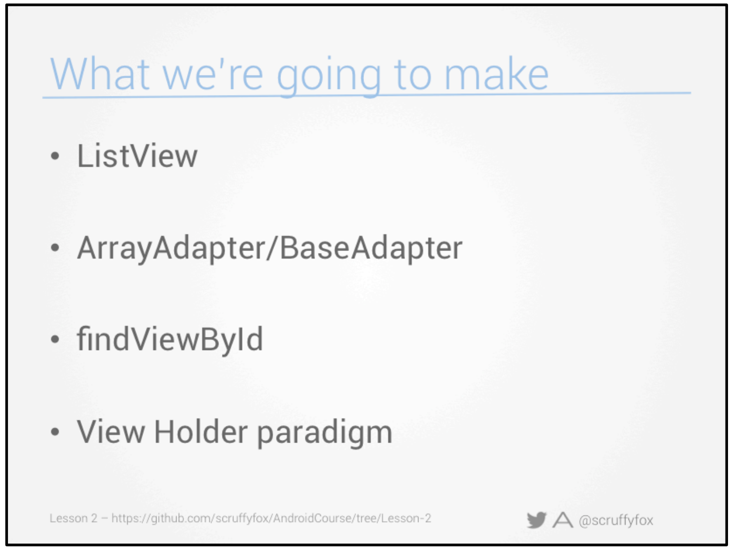
https://github.com/scruffyfox/AndroidCourse/tree/Lesson-2

What we're going to make



Lesson 2 – https://github.com/scruffyfox/AndroidCourse/tree/Lesson-2 @scruffyfox

This is what we're going to make today

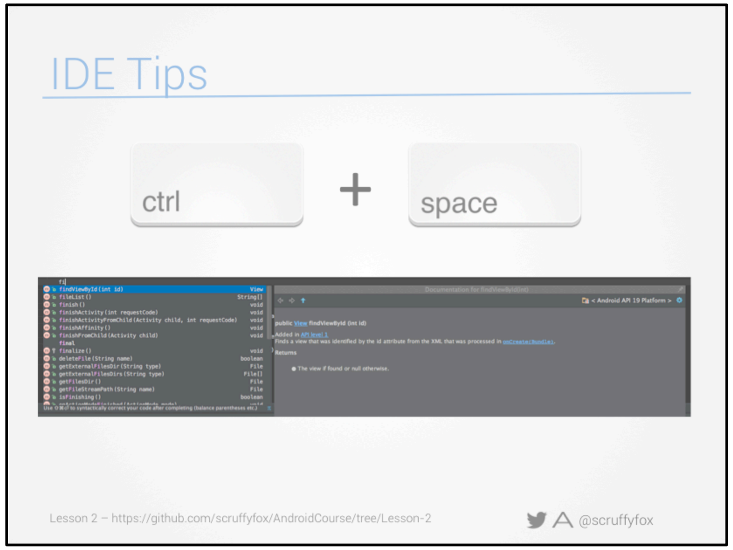


What we're going to make

- ListView
- ArrayAdapter/BaseAdapter
- findViewById
- View Holder paradigm

Lesson 2 – https://github.com/scruffyfox/AndroidCourse/tree/Lesson-2 @scruffyfox

This is what we’re going to make today



IDE Tips

ctrl + space

```
fi
findViewById(int id)                                  View
fileList()                                            String[]
finish()                                              void
finishActivity(int requestCode)                       void
finishActivityFromChild(Activity child, int requestCode)  void
finishAffinity()                                      void
finishFromChild(Activity child)                       void
final
finalize()                                            void
deleteFile(String name)                               boolean
getExternalFilesDir(String type)                      File
getExternalFilesDirs(String type)                     File[]
getFilesDir()                                         File
getFileStreamPath(String name)                        File
isFinishing()                                         boolean
Use ⇧⌘⏎ to syntactically correct your code after completing (balance parentheses etc.)
```

Documentation for findViewById(int)

< Android API 19 Platform >

public View findViewById (int id)

Added in API level 1

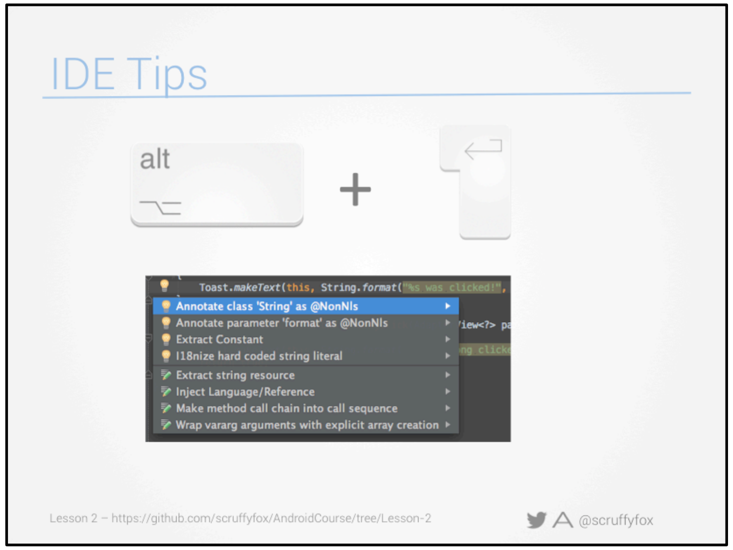
Finds a view that was identified by the id attribute from the XML that was processed in onCreate(Bundle).

Returns

- The view if found or null otherwise.

Lesson 2 – https://github.com/scruffyfox/AndroidCourse/tree/Lesson-2 @scruffyfox

Control + space will be the most popular shortcut you use, pressing it will bring up the codesense suggestion dialog when you type to allow you to find the method/variable you're looking for. It will also bring up the documentation for the highlighted method. Press enter to auto complete



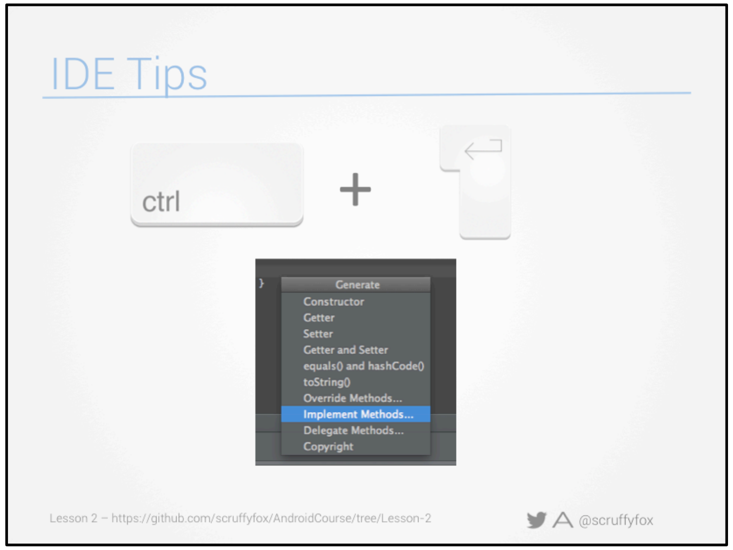
IDE Tips

```
Toast.makeText(this, String.format("%s was clicked!",
```

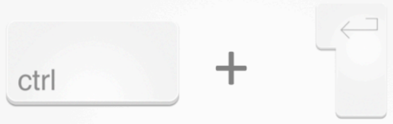
Annotate class 'String' as @NonNls
Annotate parameter 'format' as @NonNls
Extract Constant
I18nize hard coded string literal
Extract string resource
Inject Language/Reference
Make method call chain into call sequence
Wrap vararg arguments with explicit array creation

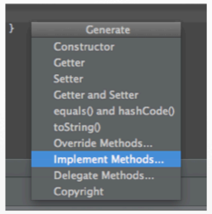
Lesson 2 – https://github.com/scruffyfox/AndroidCourse/tree/Lesson-2 @scruffyfox

Pressing alt and enter will open up the quick fix menu which will give you some useful functions such as refracting a string or reversing an if statement



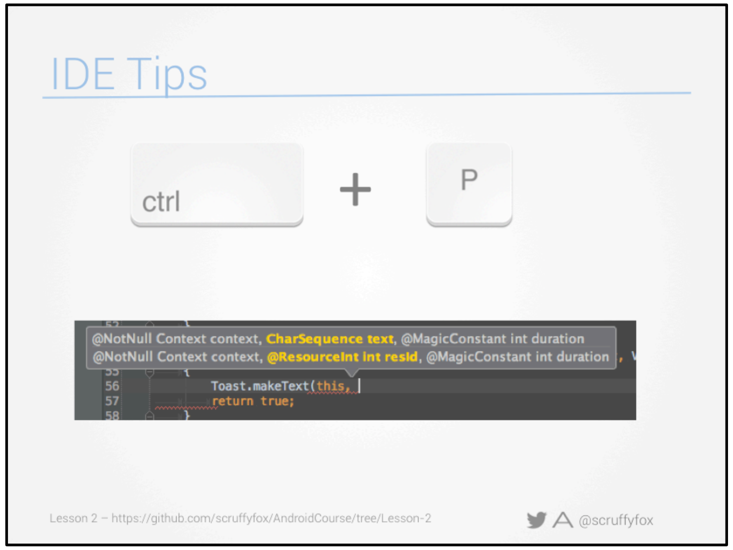
IDE Tips





Lesson 2 – https://github.com/scruffyfox/AndroidCourse/tree/Lesson-2 @scruffyfox

Pressing control and enter will bring up the generate menu which is useful when overriding methods from the super class you don’t know the name of



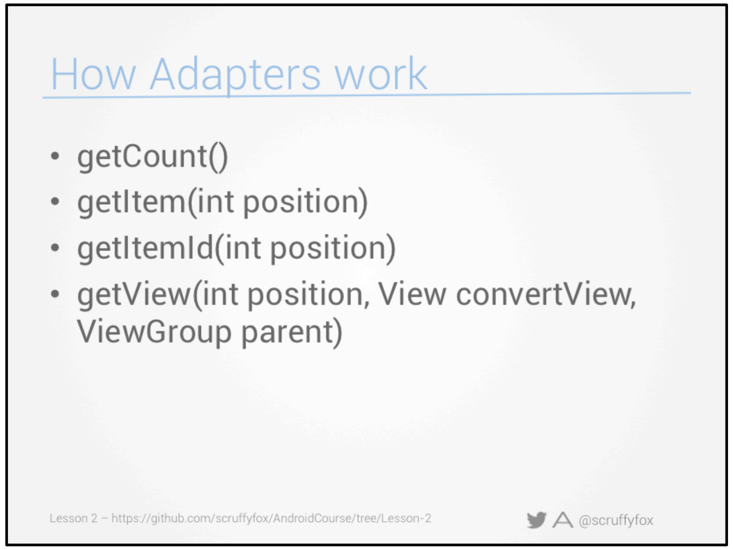
IDE Tips

ctrl + P

```
@NotNull Context context, CharSequence text, @MagicConstant int duration
@NotNull Context context, @ResourceInt int resId, @MagicConstant int duration
        Toast.makeText(this, |
        return true;
```

Lesson 2 – https://github.com/scruffyfox/AndroidCourse/tree/Lesson-2 @scruffyfox

Pressing control and p will bring up the parameters for a method as you're typing.

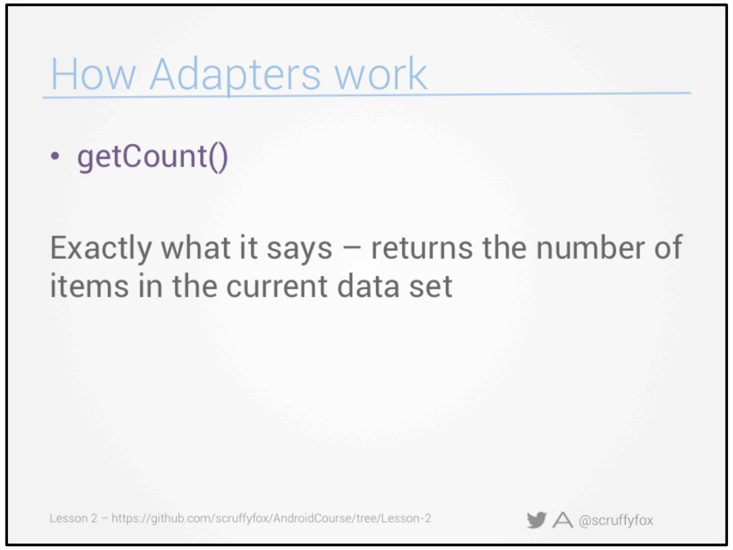


How Adapters work

- getCount()
- getItem(int position)
- getItemId(int position)
- getView(int position, View convertView, ViewGroup parent)

Lesson 2 – https://github.com/scruffyfox/AndroidCourse/tree/Lesson-2 @scruffyfox

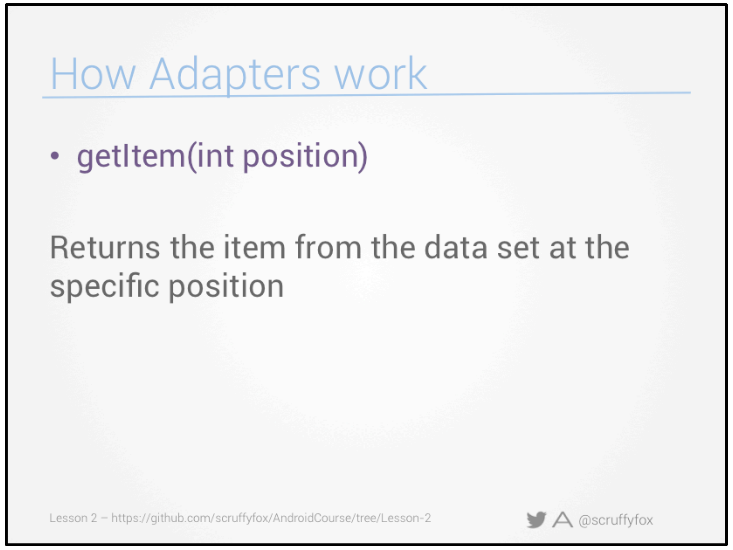
Adapters are very clever in its design of serving a large data set to the UI of an app. The pattern consists of a few core methods.



How Adapters work

- getCount()

Exactly what it says – returns the number of items in the current data set



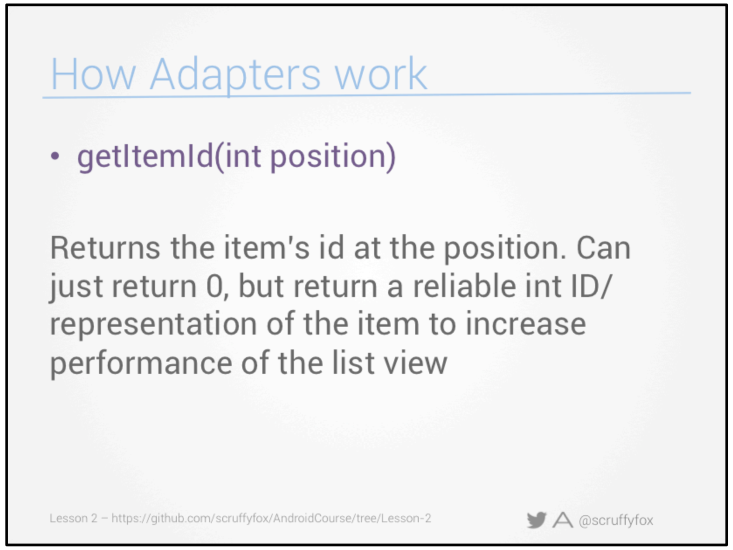
How Adapters work

- getItem(int position)

Returns the item from the data set at the specific position

Lesson 2 – https://github.com/scruffyfox/AndroidCourse/tree/Lesson-2

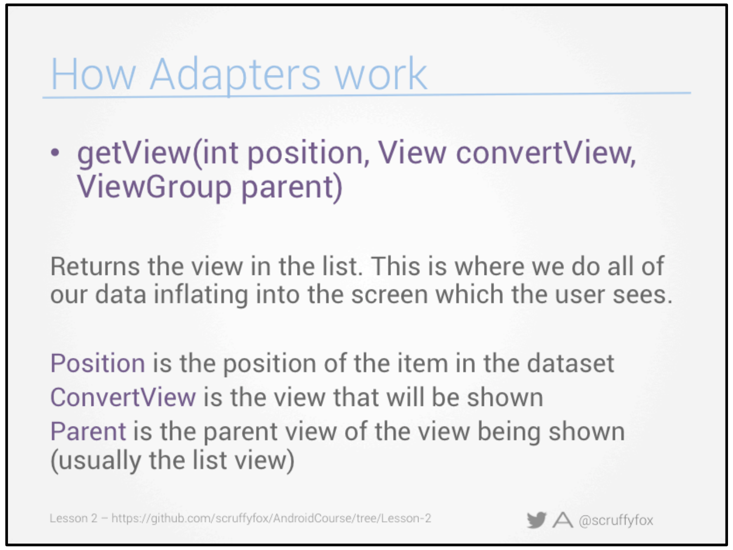
@scruffyfox



How Adapters work

- getItemId(int position)

Returns the item's id at the position. Can just return 0, but return a reliable int ID/ representation of the item to increase performance of the list view

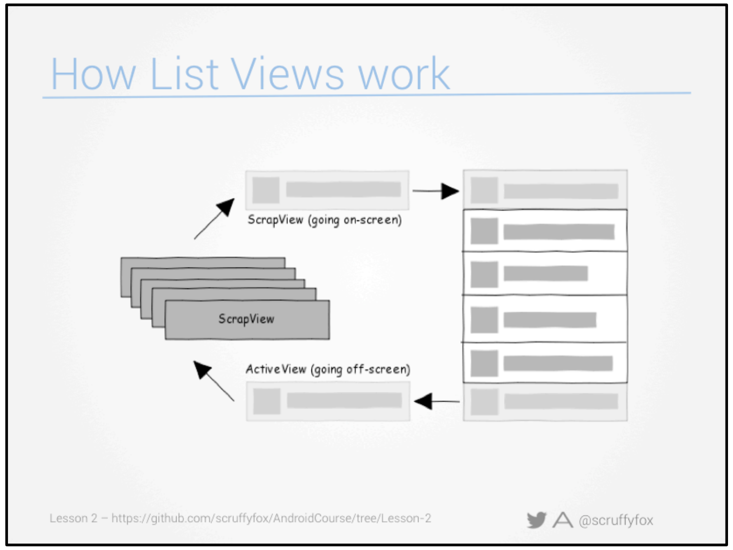


How Adapters work

- getView(int position, View convertView, ViewGroup parent)

Returns the view in the list. This is where we do all of our data inflating into the screen which the user sees.

Position is the position of the item in the dataset
ConvertView is the view that will be shown
Parent is the parent view of the view being shown (usually the list view)

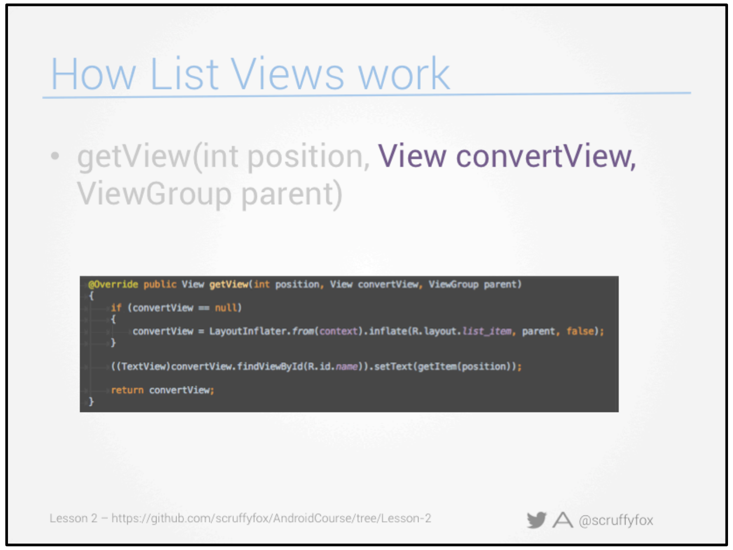


How List Views work

Lesson 2 – https://github.com/scruffyfox/AndroidCourse/tree/Lesson-2 @scruffyfox

List views are very clever with how they show the views on the screen. If you have a data set of say 10,000 items, obviously to have 10,000 views on the screen would cause the phone to firstly crash, but also to be EXTREMELY slow. The way a list view works to solve this is called **view recycling.**

When you scroll up or down, the view at the top or bottom drops off the screen, when it is no longer visible, it will get put into a scrap view heap, and when a new view is shown on the screen, that view is recycled from the stack, back on to the screen. This is why when we only set the views when convertView is null, it will jump around as you scroll.



How List Views work

- getView(int position, View convertView, ViewGroup parent)

```
@Override public View getView(int position, View convertView, ViewGroup parent)
{
    if (convertView == null)
    {
        convertView = LayoutInflater.from(context).inflate(R.layout.list_item, parent, false);
    }

    ((TextView)convertView.findViewById(R.id.name)).setText(getItem(position));

    return convertView;
}
```

Lesson 2 – https://github.com/scruffyfox/AndroidCourse/tree/Lesson-2 @scruffyfox

In our adapter, this is where convertView comes in handy.

When a new view is being created, convertView will be null. Here it will be safe to create our view and return it. But when convertView is **not** null, that is the view being recycled, so we don't have to create another view, we can just reuse the same view but change the contents to reflect the data.