JSON and RecyclerView

(again)

Erick Chang - 10/12/16
C4Q 3.3
https://goo.gl/a3kUdQ
https://github.com/ekchang/AC3.3/tree/master/lessons/json-rv

Objectives

- Who is JSON and why should I care
- What are POJOs and how does it relate to JSON
- Why should I love RecyclerView

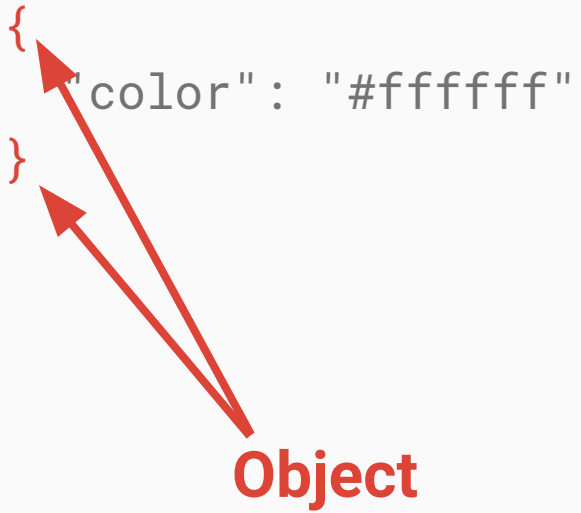
What is JSON?

- JavaScript Object Notation
- Keyword "*Notation*"

```
{
  "color": "#ffffff"
}
```

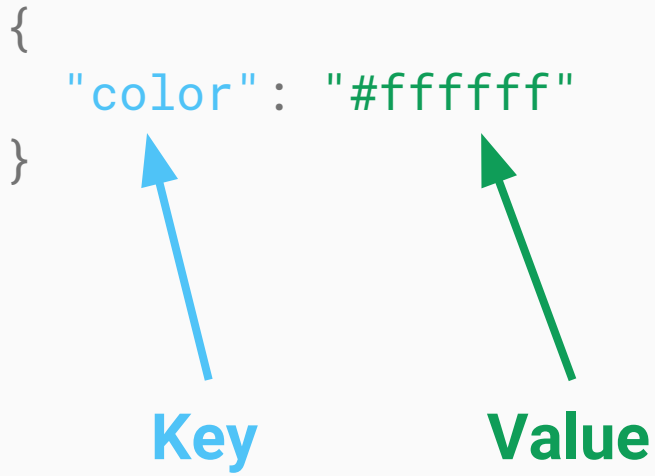
What is JSON?

- JavaScript Object Notation
- Keyword "*Notation*"



What is JSON?

- JavaScript Object Notation
- Keyword "*Notation*"



What is JSON?

- JavaScript Object Notation
- Keyword "*Notation*"

```
{
  "color": "#ffffff",
  "size": 12,
  "dimens": {
    "width": 4,
    "height": 6
  }
}
```

What is JSON?

- JavaScript Object Notation
- Keyword "*Notation*"

```
[{
  "color": "#ffffff"
},{
  "color": "#dcdcdc"
}]
```

What is JSON?

- JavaScript Object Notation
- Keyword "*Notation*"

Why do I need to know JSON? I barely know Java >:|

Phone

Internet

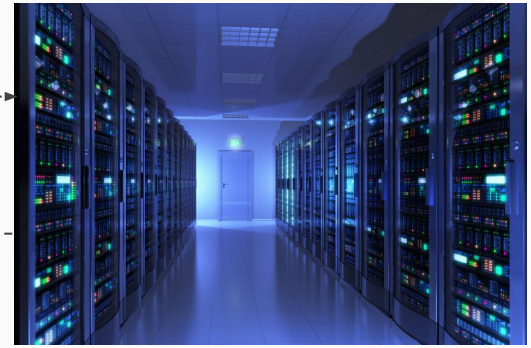
Server

JSON

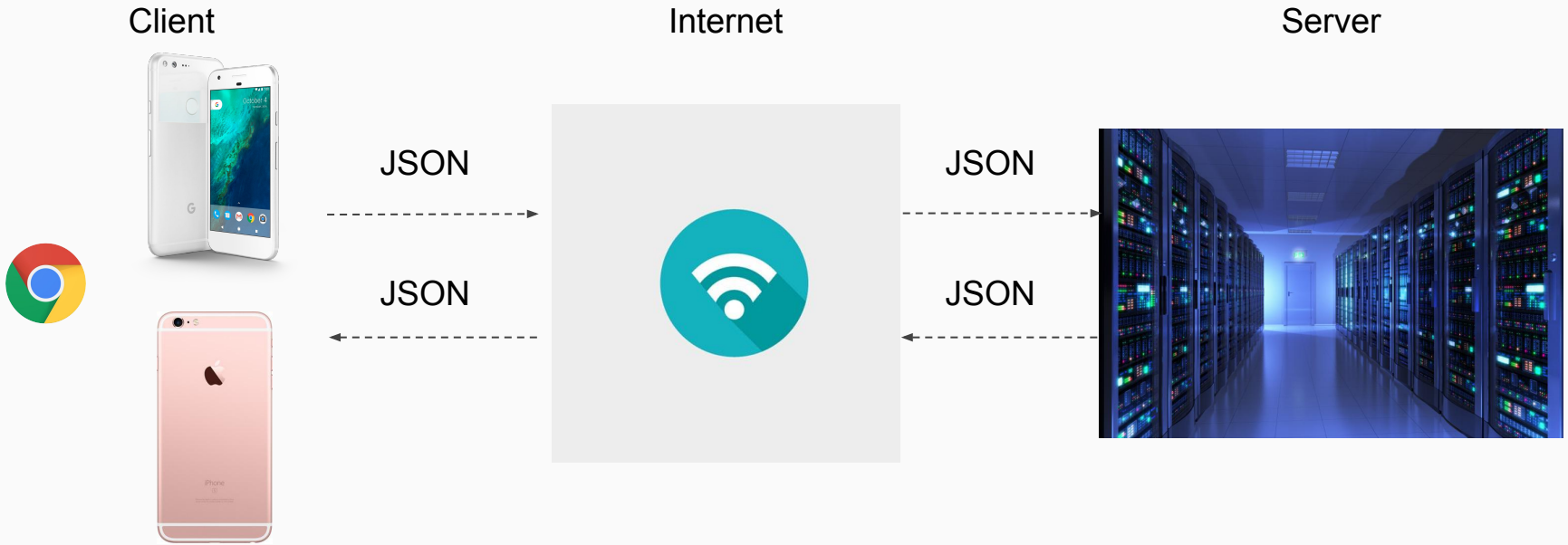
JSON

JSON

JSON



Why do I need to know JSON? I barely know Java >:|



Why do I need to know JSON? I barely know Java >:|

838 Bytes

```
public class Forecast {
  String cod;
  City city;
  ...
}

public class City {
  int id;
  String name;
}
```

539 Bytes


```
{
  "cod":"200","message":0.0032,
  "city":{"id":1851632,"name":"Shuzenji"
  "coord":{"lon":138.933334,"lat":34.966
  "country":"JP"},
  "cnt":10,
  "list":[{
    "dt":1406080800,
    "temp":{
      "day":297.77,
      "min":293.52,
      "max":297.77,
      "night":293.52,
      "eve":297.77,
      "morn":297.77},
    "pressure":925.04,
    "humidity":76,
    "weather":[{
      "id":803,
      "main":"Clouds",
      "description":"broken clouds",
      "icon":"04d"
    }],
  }
]}
```

Why do I need to know JSON? I barely know Java >:|

```
public class Forecast {
  String cod;

  public void setCod(String cod) {
    this.cod = cod;
  }

  public String getCod() {
    return cod;
  }
}
```

838 Bytes

4809 Bytes
(with getters/setters)

```
{
  "cod":"200","message":0.0032,
  "city":{"id":1851632,"name":"Shuzenji"
  "coord":{"lon":138.933334,"lat":34.966
  "country":"JP"},
  "cnt":10,
  "list":[{
    "dt":1406080800,
    "temp":{
      "day":297.77,
      "min":293.52,
      "max":297.77,
      "night":293.52,
      "eve":297.77,
      "morn":297.77},
    "pressure":925.04,
    "humidity":76,
    "weather":[{
      "id":803,
      "main":"Clouds",
      "description":"broken clouds",
      "icon":"04d"
    }],
  }
]}
```

539 Bytes

Why do I need to know JSON? I barely know Java >:|

- iOS / Web / NewHipOS can't necessarily read your Java code, but they understand JSON
- 4809 bytes vs 539 bytes (and that's just setters/getters and no real logic added yet)
- Have fun justifying using 9x more memory to send the same data
- Have fun wasting your data plan

What is a POJO?

- Plain Old Java Object

```
public class Color {
  int value;
}
```

A JSON Object (Simple)

```
class Color {
  int value;
}
```

```
{
  "value": "#ffffff"
}
```

A JSON Object - Object with multiple fields

```
public class Rectangle {
  String name;
  int area;
  Dimension dimens;
}

public class Dimension {
  int width;
  int height;
}
```

```
{
  "name": "Ricky",
  "area": 24,
  "dimens": {
    "width": 4,
    "height": 6
  }
}
```

A JSON Object - Array of Objects

```
public class Coordinate {
  float lat;
  float lon;
}

// Coordinate[] or
List<Coordinate>
```

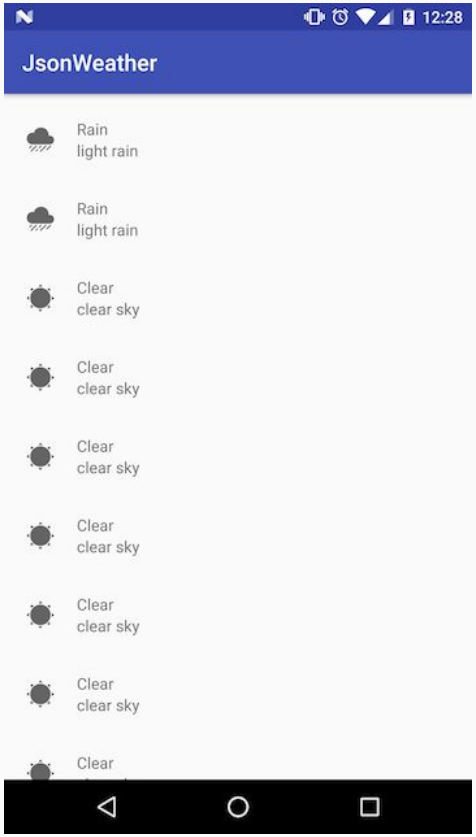
```
[{
    "lat": 40.285,
    "lon": 37.111
 }, {
    "lat": 55.532,
    "lon": 90.447
 }]
```

JSON Exercise

- Visit https://github.com/ekchang/JsonWeather
- Pull down git repo:
  - git clone https://github.com/ekchang/JsonWeather.git

RecyclerView (again)

RecyclerView - Graphic



RecyclerView

LayoutManager

(*How* do I want my children shown? List? Grid? Horizontally?)

Adapter

(*What* do I want to show?
What Views get updated based on my POJOs?)

ViewHolder

(What does just ONE View look like?)

RecyclerView

```
RecyclerView recyclerView = (RecyclerView) findViewById(...);

recyclerView.setLayoutManager(...)

recyclerView.setAdapter(...) // make your adapter first
```

- Only 1 Layout Manager
- Only 1 Adapter
- Must have both
- Neither can be reused for multiple RVs

LayoutManager

- Good news, you rarely need to write your own
- Use LinearLayoutManager, GridLayoutManager etc

```
recyclerView.setLayoutManager(new LinearLayoutManager(this));

// where "this" refers to your Context or Activity
```

Adapter

- Bad news, you almost always have to write your own Adapter and ViewHolder

```
class MyAdapter extends RecyclerView.Adapter<MyViewHolder> {
  MyViewHolder onCreateViewHolder(...) { ... }

  void onBindViewHolder(MyViewHolder vh, int position) {...}

  int getItemCount() {...}
}
```

Adapter

- Bad news, you almost always have to write your own Adapter and ViewHolder

```
class MyAdapter extends RecyclerView.Adapter<MyViewHolder> {
  MyViewHolder onCreateViewHolder(...) { ... }

  void onBindViewHolder(MyViewHolder vh, int position) {...}

  int getItemCount() {...}
}
```

Adapter - onCreateViewHolder

```
MyViewHolder onCreateViewHolder(ViewGroup parent, int type) {
  // Responsible for creating a new ViewHolder
  // You should inflate the layout here too
}
```

Adapter - onCreateViewHolder

```
MyViewHolder onCreateViewHolder(ViewGroup parent, int type) {
  // Responsible for creating a new ViewHolder
  // You should inflate the layout here too

  View childView = LayoutInflater.from(parent.getContext())
      .inflate(R.layout.item_layout, parent, false);
}
```

Adapter - onCreateViewHolder

```
MyViewHolder onCreateViewHolder(ViewGroup parent, int type) {
  // Responsible for creating a new ViewHolder
  // You should inflate the layout here too

  View childView = LayoutInflater.from(parent.getContext())
       .inflate(R.layout.item_layout, parent, false);
}
```

Adapter - onCreateViewHolder

```
MyViewHolder onCreateViewHolder(ViewGroup parent, int type) {
  // Responsible for creating a new ViewHolder
  // You should inflate the layout here too

  View childView = LayoutInflater.from(parent.getContext())
       .inflate(R.layout.item_layout, parent, false);

  return new ViewHolder(childView);
}
```

Adapter - onCreateViewHolder

```
MyViewHolder onCreateViewHolder(ViewGroup parent, int type) {
  // Responsible for creating a new ViewHolder
  // You should inflate the layout here too

  View childView = LayoutInflater.from(parent.getContext())
       .inflate(R.layout.item_layout, parent, false);

  return new MyViewHolder(childView);
}
```

MyViewHolder

```
public class MyViewHolder extends RecyclerView.ViewHolder {

  MyViewHolder(View itemView) {
    super(itemView);
  }
}
```

MyViewHolder

```
public class MyViewHolder extends RecyclerView.ViewHolder {
  ImageView image;

  MyViewHolder(View itemView) {
    super(itemView);

    image = (ImageView) itemView.findViewById(R.id.image_view);
  }
}
```

MyViewHolder

```
public class MyViewHolder extends RecyclerView.ViewHolder {
  ImageView image;
  TextView title;
  TextView desc;

  MyViewHolder(View itemView) {
    super(itemView);

    image = (ImageView) itemView.findViewById(R.id.image_view);
    title = (TextView) itemView.findViewById(R.id.title);
    desc = (TextView) itemView.findViewById(R.id.desc);
  }
}
```

Adapter - onBindViewHolder

```
void onBindViewHolder(MyViewHolder holder, int position) {
  // You should use a List or some data structure to hold the
  // order in which you want your POJOs to be displayed
}
```

- The parameter gives you the right view holder for the right position in the list
- Your job is to write how the holder should update its data
- **Think of "bind" as "update View" or "update this Row in the RecyclerView"**

Adapter - onBindViewHolder

```
void onBindViewHolder(MyViewHolder holder, int position) {
  // You should use a List or some data structure to hold the
  // order in which you want your POJOs to be displayed
  MyModel model = list.get(position);
}
```

Adapter - onBindViewHolder

```
void onBindViewHolder(MyViewHolder holder, int position) {
  // You should use a List or some data structure to hold the
  // order in which you want your POJOs to be displayed
  MyModel model = list.get(position);

  holder.image.setImageResource(model.getImageResource());
  holder.title.setText(model.getTitle());
  holder.desc.setText(model.getDescription());
}
```

MyAdapter - getItemCount

```
public int getItemCount() {
  // RecyclerView uses this method to know how many children you
have
  // You are responsible for telling RV that
}
```

MyAdapter - getItemCount

```
public int getItemCount() {
  return list.getSize();
}
```

RecyclerView Exercise

https://github.com/ekchang/AC3.3/tree/master/lessons/json-rv

If you already have the repo pulled down:

```
git checkout rv_exercise
```

Task:

Build a Weather app using a RecyclerView

- Use **MainActivity**
- Display an icon, title, description based on weather
- Use **WeatherHelper.getWeather(context)** to get your List<Weather> for now.

RecyclerView Exercise

- You will need to add the RecyclerView dependency:
  - File
  - Project Structure
  - under Modules click 'app'
  - Dependencies tab
  - + sign
  - Add Library dependency
  - Search for 'recyclerview'
- Or add **compile 'com.android.support:recyclerview-v7:24.2.1'** to root **build.gradle**

Resources

RecyclerView

- Watch Adam Powell / Yigit Boyar's Google I/O talk on RecyclerView: https://youtu.be/LqBlYJTfLP4
- Android guide on Cards/Lists has an RV example: https://developer.android.com/training/material/lists-cards.html

JSON Parsing (outside of current lecture)

- https://github.com/google/gson/blob/master/UserGuide.md

RecyclerView - Solution

If you are struggling with RV, you should checkout my solution branch and spend time *reading* the code that works.

Have two projects side by side and write out the same code by hand.

I cannot emphasize this enough.

```
git checkout soln_rv // solution
```

(if git says the branch doesn't exist, you need to use **git fetch** first)