RAVTECH

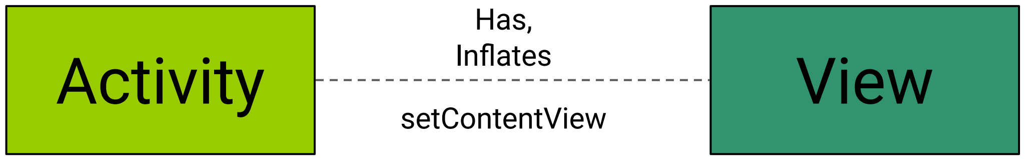
Android Lecture #4

What are we doing today?

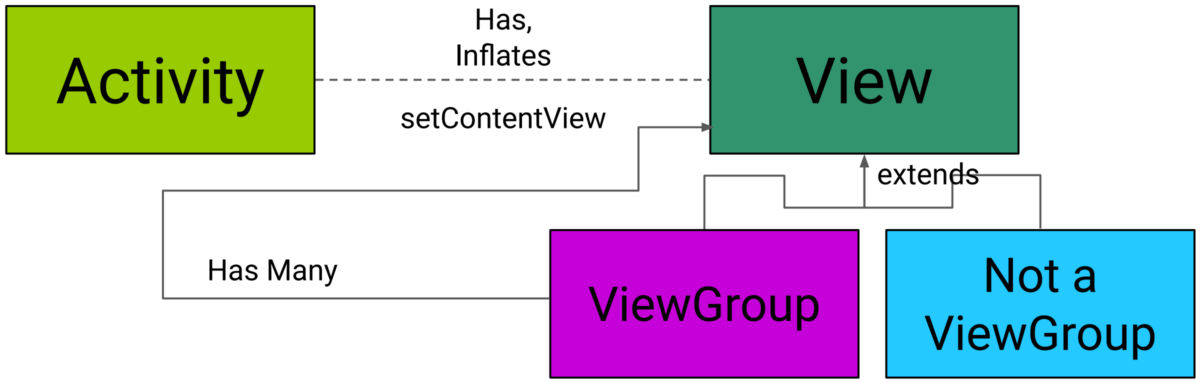
- Introduction to ListView and Adapters
- Android view recycling
- GridView
- RecyclerView

Why/When do I need to use a ListView?

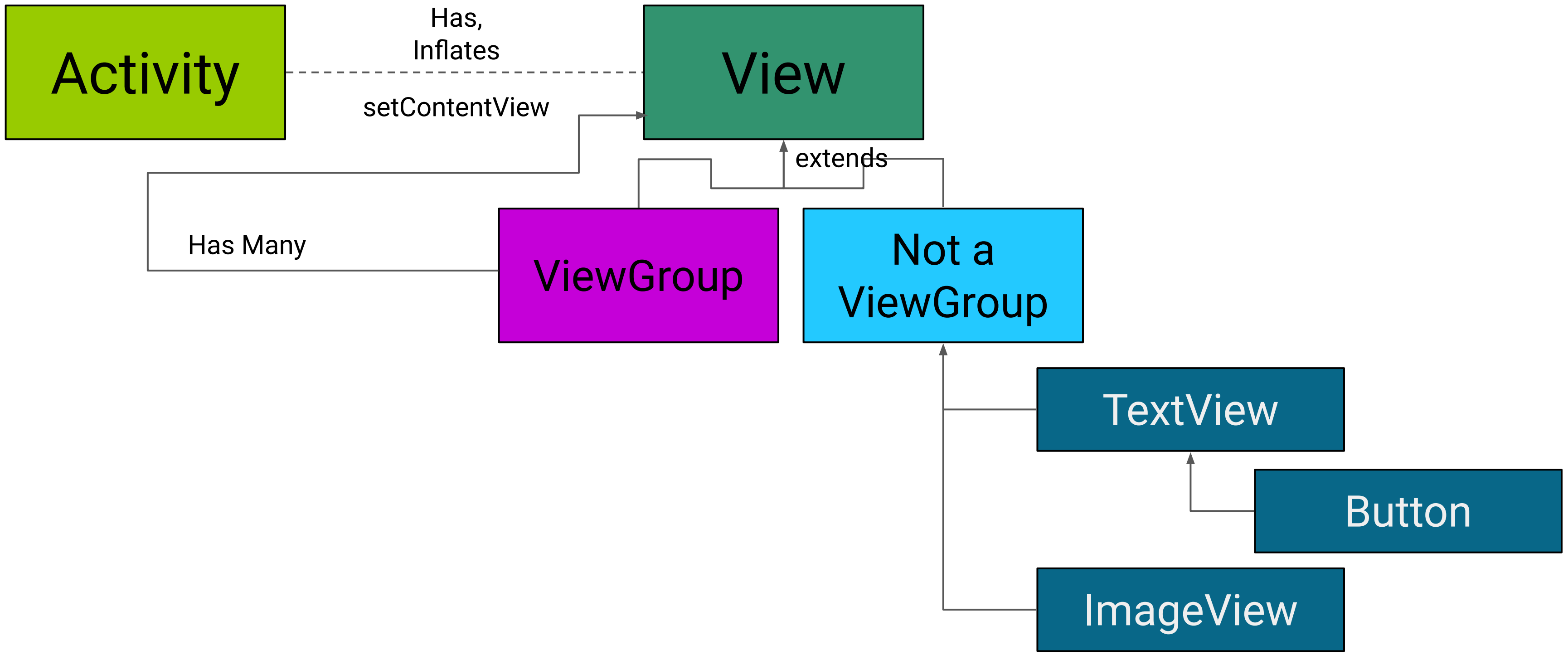
Quick UML Recap



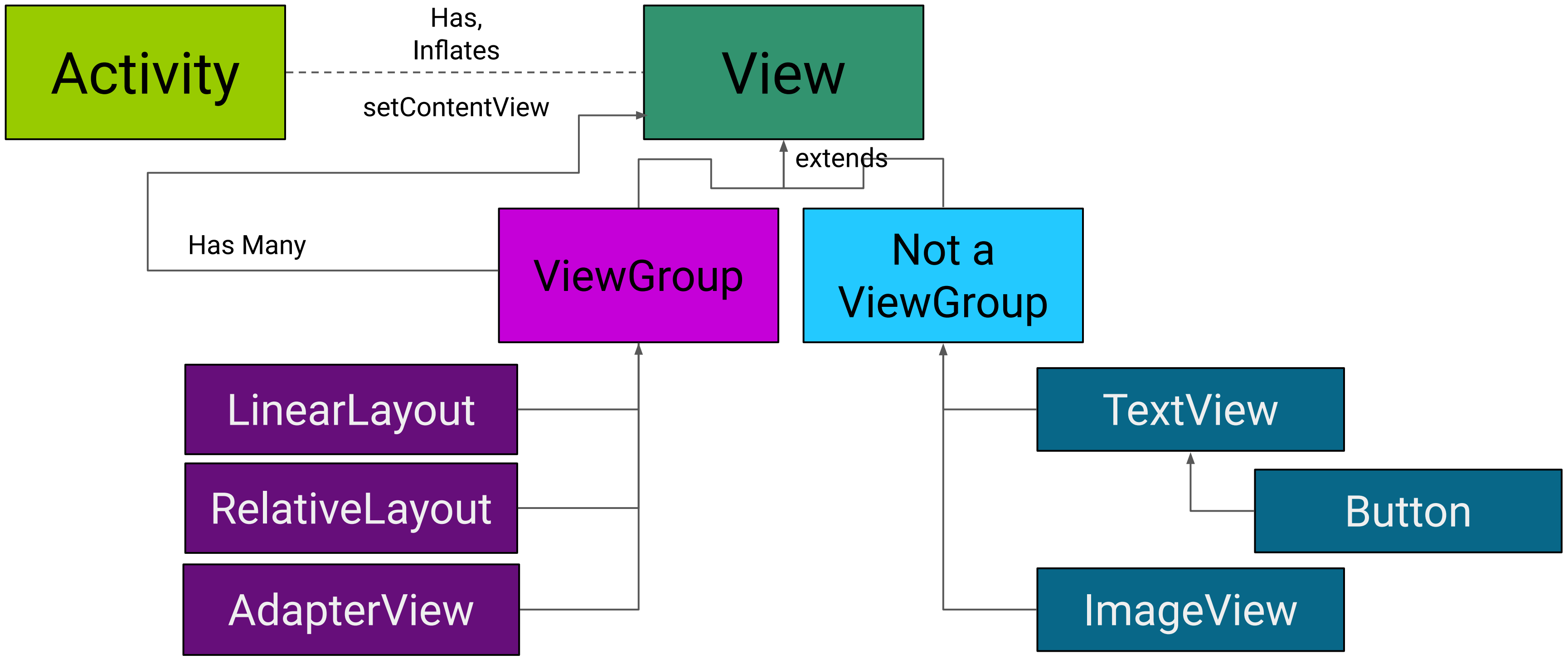
Quick UML Recap



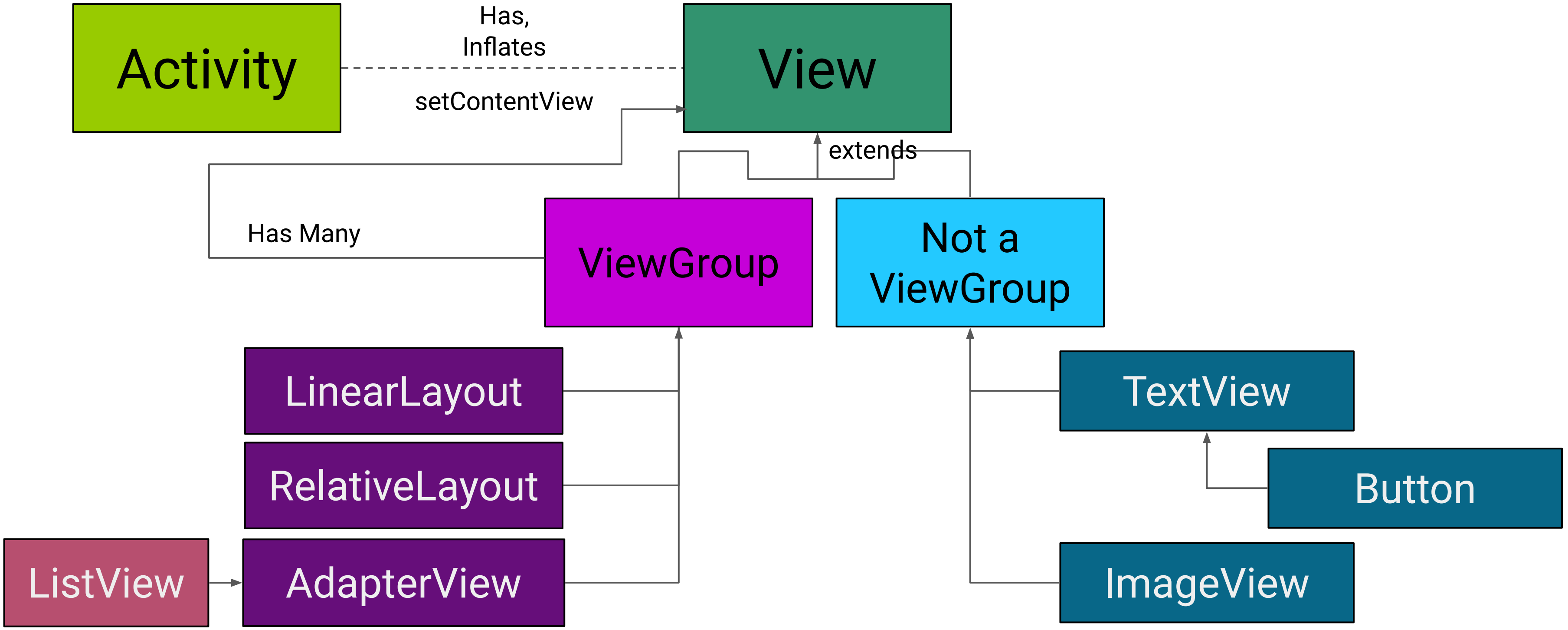
Quick UML Recap



Quick UML Recap



Quick UML Recap

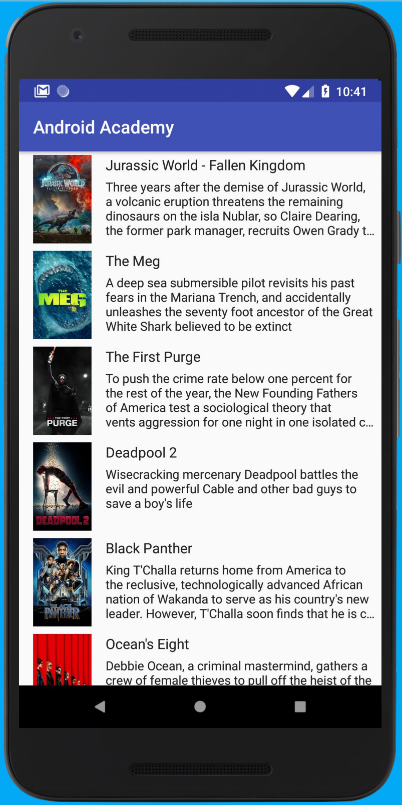


1.0 The ListView

- A ViewGroup
- Displays a list of **scrollable** items.
- The items are **automatically** inserted to the list using an **Adapter**.

Read more: http://developer.android.com/guide/topics/ui/layout/listview.html

How Does It Work?



Android Academy

Jurassic World - Fallen Kingdom
Three years after the demise of Jurassic World, a volcanic eruption threatens the remaining dinosaurs on the isla Nublar, so Claire Dearing, the former park manager, recruits Owen Grady t...

The Meg
A deep sea submersible pilot revisits his past fears in the Mariana Trench, and accidentally unleashes the seventy foot ancestor of the Great White Shark believed to be extinct

The First Purge
To push the crime rate below one percent for the rest of the year, the New Founding Fathers of America test a sociological theory that vents aggression for one night in one isolated c...

Deadpool 2
Wisecracking mercenary Deadpool battles the evil and powerful Cable and other bad guys to save a boy's life

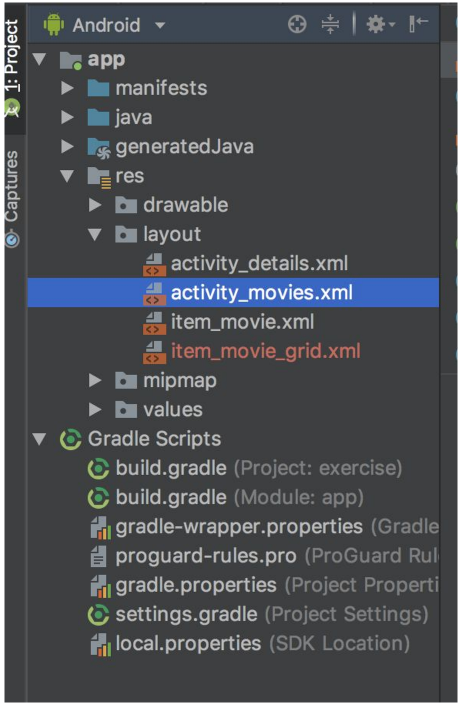
Black Panther
King T'Challa returns home from America to the reclusive, technologically advanced African nation of Wakanda to serve as his country's new leader. However, T'Challa soon finds that he is c...

Ocean's Eight
Debbie Ocean, a criminal mastermind, gathers a crew of female thieves to pull off the heist of the

1.0 ListView Recipe

1. Create a **ListView** view
2. Create a **row** layout (or use existing one)
3. Create **data** object list
4. Create an **Adapter**
5. **Bind** Adapter to the ListView

1.1 ListView Recipe


Android
app
manifests
java
generatedJava
res
drawable
layout
activity_details.xml
activity_movies.xml
item_movie.xml
item_movie_grid.xml
mipmap
values
Gradle Scripts
build.gradle (Project: exercise)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle
proguard-rules.pro (ProGuard Rul
gradle.properties (Project Properti
settings.gradle (Project Settings)
local.properties (SDK Location)

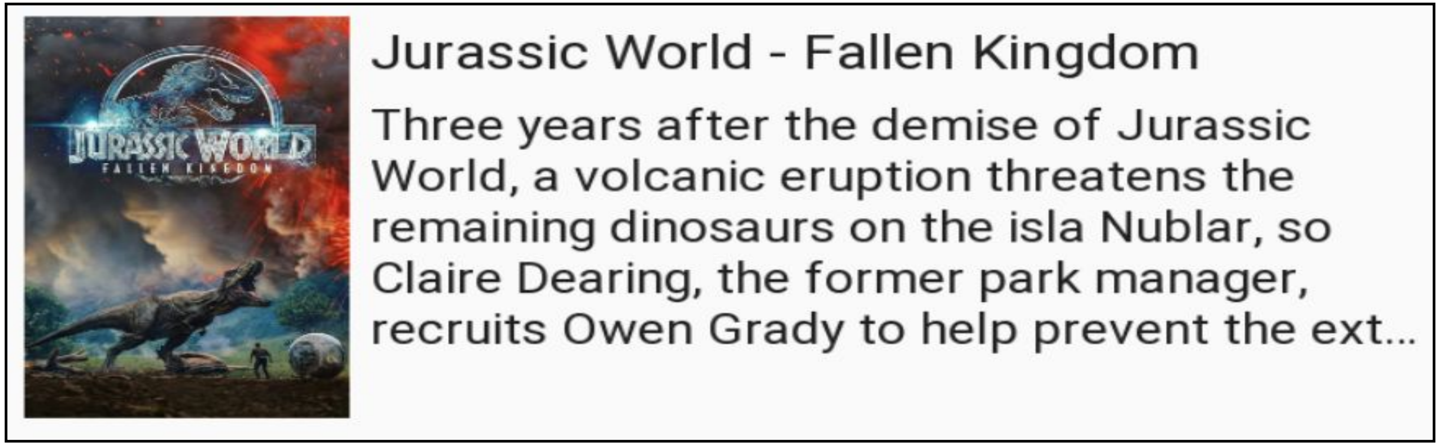
```
<?xml version="1.0" encoding="utf-8"?>

<ListView

    xmlns:android="http://schemas.android.com/apk/res/android"

    android:id="@+id/movies_lv"

    android:layout_width="match_parent"

    android:layout_height="match_parent"/>
```

1.1 ListView Recipe

1. ~~Create a **ListView** view~~
2. Create a **row** layout (or use existing one)
3. Create **data** object list
4. Create an **Adapter**
5. **Bind** Adapter to the ListView

1.1 Row Layout

Let’s consider this layout

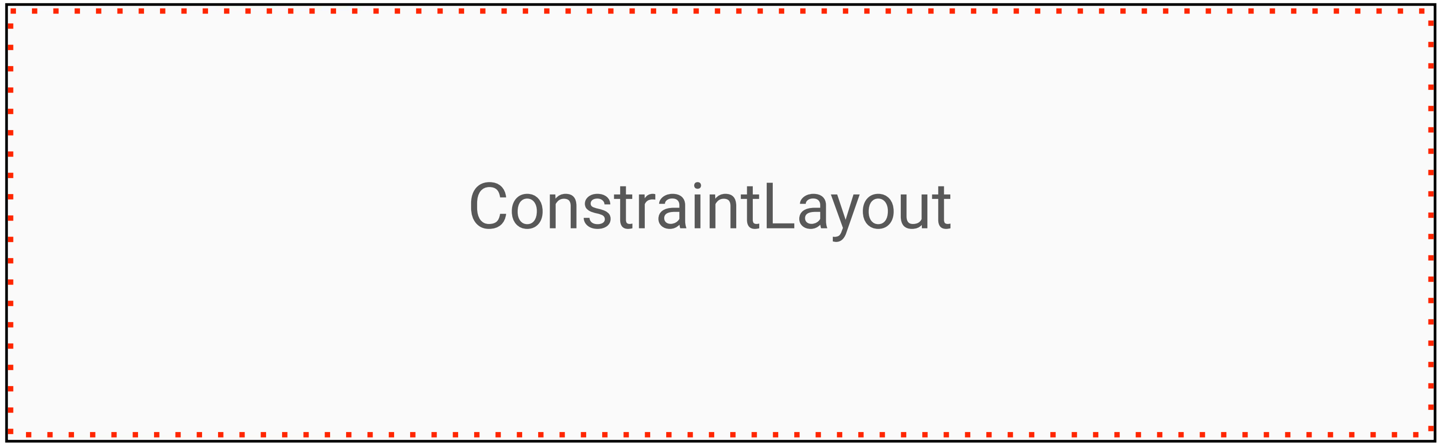


Jurassic World - Fallen Kingdom

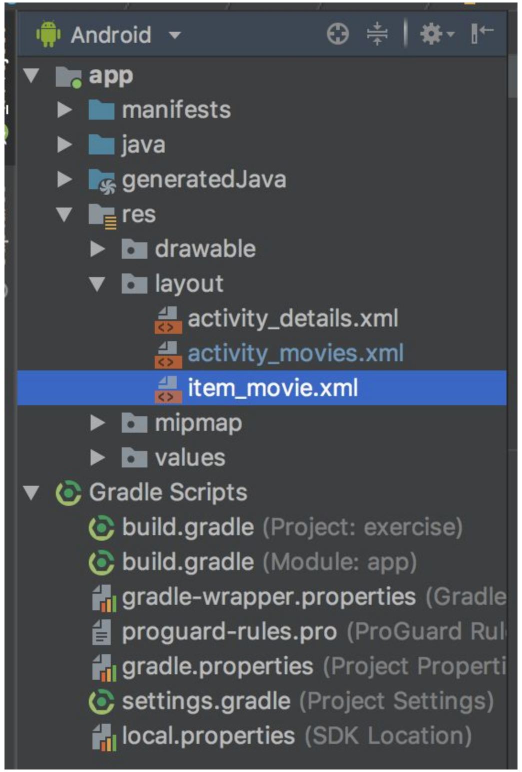
Three years after the demise of Jurassic World, a volcanic eruption threatens the remaining dinosaurs on the isla Nublar, so Claire Dearing, the former park manager, recruits Owen Grady to help prevent the ext...

1.1 Row Layout

Let’s consider this layout



1.1 Row Layout



```
Android
▼ app
  ▶ manifests
  ▶ java
  ▶ generatedJava
  ▼ res
    ▶ drawable
    ▼ layout
        activity_details.xml
        activity_movies.xml
        item_movie.xml
    ▶ mipmap
    ▶ values
▼ Gradle Scripts
    build.gradle (Project: exercise)
    build.gradle (Module: app)
    gradle-wrapper.properties (Gradle
    proguard-rules.pro (ProGuard Rul
    gradle.properties (Project Properti
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

1.1 Create new item_movie.xml file (Row Layout)

```
<?xml version="1.0" encoding="utf-8"?>

<ConstraintLayout>









</ConstraintLayout>
```

1.1 Create new item_movie.xml file (Row Layout)

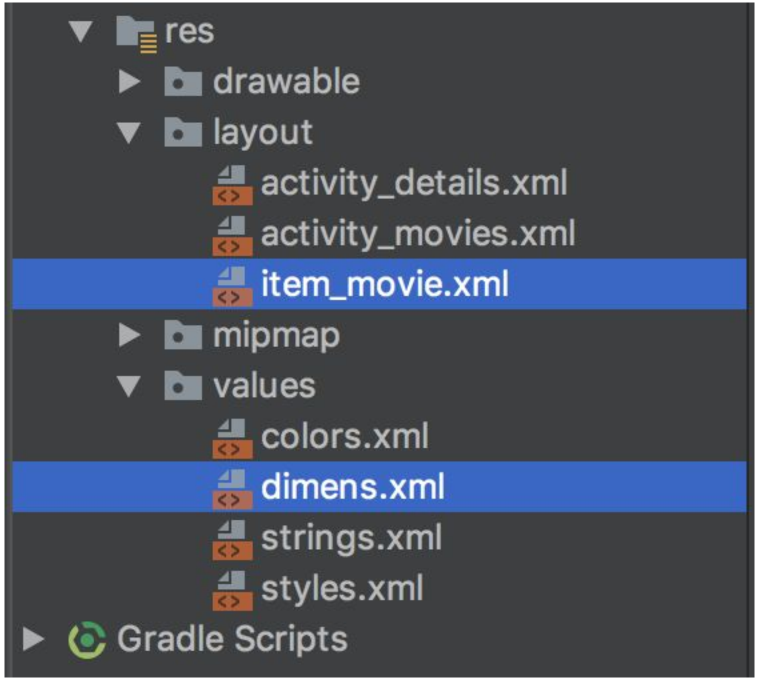
```
<?xml version="1.0" encoding="utf-8"?>
<ConstraintLayout>
   xmlns:android=“http://schemas.android.com/apk/res/android"
   xmlns:tools=“http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height=“108dp”>







</ConstraintLayout>
```

1.1 Create new item_movie.xml file (Row Layout)

```
<?xml version="1.0" encoding="utf-8"?>
<ConstraintLayout>
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="108dp">




</ConstraintLayout>
```

1.1 Best practice tip - use dimens.xml!

- Reusable
- Density Difference



```
▼ res
  ▶ drawable
  ▼ layout
      activity_details.xml
      activity_movies.xml
      item_movie.xml
  ▶ mipmap
  ▼ values
      colors.xml
      dimens.xml
      strings.xml
      styles.xml
▶ Gradle Scripts
```

1.1 dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        …



</resources>
```

1.1 dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
       …
     <!--Movie list item values→
     <dimen name="li_movie_height">108dp</dimen>
</resources>
```

1.1 Create new item_movie.xml file (Row Layout)

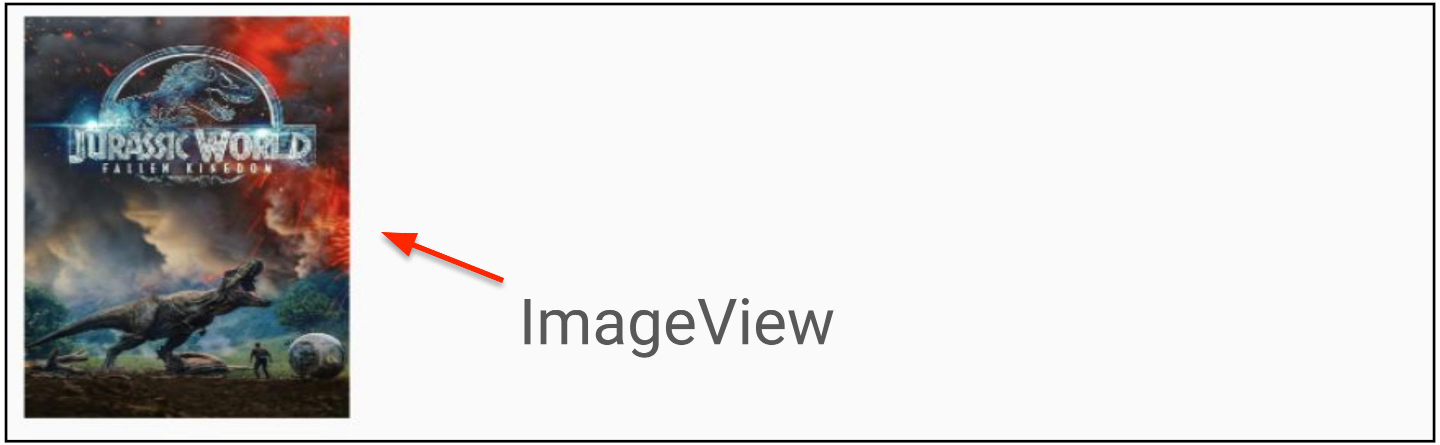
```
<?xml version="1.0" encoding="utf-8"?>
<ConstraintLayout>
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="@dimen/li_movie_height">
```

108dp row height

```




</ConstraintLayout>
```

1.1 Row Layout

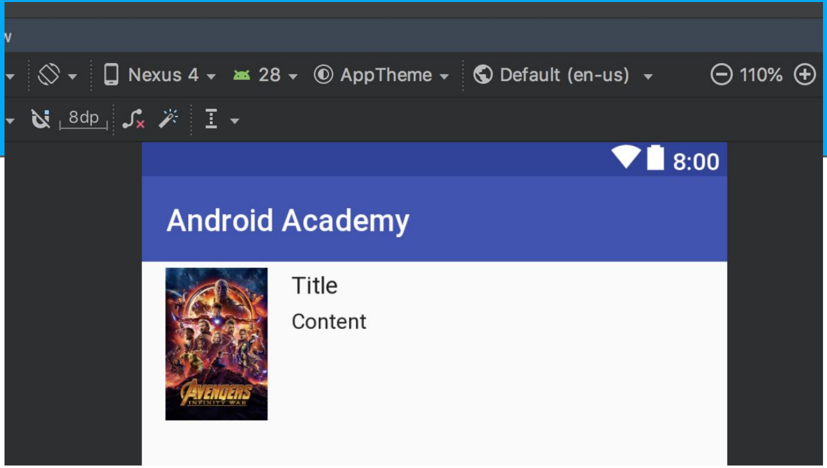
Let's consider this layout



1.1 Row Layout

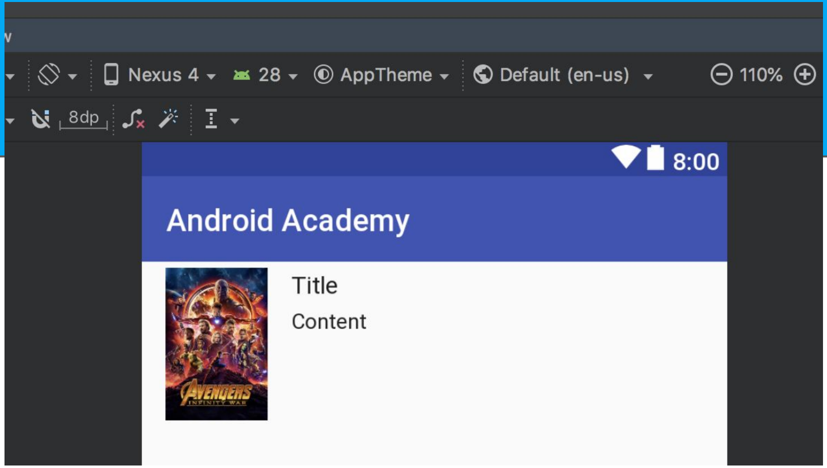
```
<?xml version="1.0" encoding="utf-8"?>
<ConstraintLayout...>


        <ImageView
         android:id="@+id/item_movie_iv"
         android:layout_width="90dp"
         android:layout_height="0dp"
         app:layout_constraintBottom_toBottomOf="parent"
         app:layout_constraintStart_toStartOf="parent"
         app:layout_constraintTop_toTopOf="parent"
         android:src="@drawable/infinity_war_image"  />


      …
</ConstraintLayout>
```

1.1 Row Layout



```
<?xml version="1.0" encoding="utf-8"?>
<ConstraintLayout...>

        <ImageView
         android:id="@+id/item_movie_iv"
         android:layout_width="90dp"
         android:layout_height="0dp"
         app:layout_constraintBottom_toBottomOf="parent"
         app:layout_constraintStart_toStartOf="parent"
         app:layout_constraintTop_toTopOf="parent"
         android:src="@drawable/infinity_war_image" />

      …
</ConstraintLayout>
```

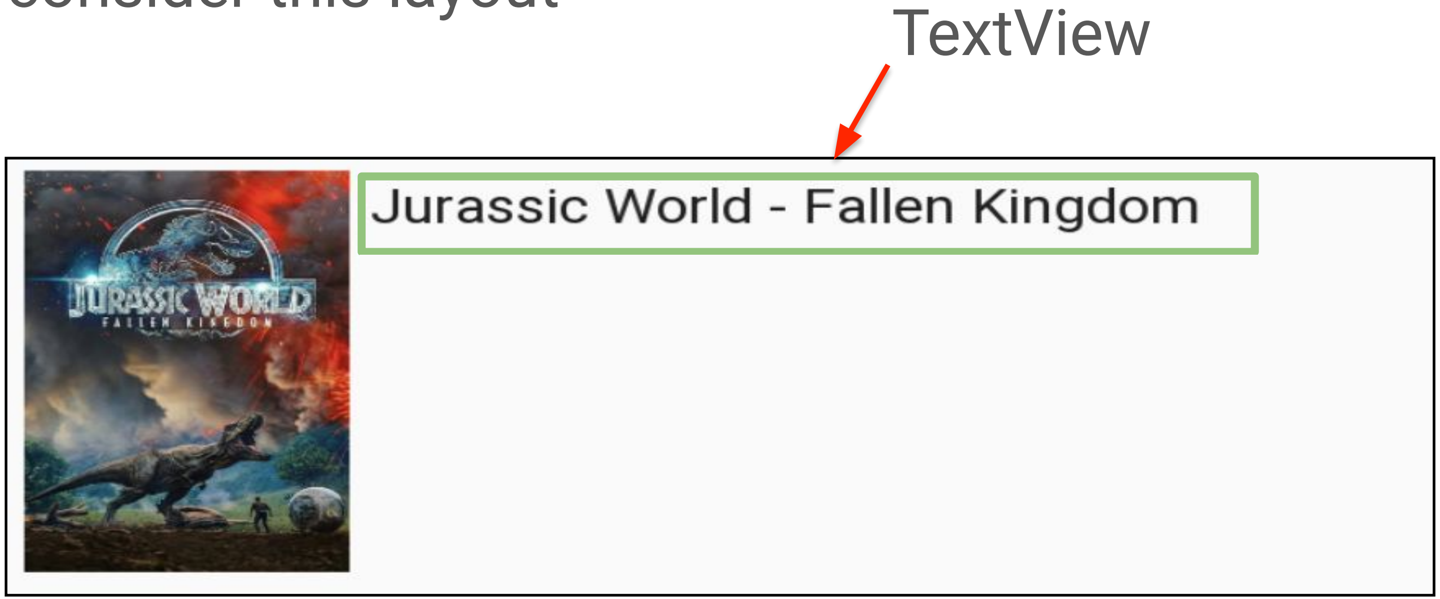
1.1 Row Layout



```
<?xml version="1.0" encoding="utf-8"?>
<ConstraintLayout...>

        <ImageView
         android:id="@+id/item_movie_iv"
         android:layout_width="90dp"
         android:layout_height="0dp"
         app:layout_constraintBottom_toBottomOf="parent"
         app:layout_constraintStart_toStartOf="parent"
         app:layout_constraintTop_toTopOf="parent"
         tools:src=“@drawable/infinity_war_image"  />

      …
</ConstraintLayout>
```

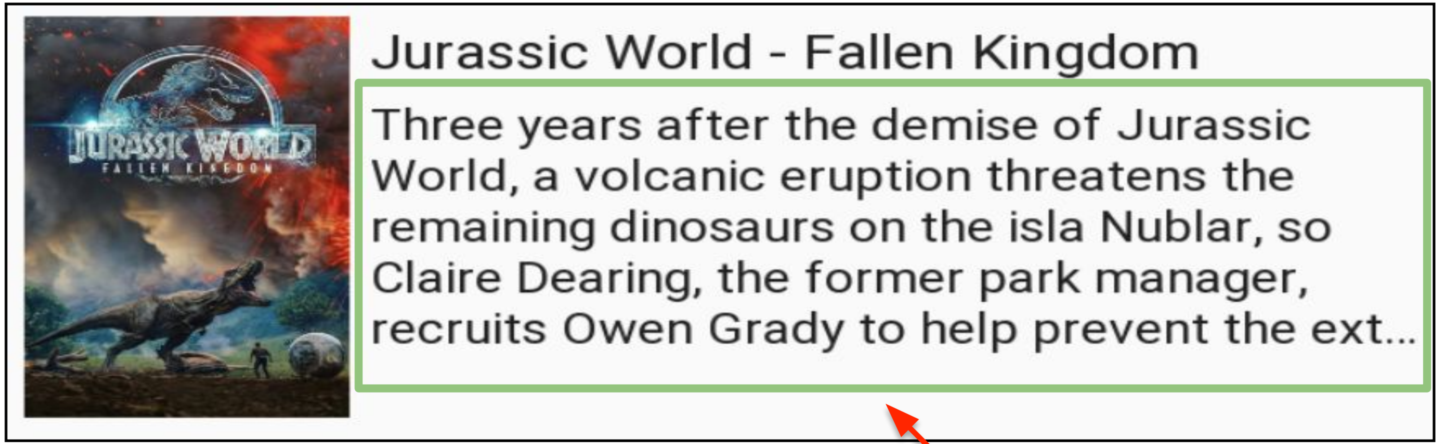
1.1 Row Layout

Let’s consider this layout



1.1 Row Layout

Let's consider this layout



Jurassic World - Fallen Kingdom

Three years after the demise of Jurassic World, a volcanic eruption threatens the remaining dinosaurs on the isla Nublar, so Claire Dearing, the former park manager, recruits Owen Grady to help prevent the ext...

TextView

1.1 Row Layout

```
<?xml version="1.0" encoding="utf-8"?>
<ConstraintLayout>



   <TextView…>


   <TextView…>






</ConstraintLayout>
```

1.1 ListView Recipe

1. ~~Create a **ListView** view~~
2. ~~Create a **row** layout (or use existing one)~~
3. Create **data** object list
4. Create an **Adapter**
5. **Bind** Adapter to the ListView

1.1 Our data (POJO)

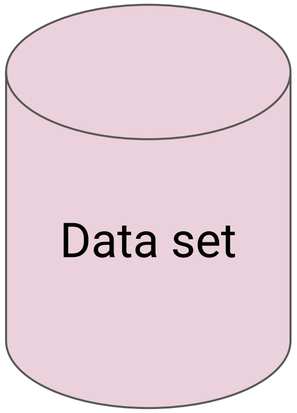
```
public class MovieModel {

            private String name;
            private int imageResourceId;
            private String overview;

             // getters and setters
             . . .
}
```

1.1 ListView Recipe

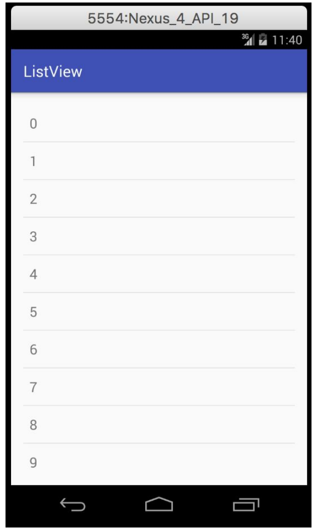
1. ~~Create a **ListView** view~~
2. ~~Create a **row** layout (or use existing one)~~
3. ~~Create **data** object list~~
4. Create an **Adapter**
5. **Bind** Adapter to the ListView

Cool stuff should never work alone

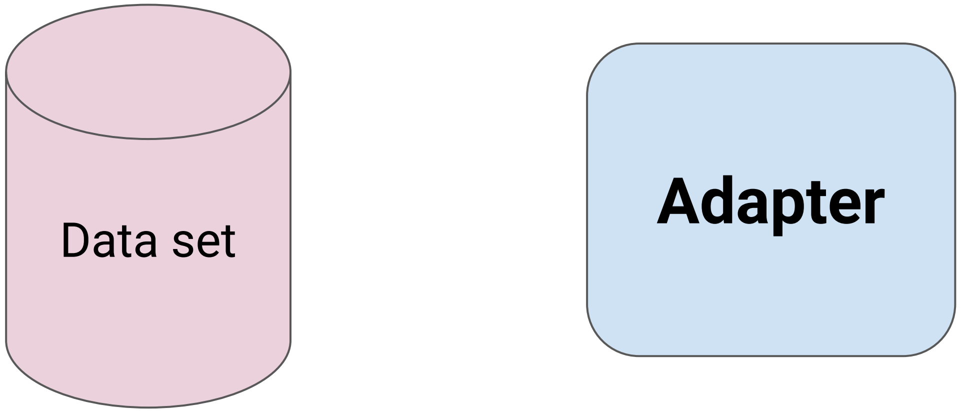
1.0 The “Holy” Trio



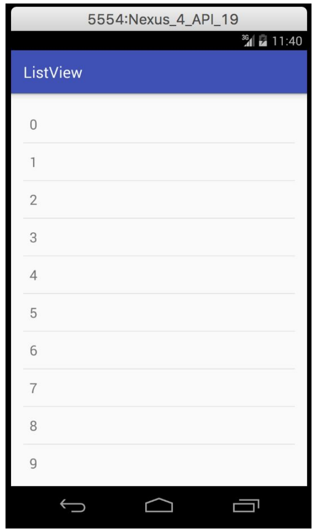
ListView (AdapterView)



1.0 The “Holy” Trio



ListView (AdapterView)



1.0 Adapters

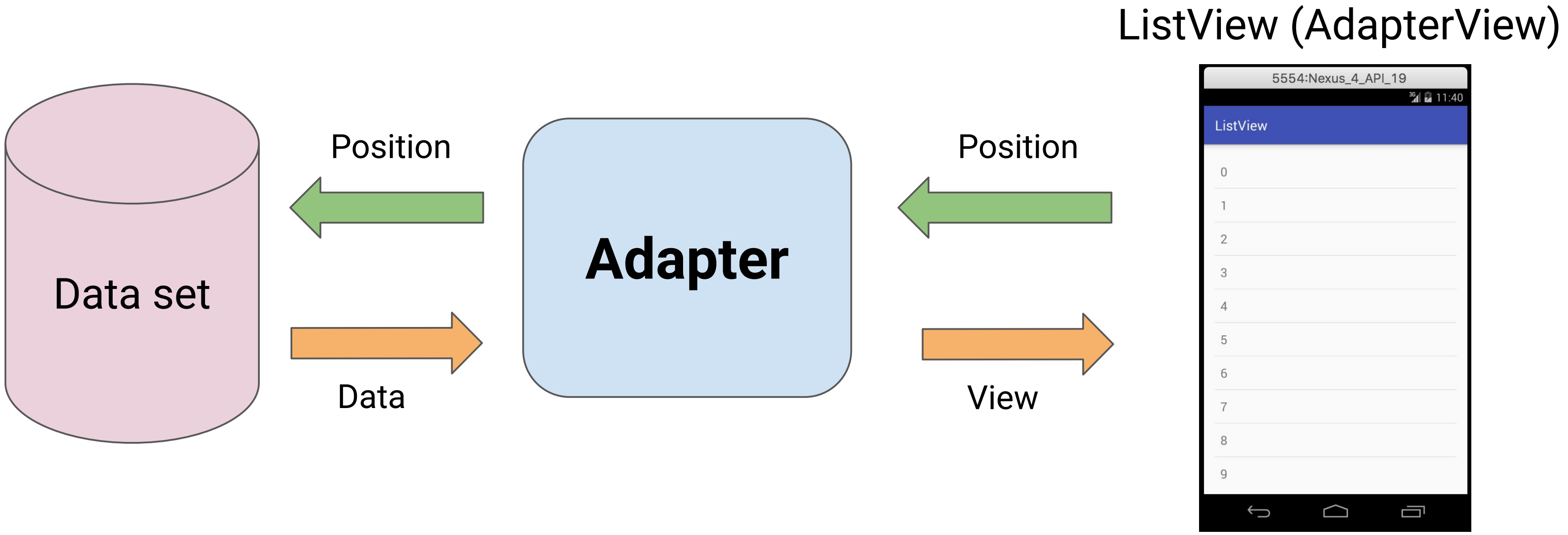
Adapter interface

BaseAdapter
Base class of common implementation for an Adapter

ArrayAdapter<T>
Uses array as a data source

CursorAdapter
Uses Cursor as a data source

1.0 The “Holy” Trio



1.0 Adapters

```
private List<MovieModel> loadMovies() {

      List<MovieModel> movies = new ArrayList<>(9);

      MovieModel movie1 = new MovieModel();
      MovieModel movie2 = new MovieModel();
      ...
      movie1.setName("Jurassic World - Fallen Kingdom");
      movie2.setName("The Meg");
      ...
      movie1.setImageRes(R.drawable.jurassic_world_fallen_kingdom);
      movie2.setImageRes(R.drawable.the_meg);
      ...
      movie1.setOverview("Three years after…");
      movie2.setOverview("A deep …");

      movies.add(movie1);
      movies.add(movie2);

      return movies
  }
```

1.0 Adapters

```
private List<MovieModel> loadMovies() {

      List<MovieModel> movies = new ArrayList<>(9);

      MovieModel movie1 = new MovieModel();
      MovieModel movie2 = new MovieModel();
      ...
      movie1.setName("Jurassic World - Fallen Kingdom");
      movie2.setName("The Meg");
      ...
      movie1.setImageRes(R.drawable.jurassic_world_fallen_kingdom);
      movie2.setImageRes(R.drawable.the_meg);
      ...
      movie1.setOverview("Three years after…");
      movie2.setOverview("A deep …");

      movies.add(movie1);
      movies.add(movie2);

      return movies
  }
```

1.0 Adapters

```
private List<MovieModel> loadMovies() {

      List<MovieModel> movies = new ArrayList<>(9);

      MovieModel movie1 = new MovieModel();
      MovieModel movie2 = new MovieModel();
      ...
      movie1.setName("Jurassic World - Fallen Kingdom");
      movie2.setName("The Meg");
      ...
      movie1.setImageRes(R.drawable.jurassic_world_fallen_kingdom);
      movie2.setImageRes(R.drawable.the_meg);
      ...
      movie1.setOverview("Three years after…");
      movie2.setOverview("A deep …");

      movies.add(movie1);
      movies.add(movie2);

      return movies
  }
```

1.0 Adapters

```
private List<MovieModel> loadMovies() {

      List<MovieModel> movies = new ArrayList<>(9);

      MovieModel movie1 = new MovieModel();
      MovieModel movie2 = new MovieModel();
      ...
      movie1.setName("Jurassic World - Fallen Kingdom");
      movie2.setName("The Meg");
      ...
      movie1.setImageRes(R.drawable.jurassic_world_fallen_kingdom);
      movie2.setImageRes(R.drawable.the_meg);
      ...
      movie1.setOverview("Three years after…”);
      movie2.setOverview("A deep …”);

      movies.add(movie1);
      movies.add(movie2);

      return movies
  }
```

1.0 Adapters

```
private List<MovieModel> loadMovies() {

      List<MovieModel> movies = new ArrayList<>(9);

      MovieModel movie1 = new MovieModel();
      MovieModel movie2 = new MovieModel();
      ...
      movie1.setName("Jurassic World - Fallen Kingdom");
      movie2.setName("The Meg");
      ...
      movie1.setImageRes(R.drawable.jurassic_world_fallen_kingdom);
      movie2.setImageRes(R.drawable.the_meg);
      ...
      movie1.setOverview("Three years after…”);
      movie2.setOverview("A deep …”);

      movies.add(movie1);
      movies.add(movie2);

      return movies
  }
```

1.0 Adapters

```
private List<MovieModel> loadMovies() {

      List<MovieModel> movies = new ArrayList<>(9);

      MovieModel movie1 = new MovieModel();
      MovieModel movie2 = new MovieModel();
      ...
      movie1.setName("Jurassic World - Fallen Kingdom");
      movie2.setName("The Meg");
      ...
      movie1.setImageRes(R.drawable.jurassic_world_fallen_kingdom);
      movie2.setImageRes(R.drawable.the_meg);
      ...
      movie1.setOverview("Three years after…");
      movie2.setOverview("A deep …");

      movies.add(movie1);
      movies.add(movie2);

      return movies
  }
```

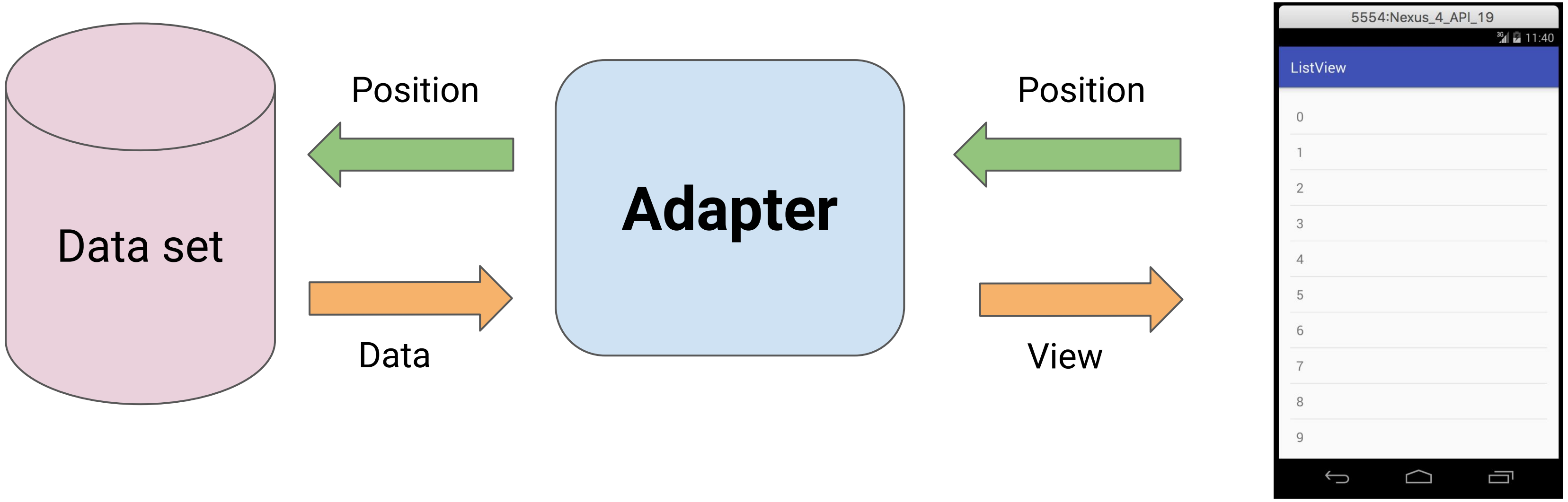
1.0 Adapters

```
private List<MovieModel> loadMovies() {

      List<MovieModel> movies = new ArrayList<>(9);

      MovieModel movie1 = new MovieModel();
      MovieModel movie2 = new MovieModel();
      ...
      movie1.setName("Jurassic World - Fallen Kingdom");
      movie2.setName("The Meg");
      ...
      movie1.setImageRes(R.drawable.jurassic_world_fallen_kingdom);
      movie2.setImageRes(R.drawable.the_meg);
      ...
      movie1.setOverview("Three years after…");
      movie2.setOverview("A deep …");

      movies.add(movie1);
      movies.add(movie2);

      return movies
  }
```

1.0 The "Holy" Trio

`List<MovieModel>`

ListView (AdapterView)



1.2 - BaseAdapter

- So.. Let’s Write Code!

1.2 - BaseAdapter

```
public class MoviesBaseAdapter extends BaseAdapter {

     private LayoutInflater mInflater;
     private ArrayList<MovieModel> mDataSource;

     public MoviesBaseAdapter(Context context, ArrayList<MovieModel> items) {
               mDataSource = items;
               mInflater = (LayoutInflater)context
                              .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
     }
}
```

1.2 - BaseAdapter

```
public class MoviesBaseAdapter extends BaseAdapter {

     private LayoutInflater mInflater;
     private ArrayList<MovieModel> mDataSource;

     public MoviesBaseAdapter(Context context, ArrayList<MovieModel> items) {
                mDataSource = items;
                mInflater = (LayoutInflater)context
                               .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
     }
}
```

1.2 - BaseAdapter

```
public class MoviesBaseAdapter extends BaseAdapter {

    private LayoutInflater mInflater;




}
```

1.2 - BaseAdapter

```
public class MoviesBaseAdapter extends BaseAdapter {

    @Override
    public int getCount() {
         return mDataSource.size();
    }

    @Override
    public MovieModel getItem(int position) {
         return mDataSource.get(position);
    }

    @Override
    public int getItemId(int position) {
         return position;
    }

}
```

1.2 - BaseAdapter

```java
public class MoviesBaseAdapter extends BaseAdapter {

     @Override
     public int getCount() {
          return mDataSource.size();
     }

     @Override
     public MovieModel getItem(int position) {
          return mDataSource.get(position);
     }

     @Override
     public int getItemId(int position) {
          return position;
     }

}
```

1.2 - BaseAdapter

```java
public class MoviesBaseAdapter extends BaseAdapter {

     @Override
     public int getCount() {
          return mDataSource.size();
     }

     @Override
     public MovieModel getItem(int position) {
          return mDataSource.get(position);
     }

     @Override
     public int getItemId(int position) {
          return position;
     }

}
```

1.2 - BaseAdapter

```
public class MoviesBaseAdapter extends BaseAdapter {
     . . .

     @Override
     public View getView(int position, View convertView, ViewGroup parent) {
          View rowItem = mInflater.inflate(R.layout.item_movie, parent, false);
          return rowItem;
     }

}
```

getView() Called for every row!

getView()

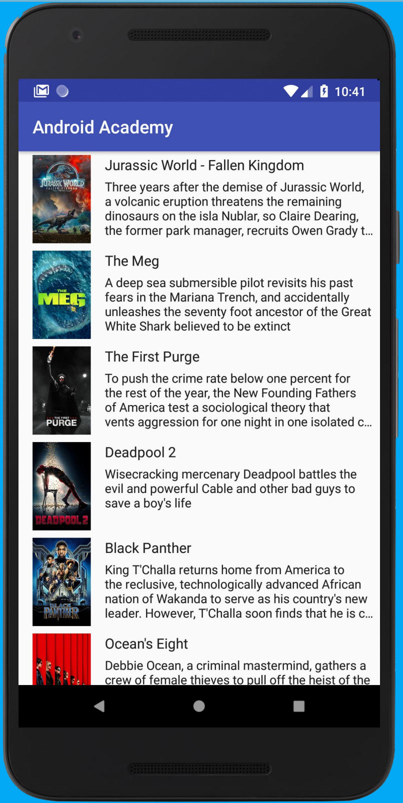
getView()

getView()

getView()

getView()

getView()



Jurassic World - Fallen Kingdom
Three years after the demise of Jurassic World, a volcanic eruption threatens the remaining dinosaurs on the isla Nublar, so Claire Dearing, the former park manager, recruits Owen Grady t...

The Meg
A deep sea submersible pilot revisits his past fears in the Mariana Trench, and accidentally unleashes the seventy foot ancestor of the Great White Shark believed to be extinct

The First Purge
To push the crime rate below one percent for the rest of the year, the New Founding Fathers of America test a sociological theory that vents aggression for one night in one isolated c...

Deadpool 2
Wisecracking mercenary Deadpool battles the evil and powerful Cable and other bad guys to save a boy's life

Black Panther
King T'Challa returns home from America to the reclusive, technologically advanced African nation of Wakanda to serve as his country's new leader. However, T'Challa soon finds that he is c...

Ocean's Eight
Debbie Ocean, a criminal mastermind, gathers a crew of female thieves to pull off the heist of the

1.2 - BaseAdapter

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
     // Inflate our row and find our views!
     View rowItem = mInflater.inflate(R.layout.item_movie, parent, false);
     ImageView img = (ImageView) rowItem.findViewById(R.id.item_movie_iv);
     TextView title = (TextView) rowItem.findViewById(R.id.item_movie_tv_title);
     TextView overview = (TextView) rowItem.findViewById(R.id.item_movie_tv_overview);









     return rowItem;
}
```

1.2 - BaseAdapter

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
     // Inflate our row and find our views!
     View rowItem = mInflater.inflate(R.layout.item_movie, parent, false);
     ImageView img = (ImageView) rowItem.findViewById(R.id.item_movie_iv);
     TextView title = (TextView) rowItem.findViewById(R.id.item_movie_tv_title);
     TextView overview = (TextView) rowItem.findViewById(R.id.item_movie_tv_overview);









     return rowItem;
}
```

1.2 - BaseAdapter

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
     // Inflate our row and find our views!
     View rowItem = mInflater.inflate(R.layout.item_movie, parent, false);
     ImageView img = (ImageView) rowItem.findViewById(R.id.item_movie_iv);
     TextView title = (TextView) rowItem.findViewById(R.id.item_movie_tv_title);
     TextView overview = (TextView) rowItem.findViewById(R.id.item_movie_tv_overview);






     return rowItem;
}
```

1.2 - BaseAdapter

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
     // Inflate our row and find our views!
     View rowItem = mInflater.inflate(R.layout.item_movie, parent, false);
     ImageView img = (ImageView) rowItem.findViewById(R.id.item_movie_iv);
     TextView title = (TextView) rowItem.findViewById(R.id.item_movie_tv_title);
     TextView overview = (TextView) rowItem.findViewById(R.id.item_movie_tv_overview);


      // Getting the data for this specific row!
     MovieModel movie = getItem(position);
```

1.2 - BaseAdapter

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
     // Inflate our row and find our views!
     ..
     // Getting the data for this specific row!
     MovieModel movie = getItem(position);
     // Fill our views with our data!
     image.setImageResource(movieModel.getImageRes());
     title.setText(movieModel.getName());
     overview.setText(movieModel.getOverview());



     return rowItem;
}
```

1.2 - BaseAdapter

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
     // Inflate our row and find our views!
     ..
     // Getting the data for this specific row!
     MovieModel movie = getItem(position);
     // Fill our views with our data!
     image.setImageResource(movieModel.getImageRes());
     title.setText(movieModel.getName());
     overview.setText(movieModel.getOverview());



     return rowItem;
}
```

1.1 ListView Recipe

1. ~~Create a **ListView** view~~
2. ~~Create a **row** layout (or use existing one)~~
3. ~~Create **data** object list~~
4. ~~Create an **Adapter**~~
5. **Bind** Adapter to the ListView

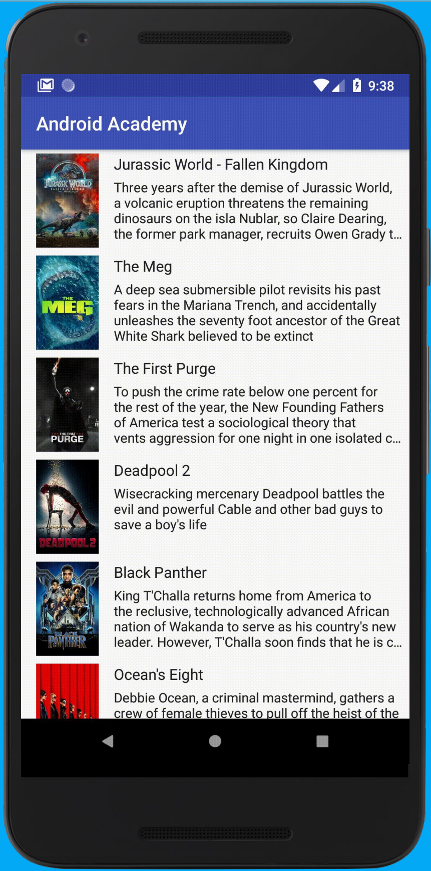
1.2 - BaseAdapter

```java
// In our Activity

MovieBaseAdapter adapter = new MovieBaseAdapter(this, dataSources);
mListView.setAdapter(adapter);
```

ListView Recipe

1. ~~Create a **ListView** view~~
2. ~~Create a **row** layout (or use existing one)~~
3. ~~Create **data** object list~~
4. ~~Create an **Adapter**~~
5. ~~**Bind** Adapter to the ListView~~

Let’s run it!



9:38

Android Academy

Jurassic World - Fallen Kingdom

Three years after the demise of Jurassic World, a volcanic eruption threatens the remaining dinosaurs on the isla Nublar, so Claire Dearing, the former park manager, recruits Owen Grady t...

The Meg

A deep sea submersible pilot revisits his past fears in the Mariana Trench, and accidentally unleashes the seventy foot ancestor of the Great White Shark believed to be extinct

The First Purge

To push the crime rate below one percent for the rest of the year, the New Founding Fathers of America test a sociological theory that vents aggression for one night in one isolated c...

Deadpool 2

Wisecracking mercenary Deadpool battles the evil and powerful Cable and other bad guys to save a boy's life

Black Panther

King T'Challa returns home from America to the reclusive, technologically advanced African nation of Wakanda to serve as his country's new leader. However, T'Challa soon finds that he is c...

Ocean's Eight

Debbie Ocean, a criminal mastermind, gathers a crew of female thieves to pull off the heist of the

1.3 The Big Problem

With small amounts of items.. It all feels good!

But...

With a long long array list..

1.3 The Big Problem

1.3 The Big Problem

Sooooooo

What makes our list slower?

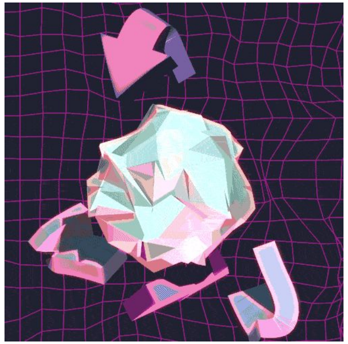
1.2 - BaseAdapter

```
public class MoviesBaseAdapter extends BaseAdapter {
     . . .

     @Override
     public View getView(int position, View convertView, ViewGroup parent) {
          View rowItem = mInflater.inflate(R.layout.item_movie, parent, false);
          ...
          return rowItem;
          }

     }
```

What are we doing today?

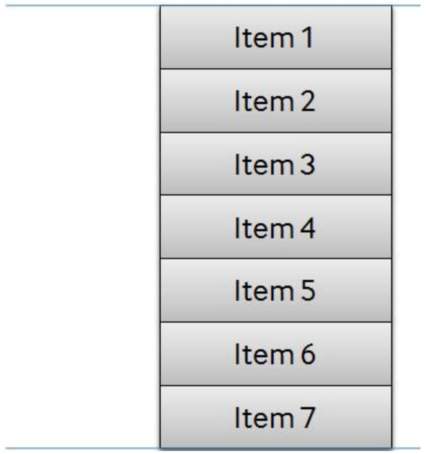
~~Introduction to ListView and Adapters~~

- Android view recycling
- GridView
- RecyclerView

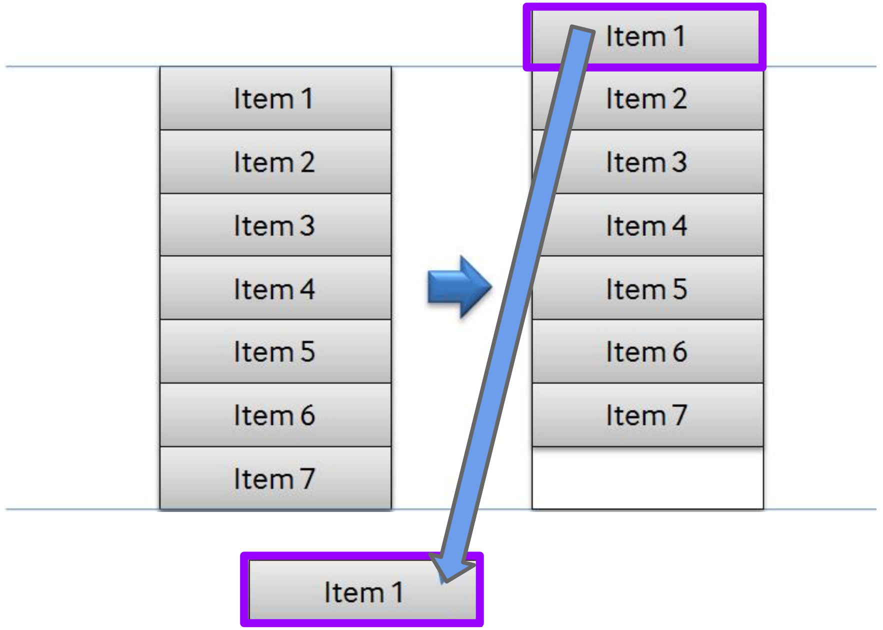


1.4 Recycling views and ViewHolder Pattern

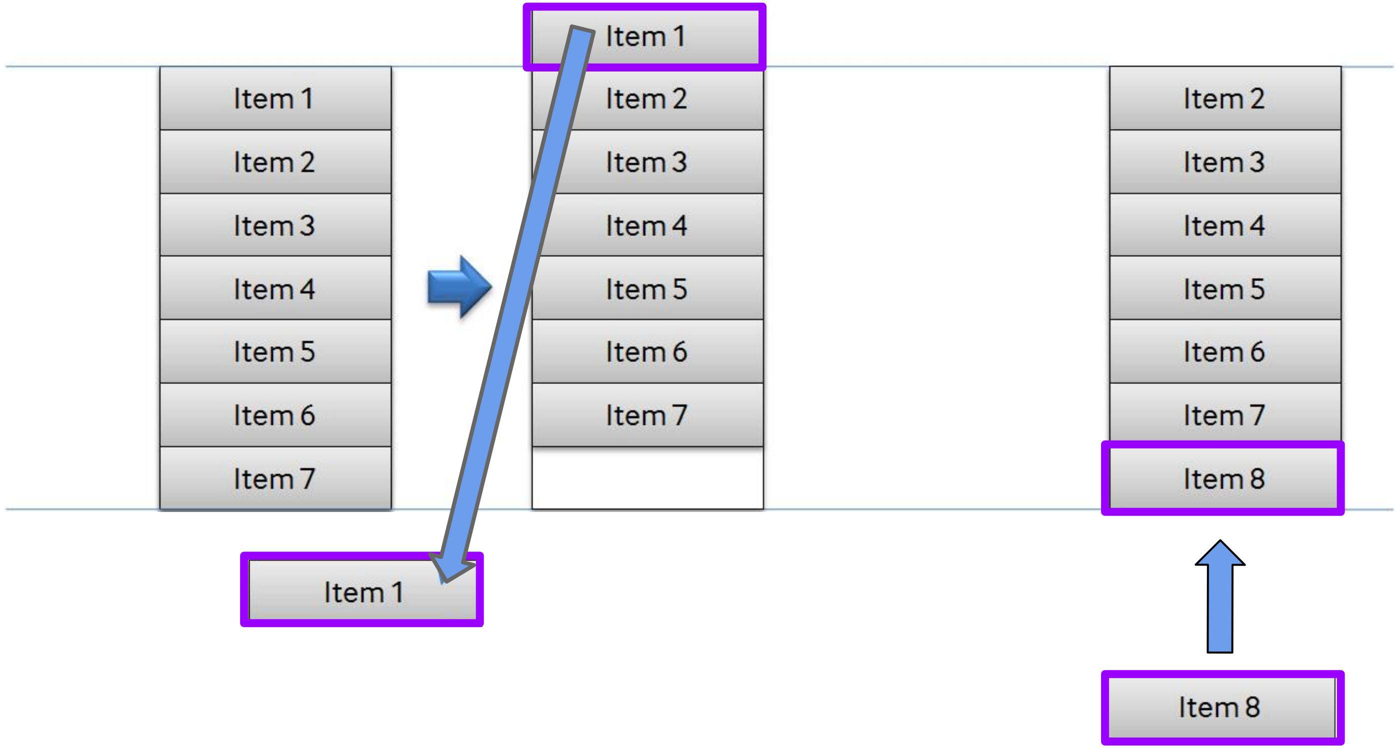
1.4 Recycling views and ViewHolder Pattern



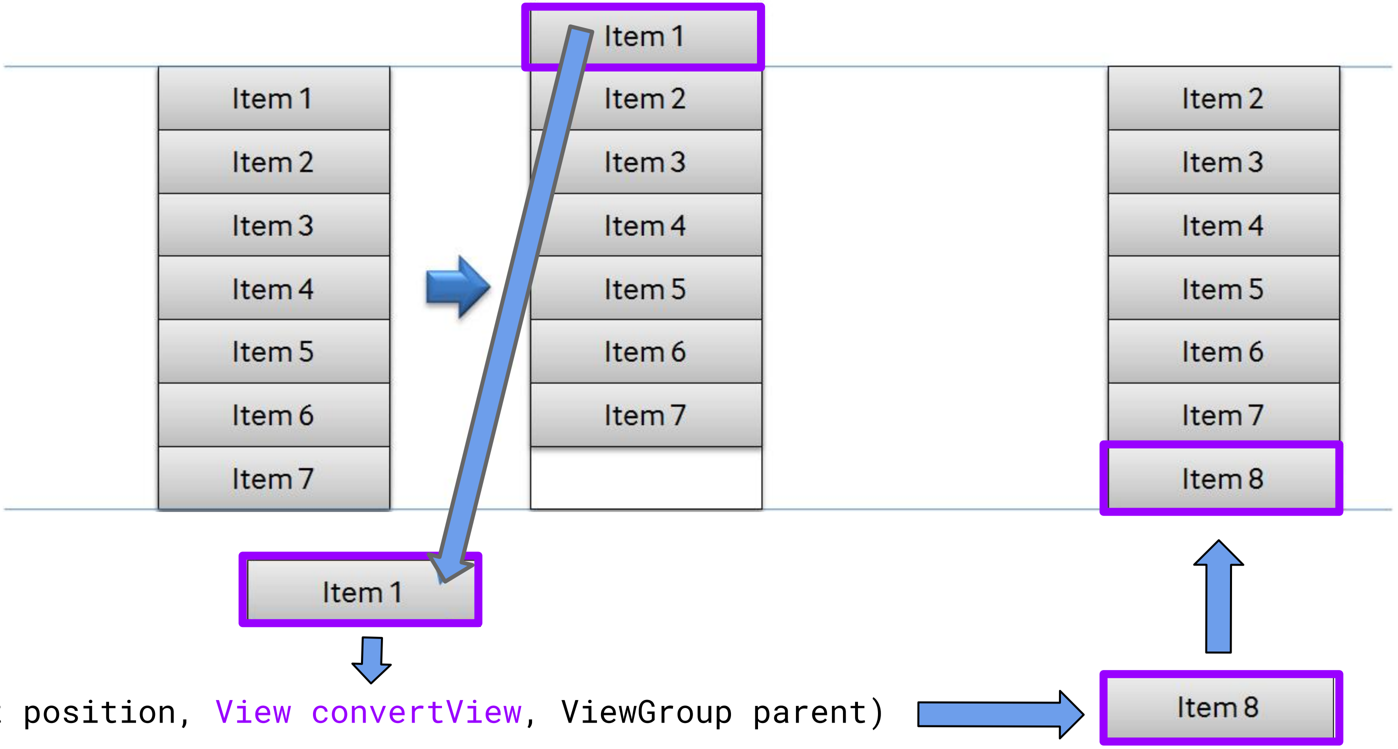
1.4 Recycling views and ViewHolder Pattern



1.4 Recycling views and ViewHolder Pattern



1.4 Recycling views and ViewHolder Pattern



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

1.2 - BaseAdapter

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    if (convertView == null) {
        convertView = mInflater.inflate(R.layout.item_movie, parent, false);
    }






}
```

1.2 - BaseAdapter

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    if (convertView == null) {
        convertView = mInflater.inflate(R.layout.item_movie, parent, false);
    }

    ImageView image = convertView.findViewById(R.id.item_movie_iv);
    TextView title  = convertView.findViewById(R.id.item_movie_tv_title);
    TextView description = convertView.findViewById(R.id.item_movie_tv_overview);
```

1.2 - BaseAdapter

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    if (convertView == null) {
        convertView = mInflater.inflate(R.layout.item_movie, parent, false);
    }

    ImageView image = convertView.findViewById(R.id.item_movie_iv);
    TextView title  = convertView.findViewById(R.id.item_movie_tv_title);
    TextView description = convertView.findViewById(R.id.item_movie_tv_overview);

    MovieModel movie = getItem(position);

    image.setImageResource(movie.getImageRes());
    title.setText(movie.getName());
    description.setText(movie.getOverview());


    return rowItem;

}
```

Let’s improve it
Even more!

1.2 - BaseAdapter

```java
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    if (convertView == null) {
        convertView = mInflater.inflate(R.layout.item_movie, parent, false);
    }

    ImageView image = convertView.findViewById(R.id.item_movie_iv);
    TextView title  = convertView.findViewById(R.id.item_movie_tv_title);
    TextView description = convertView.findViewById(R.id.item_movie_tv_overview);

    MovieModel movie = getItem(position);

    image.setImageResource(movie.getImageRes());
    title.setText(movie.getName());
    description.setText(movie.getOverview());


    return rowItem;

}
```

1.4 Recycling views and ViewHolder Pattern

```
public static class MovieViewHolder {
    public ImageView image;
    public TextView title, overview;
}
```

1.4 Recycling views and ViewHolder Pattern

```
public View getView(int position, View convertView, ViewGroup parent) {

            // Inflate our row and find our views!
            MovieViewHolder holder;
```

1.4 Recycling views and ViewHolder Pattern

```
public View getView(int position, View convertView, ViewGroup parent) {

            // Inflate our row and find our views!
            MovieViewHolder holder;
            if (convertView == null) {
                convertView = mInflater.inflate(R.layout.item_movie, parent, false);
                holder = new MovieViewHolder();
```

1.4 Recycling views and ViewHolder Pattern

```
public View getView(int position, View convertView, ViewGroup parent) {

            // Inflate our row and find our views!
            MovieViewHolder holder;
            if (convertView == null) {
                convertView = mInflater.inflate(R.layout.item_movie, parent, false);
                holder = new MovieViewHolder();
                holder.image = convertView.findViewById(R.id.item_movie_iv);
                holder.title = convertView.findViewById(R.id.item_movie_tv_title);
                holder.overview = convertView.findViewById(R.id.item_movie_tv_overview);
```

1.4 Recycling views and ViewHolder Pattern

```
public View getView(int position, View convertView, ViewGroup parent) {

            // Inflate our row and find our views!
            MovieViewHolder holder;
            if (convertView == null) {
                convertView = mInflater.inflate(R.layout.item_movie, parent, false);
                holder = new MovieViewHolder();
                holder.image = convertView.findViewById(R.id.item_movie_iv);
                holder.title = convertView.findViewById(R.id.item_movie_tv_title);
                holder.overview = convertView.findViewById(R.id.item_movie_tv_overview);
                convertView.setTag(holder);
```

1.4 Recycling views and ViewHolder Pattern

```
public View getView(int position, View convertView, ViewGroup parent) {

            // Inflate our row and find our views!
            MovieViewHolder holder;
            if (convertView == null) {
                convertView = mInflater.inflate(R.layout.item_movie, parent, false);
                holder = new MovieViewHolder();
                holder.image = convertView.findViewById(R.id.item_movie_iv);
                holder.title = convertView.findViewById(R.id.item_movie_tv_title);
                holder.overview = convertView.findViewById(R.id.item_movie_tv_overview);
                convertView.setTag(holder);
            } else {
                holder = (MovieViewHolder) convertView.getTag();

}
```

1.4 Recycling views and ViewHolder Pattern

```
public View getView(int position, View convertView, ViewGroup parent) {

            // Inflate our row and find our views!
            ..
            } else {
                holder = (MovieViewHolder) convertView.getTag();
            }

            // Getting the data for this specific row!
            MovieModel movie = getItem(position);

            holder.image.setImageResource(movie.getImageRes());
            holder.title.setText(movie.getName());
            holder.overview.setText(movie.getOverview());
}
```

1.4 Recycling views and ViewHolder Pattern

```
public View getView(int position, View convertView, ViewGroup parent) {

            // Inflate our row and find our views!

            // Getting the data for this specific row!
            MovieModel movie = getItem(position);

            holder.image.setImageResource(movie.getImageRes());
            holder.title.setText(movie.getName());
            holder.overview.setText(movie.getOverview());

            return convertView;
}
```

We’re displaying our data!!!

Efficiently!!!

1.2 - BaseAdapter - Small Refactor

```
public View getView(int position, View convertView, ViewGroup parent) {

            // Inflate our row and find our views!
            MovieViewHolder holder;
            if (convertView == null) {
                convertView = mInflater.inflate(R.layout.item_movie, parent, false);
                holder = new MovieViewHolder();
                holder.image = convertView.findViewById(R.id.item_movie_iv);
                holder.title = convertView.findViewById(R.id.item_movie_tv_title);
                holder.overview = convertView.findViewById(R.id.item_movie_tv_overview);
                convertView.setTag(holder);
            } else {
                holder = (MovieViewHolder) convertView.getTag();
            }
```

1.2 - BaseAdapter - Small Refactor

```
public View getView(int position, View convertView, ViewGroup parent) {

            // Inflate our row and find our views!
            MovieViewHolder holder;
            if (convertView == null) {
                convertView = onCreateViewHolder(convertView, parent);
            } else {
                holder = (MovieViewHolder) convertView.getTag();
            }
```

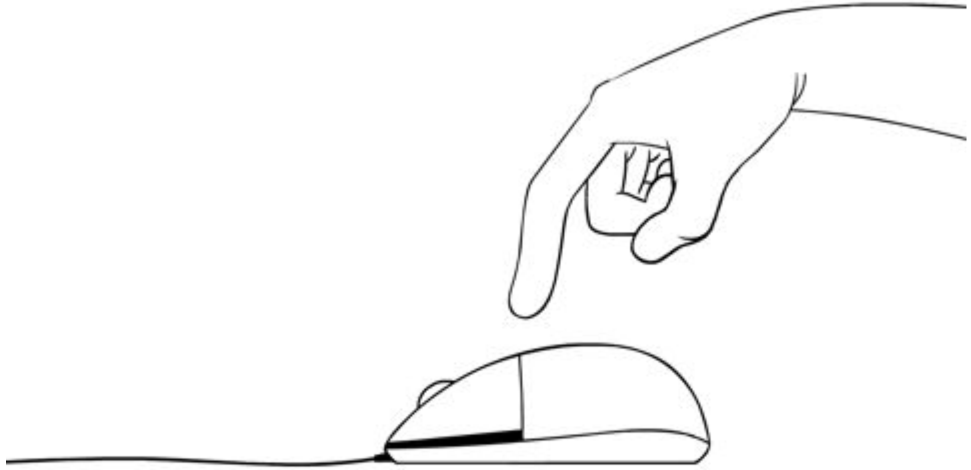
1.2 - BaseAdapter - Small Refactor

```
public View getView(int position, View convertView, ViewGroup parent) {

            // Inflate our row and find our views!

            // Getting the data for this specific row!
            MovieModel movie = getItem(position);

            holder.image.setImageResource(movie.getImageRes());
            holder.title.setText(movie.getName());
            holder.overview.setText(movie.getOverview());

            return convertView;
}
```

1.2 - BaseAdapter - Small Refactor

```
public View getView(int position, View convertView,
ViewGroup parent) {

            // Inflate our row and find our views!

            // Getting the data for this specific row!
            onBindViewHolder(holder, movie);

           return convertView;
}
```

1.5 User Interactions



1.5 User Interactions

ListView extends AdapterView, which makes our life easier.

We can just use the OnItemClick method of the AdapterView.

1.5 User Interactions

```java
mListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)     {
        Movie movie = mDataSources.get(position);
        // run your on click code
    }
});
```

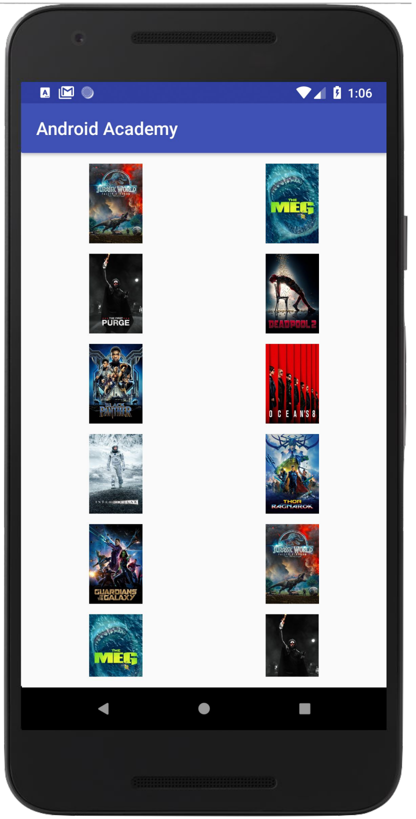
Any Questions?

What are we doing today?

- ~~Introduction to ListView and Adapters~~
- ~~Android view recycling~~
- GridView
- RecyclerView

So… ListView is cool and all…

But what if I want something like this?

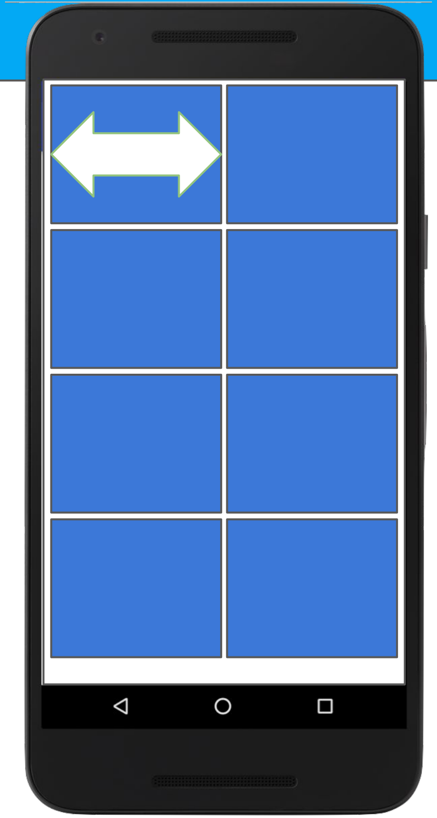


2.0 GridView

```
<GridView
    android:id=”@+id/grid_view”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent”
    android:gravity=”center”
    android:columnWidth=”150dp”
    android:numColumns=”auto_fit”
    android:verticalSpacing=”12dp”
    android:horizontalSpacing=”12dp”
    android:stretchMode=”spacingWidthUniform”/>
```

2.0 GridView

```
<GridView
    android:id="@+id/grid_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:columnWidth="100dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="12dp"
    android:horizontalSpacing="12dp"
    android:stretchMode="spacingWidthUniform"/>
```

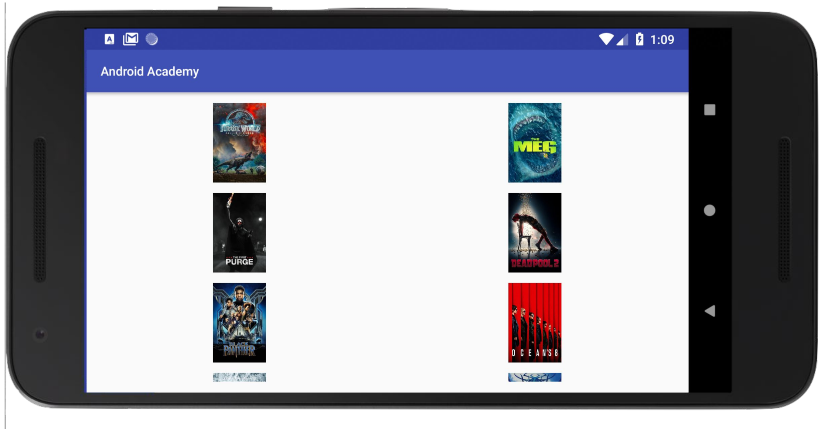


2.0 GridView

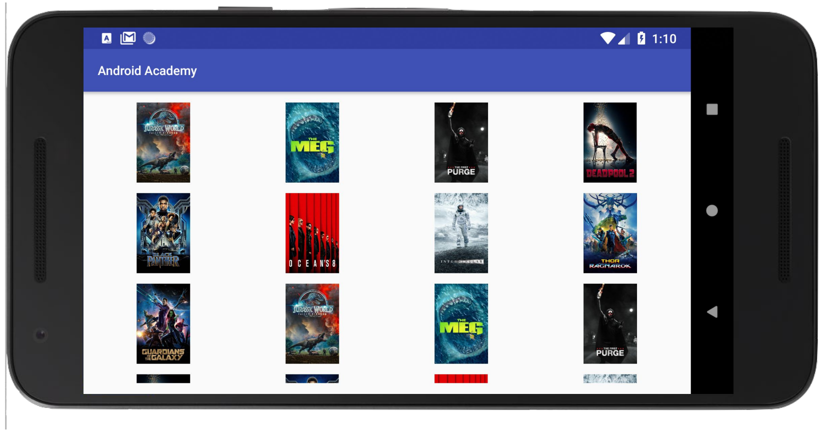
```
<GridView
    android:id="@+id/grid_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:columnWidth="100dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="12dp"
    android:horizontalSpacing="12dp"
    android:stretchMode="spacingWidthUniform"/>
```

2.0 GridView - numColumns Example

numColumns="2"

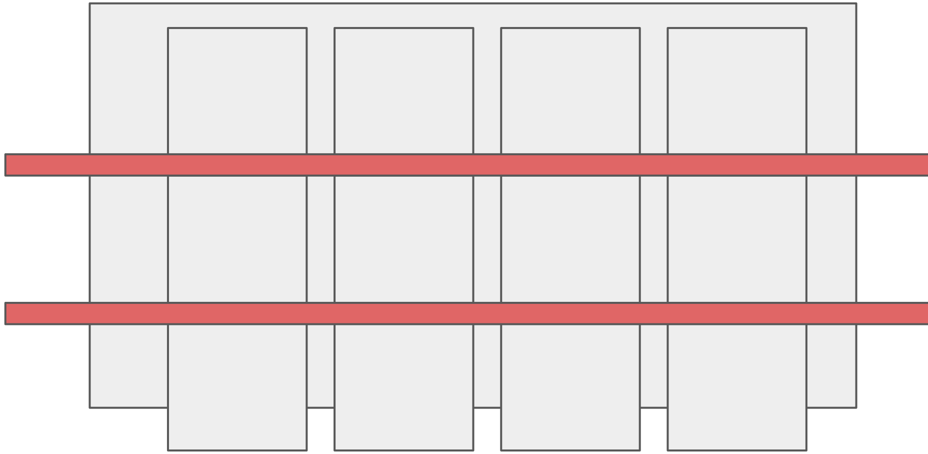


numColumns="auto_fit"



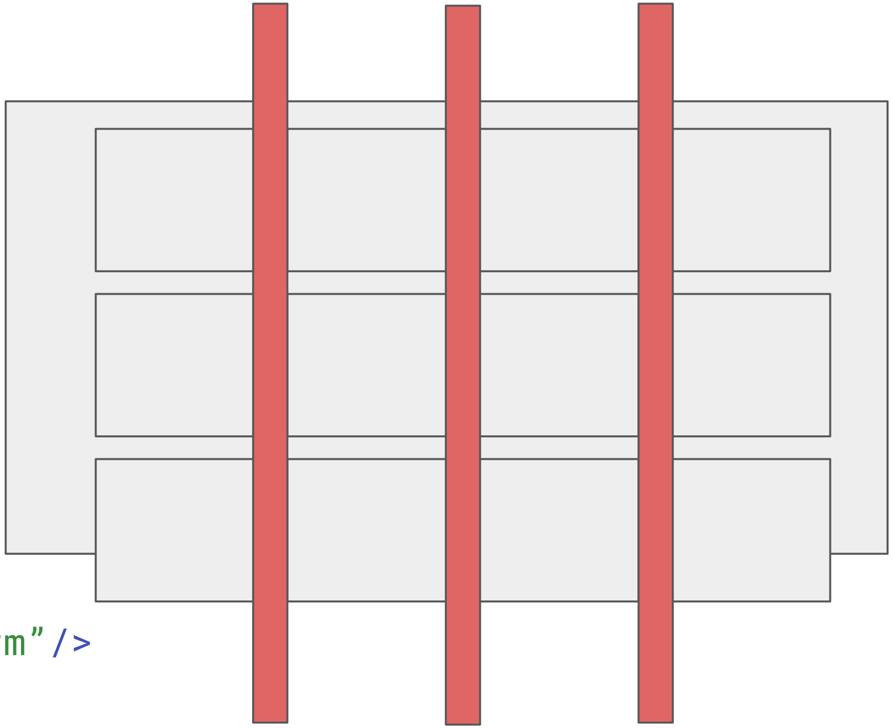
2.0 GridView

```
<GridView
    android:id="@+id/grid_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:columnWidth="100dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="12dp"
    android:horizontalSpacing="12dp"
    android:stretchMode="spacingWidthUniform"/>
```



2.0 GridView

```
<GridView
    android:id="@+id/grid_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:columnWidth="100dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="12dp"
    android:horizontalSpacing="12dp"
    android:stretchMode="spacingWidthUniform"/>
```

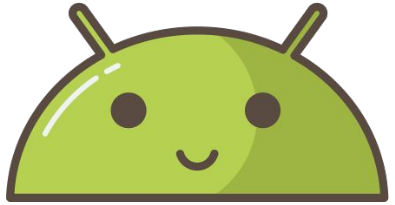


2.0 GridView

```
<GridView
    android:id=”@+id/grid_view”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent”
    android:gravity=”center”
    android:columnWidth=”100dp”
    android:numColumns=”auto_fit”
    android:verticalSpacing=”12dp”
    android:horizontalSpacing=”12dp”
    android:stretchMode=”spacingWidthUniform”/>
```

2.0 GridView

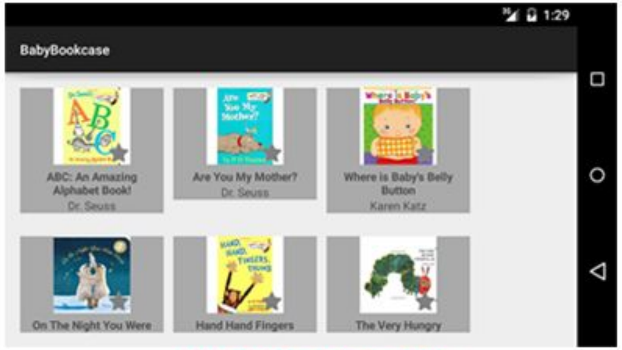
```
<GridView
    android:id=”@+id/grid_view”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent”
    android:gravity=”center”
    android:columnWidth=”@dimen/grid_column_width”
    android:numColumns=”auto_fit”
    android:verticalSpacing=”@dimen/grid_vertical_spacing”
    android:horizontalSpacing=”@dimen/grid_horizontal_spacing”
    android:stretchMode=”spacingWidthUniform”/>
```



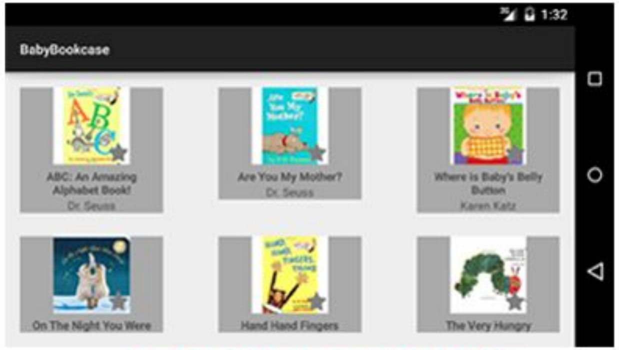
2.0 GridView

```
<GridView
    android:id="@+id/grid_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:columnWidth="@dimen/grid_column_width"
    android:numColumns="auto_fit"
    android:verticalSpacing="@dimen/grid_vertical_spacing"
    android:horizontalSpacing="@dimen/grid_horizontal_spacing"
    android:stretchMode="spacingWidthUniform"/>
```

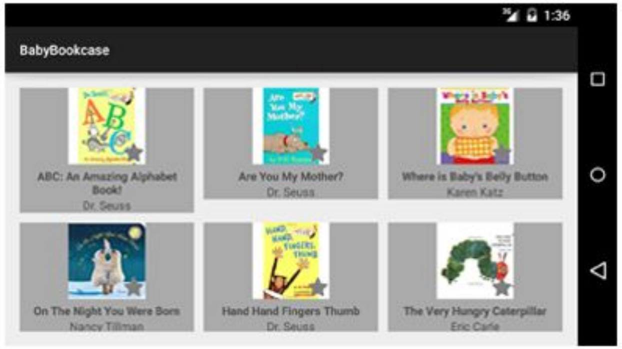
2.0 GridView - stretchMode Example



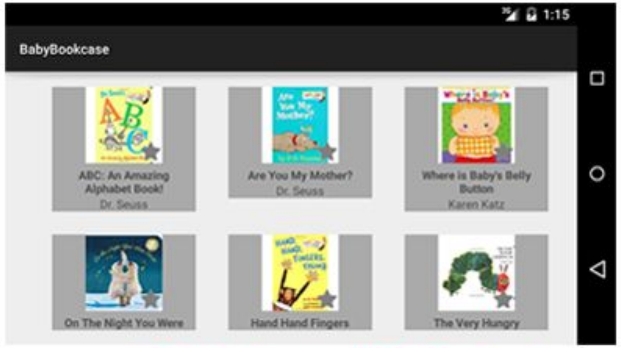
stretchMode="none"



stretchMode="spacingWidth"



stretchMode="columnWidth"



stretchMode="spacingWidthUniform"

2.0 GridView - Usage?

- Uses the same Adapters like a ListView
- All of the logic applies as well (ViewHolders, Data, etc..)

Any Questions?

What are we doing today?

- ~~Introduction to ListView and Adapters~~
- ~~Android view recycling~~
- ~~GridView~~
- RecyclerView

And still, It wasn’t enough

ListViews and GridViews have a few common problems:

1. It’s hard to add animations to them. Seriously hard.
2. It’s hard to make it look not like a list (or a grid).
3. It’s hard to add GestureDetection to it.
4. It’s easy **not** to use the ViewHolder pattern.

That’s why Google created the **RecyclerView**

http://developer.android.com/training/material/lists-cards.html
http://www.truiton.com/2015/03/android-recyclerview-vs-listview-comparison/

3.0 - RecyclerView

- Added in 2014 with Android 5.0 Lollipop
- More powerful and flexible
- Considered as a major enhancement over the good old ListView

3.0 - RecyclerView VS ListView

- ViewHolder Pattern

In ListView - it was recommended.

In RecyclerView - using it is mandatory using the RecyclerView.ViewHolder class.

3.0 - RecyclerView VS ListView

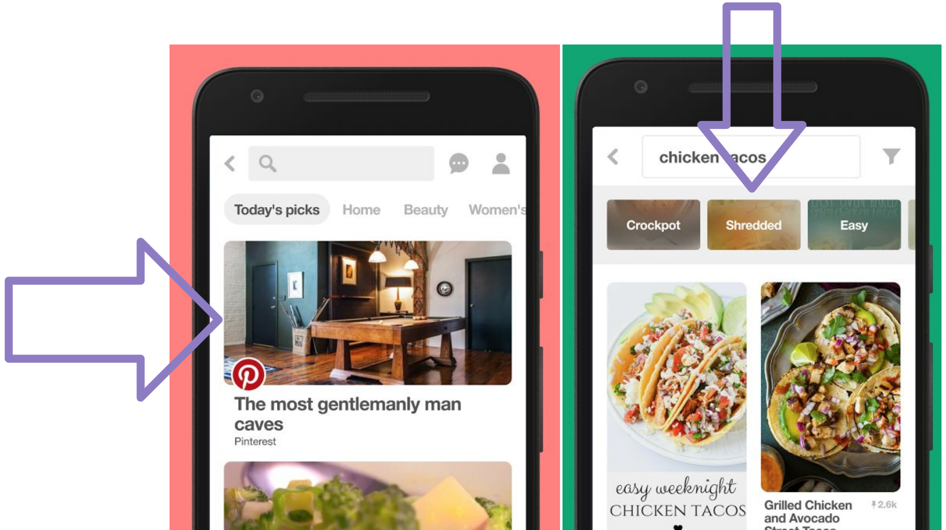
- LayoutManager

1. LinearLayoutManager - supports both Vertical/Horizontal lists
2. StaggeredGridLayoutManager - Pinterest like staggered lists
3. GridLayoutManager - supports grids as seen in Gallery apps

3.0 - RecyclerView VS ListView

- LayoutManager

LinearLayoutManager - supports both Vertical/Horizontal lists

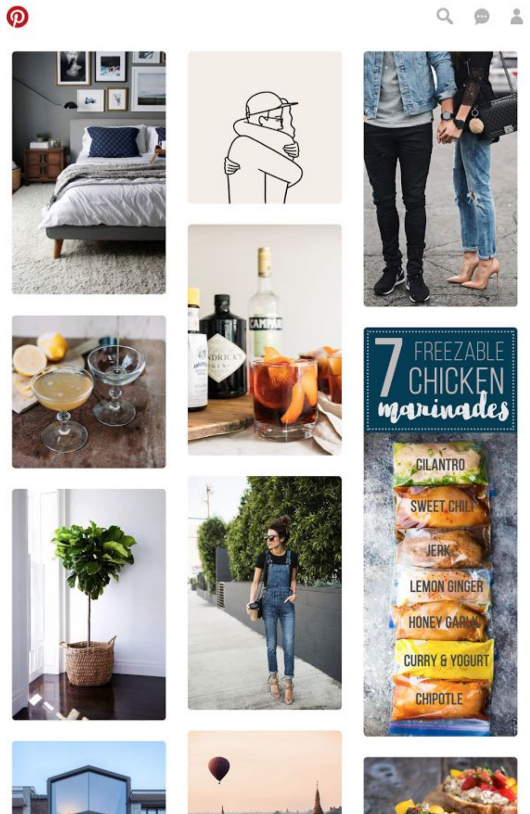


Today's picks
Home
Beauty
Women's
The most gentlemanly man caves
Pinterest
chicken tacos
Crockpot
Shredded
Easy
easy weeknight CHICKEN TACOS
Grilled Chicken and Avocado
2.6k

3.0 - RecyclerView VS ListView

- LayoutManager

StaggeredGridLayoutManager

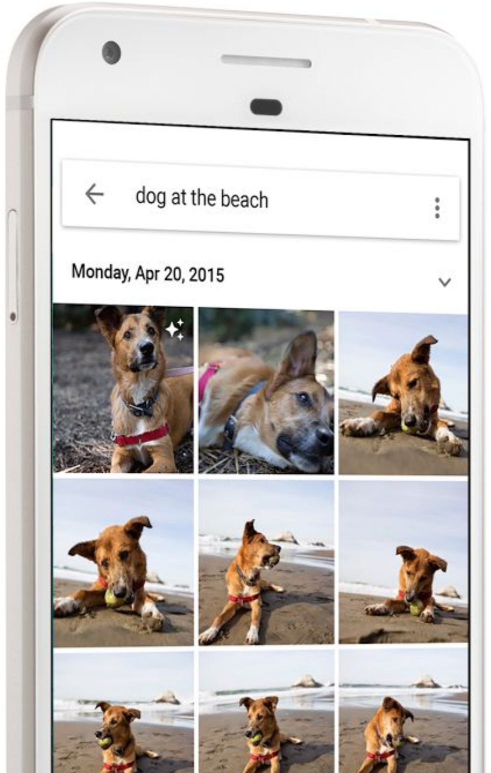


3.0 - RecyclerView VS ListView

- LayoutManager

GridLayoutManager - supports grids

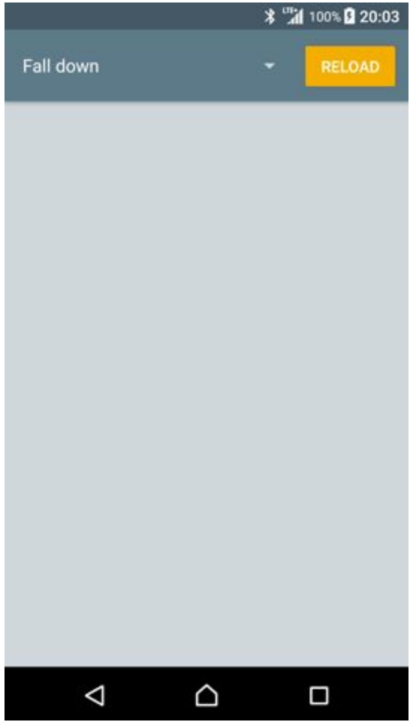
as seen in Gallery apps



3.0 - RecyclerView VS ListView

- Item Animator

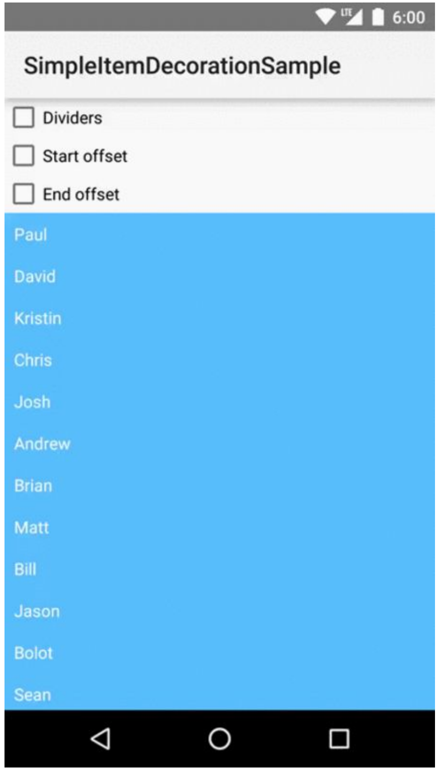
- In ListView - doesn’t exist (lacking in animation support)
- In RecyclerView - by using the ItemAnimator class,animations becomes a lot more easy and intuitive.



3.0 - RecyclerView VS ListView

- Item Decoration

- In ListView - dynamically decorating items like adding borders / dividers isn't easy.
- In RecyclerView - huge control for us developers but a bit more time consuming In terms of code


6:00

SimpleItemDecorationSample

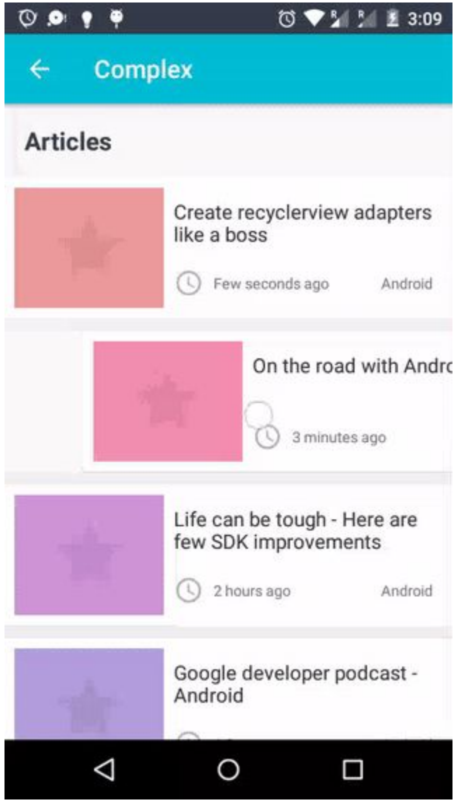
☐ Dividers
☐ Start offset
☐ End offset

Paul
David
Kristin
Chris
Josh
Andrew
Brian
Matt
Bill
Jason
Bolot
Sean

3.0 - RecyclerView VS ListView

- OnItemTouchListener

- In ListView - clicking is easy thanks to AdapterView.OnItemClick but very limited
- In RecyclerView - more power and control over touch gestures and not just clicks. Swipes, clicks, long clicks, drag and drops etc..



3:09
Complex
Articles
Create recyclerview adapters like a boss
Few seconds ago
Android
On the road with Andro
3 minutes ago
Life can be tough - Here are few SDK improvements
2 hours ago
Android
Google developer podcast - Android

~~ListView~~ RecyclerView Recipe

1. Create a ~~**ListView**~~ **RecyclerView** view
2. Create a **row layout**
3. Create **data** object list
4. Create a **View Holder** object
5. Create an **Adapter**
6. **Bind** Adapter to the ~~ListView~~ RecyclerView
7. Set **LayoutManager** to the RecyclerView

3.1 RecyclerView - Usage?

```
// Our app module gradle file


dependencies {
    implementation ‘com.android.support:recyclerview-v7:28.0.0'
}
```

3.1 RecyclerView

```
// In our XML

<android.support.v7.widget.RecyclerView
      android:id="@+id/recyclerView"
      android:layout_width="match_parent"
      android:layout_height="match_parent" />
```

~~ListView~~ RecyclerView Recipe

1. ~~Create a **ListView RecyclerView** view~~
2. Create a **row layout**
3. Create **data** object list
4. Create a **View Holder** object
5. Create an **Adapter**
6. **Bind** Adapter to the ~~ListView~~ RecyclerView
7. Set **LayoutManager** to the RecyclerView

~~ListView~~ RecyclerView Recipe

1. ~~Create a **ListView RecyclerView** view~~
2. ~~Create a **row layout**~~
3. Create **data** object list
4. Create a **View Holder** object
5. Create an **Adapter**
6. **Bind** Adapter to the ~~ListView~~ RecyclerView
7. Set **LayoutManager** to the RecyclerView

~~ListView~~ RecyclerView Recipe

1. ~~Create a **~~ListView~~ RecyclerView** view~~
2. ~~Create a **row layout**~~
3. ~~Create **data** object list~~
4. Create a **View Holder** object
5. Create an **Adapter**
6. **Bind** Adapter to the ~~ListView~~ RecyclerView
7. Set **LayoutManager** to the RecyclerView

3.1 RecyclerView.Adapter

```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder>{
             ..
             ..

       public class ViewHolder extends RecyclerView.ViewHolder{




       }
```

3.1 RecyclerView.Adapter

```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder>{
            ..
            ..

       public class ViewHolder extends RecyclerView.ViewHolder{




       }
```

3.1 RecyclerView.Adapter

```
public class ViewHolder extends RecyclerView.ViewHolder{

    public final ImageView ivImage;
    public final TextView tvTitle;
    public final TextView tvOverview;

    public ViewHolder(View view) {
        super(view);
        ivImage = view.findViewById(R.id.item_movie_iv);
        tvTitle = view.findViewById(R.id.item_movie_tv_title);
        tvOverview = view.findViewById(R.id.item_movie_tv_overview);
    }

   …
```

3.1 RecyclerView.Adapter

```
public class ViewHolder extends RecyclerView.ViewHolder{

      …
      public void onBindViewHolder(MovieModel movieModel) {
         ivImage.setImageResource(movieModel.getImageRes());
         tvTitle.setText(movieModel.getName());
         tvOverview.setText(movieModel.getOverview());
     }


   }
```

~~ListView~~ RecyclerView Recipe

1. ~~Create a **ListView RecyclerView** view~~
2. ~~Create a **row layout**~~
3. ~~Create **data** object list~~
4. ~~Create a **View Holder** object~~
5. Create an **Adapter**
6. **Bind** Adapter to the ~~ListView~~ RecyclerView
7. Set **LayoutManager** to the RecyclerView

3.1 RecyclerView.Adapter

```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {

        private LayoutInflater mInflater;
        private ArrayList<MovieModel> mDataSource;

        public MoviesViewAdapter(Context context, ArrayList<MovieModel> items) {
            mDataSource = items;
            mInflater = (LayoutInflater)context
                                  .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        }


        ...

}
```

3.1 RecyclerView.Adapter

```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {

        private LayoutInflater mInflater;
        private ArrayList<MovieModel> mDataSource;

        public MoviesViewAdapter(Context context, ArrayList<MovieModel> items) {
            mDataSource = items;
            mInflater = (LayoutInflater)context
                                   .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        }


        ...

}
```

3.1 RecyclerView.Adapter

```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {


            . . .

            @Override
            public int getItemCount() {
                return mDataSource.size();
            }

            . . .



}
```

3.1 RecyclerView.Adapter

```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {

        @Override
         public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
         …
         }

        @Override
        public void onBindViewHolder(final ViewHolder holder, int position) {
        …
        }
}
```

3.1 RecyclerView.Adapter

```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {

         ...

         @Override
         public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
             View view = mLayoutInflater.inflate(R.layout.item_movie, parent, false);
             return new ViewHolder(view);
         }

         ...

}
```

3.1 RecyclerView.Adapter

```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {

         ...

         @Override
         public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
             View view = mLayoutInflater.inflate(R.layout.item_movie, parent, false);
             return new ViewHolder(view);
         }

         ...

}
```

3.1 RecyclerView.Adapter

```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {

           ...

           @Override
           public void onBindViewHolder(final ViewHolder holder, int position) {
               holder.onBindViewHolder(movies.get(position));
           }

           …

}
```

~~ListView~~ RecyclerView Recipe

1. ~~Create a **ListView RecyclerView** view~~
2. ~~Create a **row layout**~~
3. ~~Create **data** object list~~
4. ~~Create a **View Holder** object~~
5. ~~Create an **Adapter**~~
6. **Bind** Adapter to the ~~ListView~~ RecyclerView
7. Set **LayoutManager** to the RecyclerView

All we need to do now is to configure our adapter in our activity :)

3.2 RecyclerView implementation

```
public class MoviesActivity extends Activity {
    private RecyclerView mRecyclerView;
    private RecyclerView.Adapter mAdapter;
    private RecyclerView.LayoutManager mLayoutManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_movies);
        initRecyclerView();
    }
}
```

3.2 RecyclerView implementation

```
private void initRecyclerView() {
    mRecyclerView = (RecyclerView) findViewById(R.id.recyclerView);
```

3.2 RecyclerView implementation

```
private void initRecyclerView() {
    mRecyclerView = (RecyclerView) findViewById(R.id.recyclerView);
    mLayoutManager = new LinearLayoutManager(this);
```

3.2 RecyclerView implementation

```
private void initRecyclerView() {
    mRecyclerView = (RecyclerView) findViewById(R.id.recyclerView);
    mLayoutManager = new LinearLayoutManager(this);
    mRecyclerView.setLayoutManager(mLayoutManager);
```

3.2 RecyclerView implementation

```
private void initRecyclerView() {
    mRecyclerView = (RecyclerView) findViewById(R.id.recyclerView);
    mLayoutManager = new LinearLayoutManager(this);
    mRecyclerView.setLayoutManager(mLayoutManager);
    mAdapter = new MoviesViewAdapter(this, mDataSource);
```

3.2 RecyclerView implementation

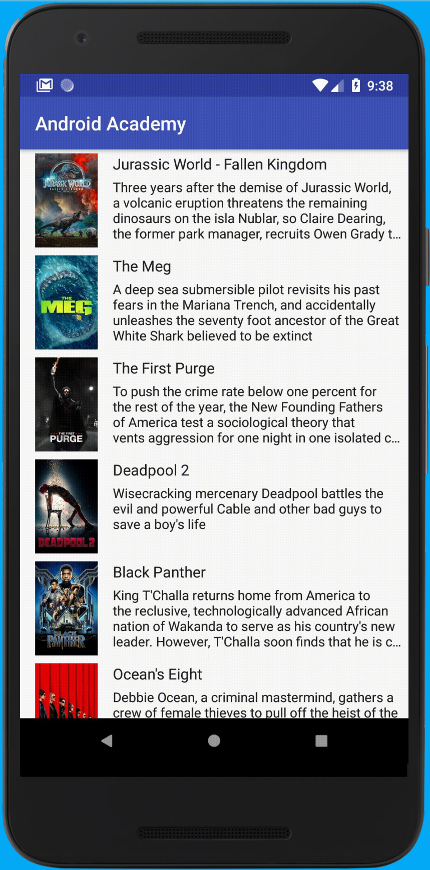
```
private void initRecyclerView() {
    mRecyclerView = (RecyclerView) findViewById(R.id.recyclerView);
    mLayoutManager = new LinearLayoutManager(this);
    mRecyclerView.setLayoutManager(mLayoutManager);
    mAdapter = new MoviesViewAdapter(this, mDataSource);
    mRecyclerView.setAdapter(mAdapter);
```

~~ListView~~ RecyclerView Recipe

1. ~~Create a **ListView RecyclerView** view~~
2. ~~Create a **row layout**~~
3. ~~Create **data** object list~~
4. ~~Create a **View Holder** object~~
5. ~~Create an **Adapter**~~
6. ~~**Bind** Adapter to the ListView RecyclerView~~
7. ~~Set **LayoutManager** to the RecyclerView~~

3.2 RecyclerView implementation

Now let's run this code…



9:38

Android Academy

Jurassic World - Fallen Kingdom

Three years after the demise of Jurassic World, a volcanic eruption threatens the remaining dinosaurs on the isla Nublar, so Claire Dearing, the former park manager, recruits Owen Grady t...

The Meg

A deep sea submersible pilot revisits his past fears in the Mariana Trench, and accidentally unleashes the seventy foot ancestor of the Great White Shark believed to be extinct

The First Purge

To push the crime rate below one percent for the rest of the year, the New Founding Fathers of America test a sociological theory that vents aggression for one night in one isolated c...

Deadpool 2

Wisecracking mercenary Deadpool battles the evil and powerful Cable and other bad guys to save a boy's life

Black Panther

King T'Challa returns home from America to the reclusive, technologically advanced African nation of Wakanda to serve as his country's new leader. However, T'Challa soon finds that he is c...

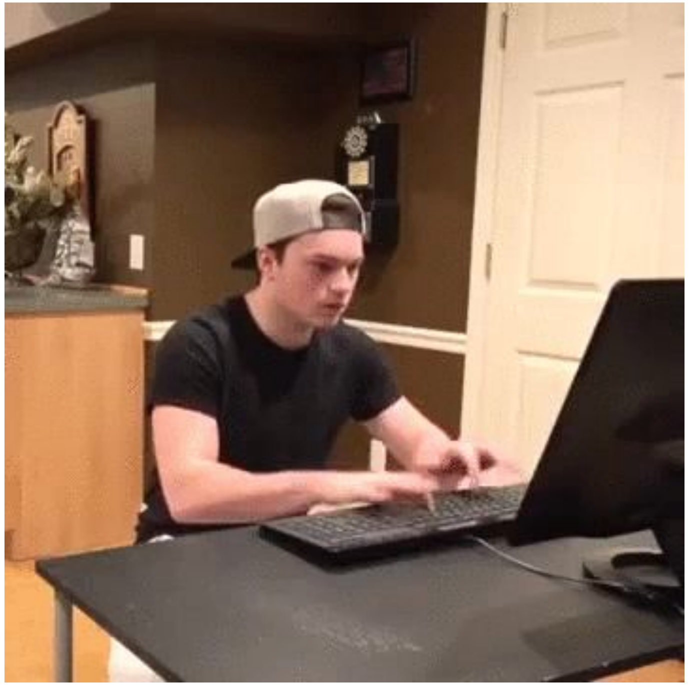
Ocean's Eight

Debbie Ocean, a criminal mastermind, gathers a crew of female thieves to pull off the heist of the

GIFSec.com

Any Questions?

Last thing, really...
homework link



Thank you.