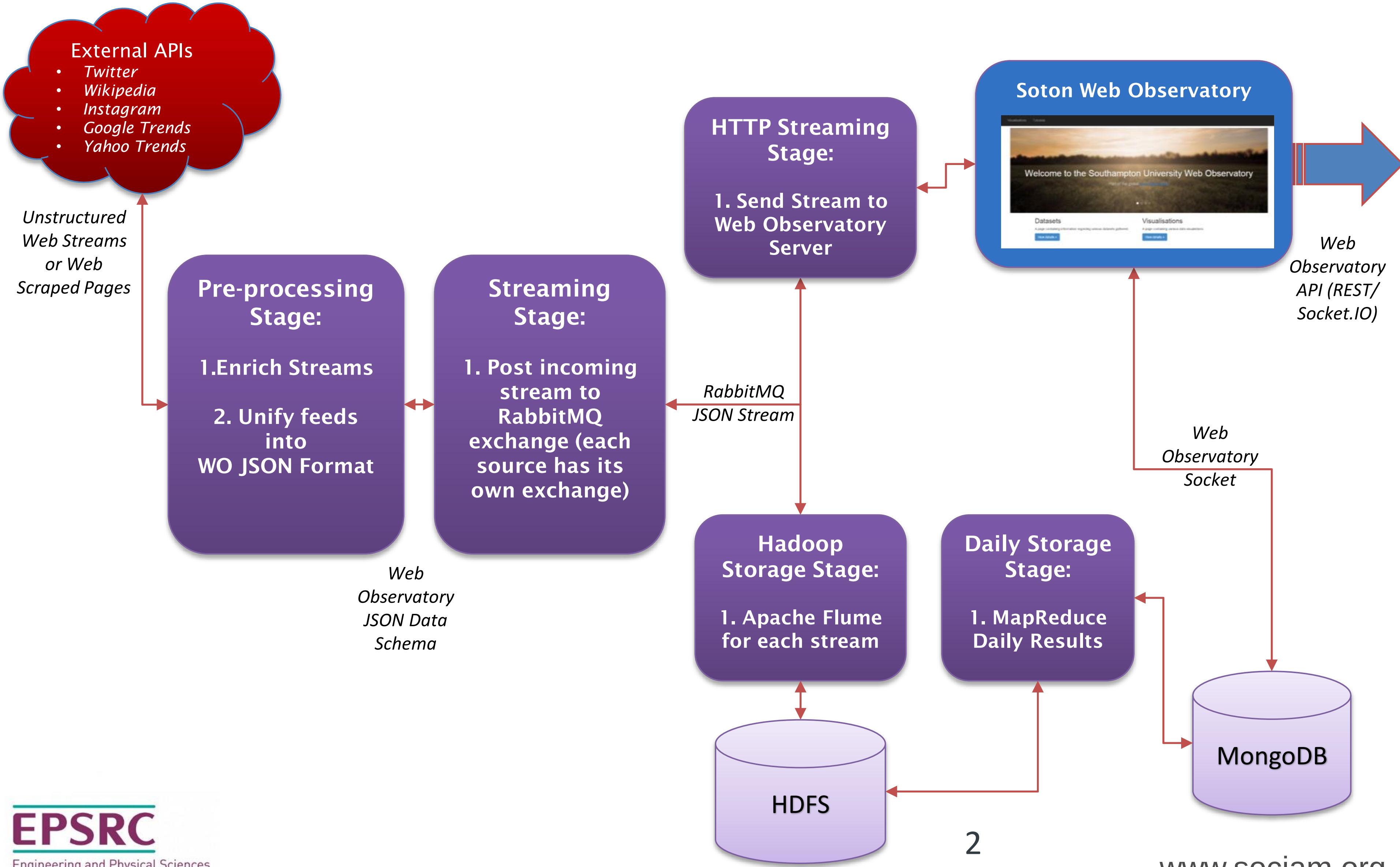
Web Observatory: *Components of an Application*

Ramine Tinati
@raminetinati

EPSRC
Engineering and Physical Sciences
Research Council

Web Observatory Architecture…



External APIs
- *Twitter*
- *Wikipedia*
- *Instagram*
- *Google Trends*
- *Yahoo Trends*

Unstructured Web Streams or Web Scraped Pages

Pre-processing Stage:

1.Enrich Streams

2. Unify feeds into WO JSON Format

Streaming Stage:

1. Post incoming stream to RabbitMQ exchange (each source has its own exchange)

Web Observatory JSON Data Schema

RabbitMQ JSON Stream

HTTP Streaming Stage:

1. Send Stream to Web Observatory Server

Soton Web Observatory

Welcome to the Southampton University Web Observatory

Datasets

Visualisations

Web Observatory API (REST/ Socket.IO)

Web Observatory Socket

Hadoop Storage Stage:

1. Apache Flume for each stream

Daily Storage Stage:

1. MapReduce Daily Results

HDFS

MongoDB

EPSRC
Engineering and Physical Sciences
Research Council

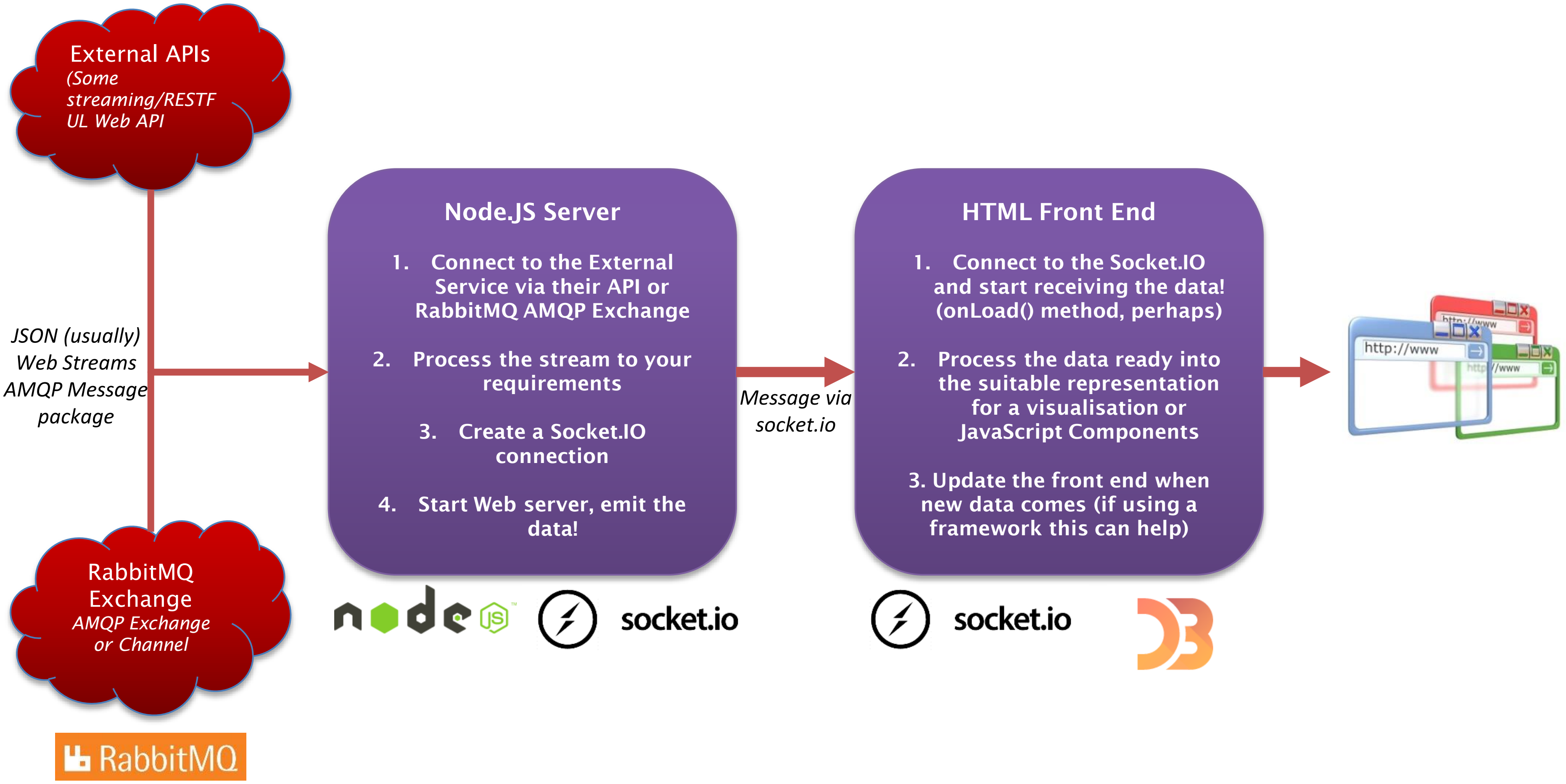
Making an App – What you Need

- **Backend**
  - Node.JS
  - RabbitMQ/AMQP
  - Socket.IO
- **Front End**
  - Socket.IO
  - D3.j3 (Or other Javascript Vis library)

EPSRC
Engineering and Physical Sciences
Research Council

The components of an App*

*This is only *one* way of doing it – experiment to find others!



External APIs
(Some streaming/RESTFUL Web API

JSON (usually) Web Streams AMQP Message package

RabbitMQ Exchange
AMQP Exchange or Channel

RabbitMQ

Node.JS Server

1. **Connect to the External Service via their API or RabbitMQ AMQP Exchange**
2. **Process the stream to your requirements**
3. **Create a Socket.IO connection**
4. **Start Web server, emit the data!**

node JS · socket.io

Message via socket.io

HTML Front End

1. **Connect to the Socket.IO and start receiving the data! (onLoad() method, perhaps)**
2. **Process the data ready into the suitable representation for a visualisation or JavaScript Components**
3. **Update the front end when new data comes (if using a framework this can help)**

socket.io · D3

*This is only *one* way of doing it – experiment to find others!

An Example – Twitter Stream, Node.JS and Socket.IO

- Fork from:
    - https://github.com/raminetinati/TwitterNodeSocketExample.git

- Provides simple template of how to use:
    - Twitter API (and Web Observatory API)
    - http
    - Node.JS
    - Socket.IO

Step One – Create the Backend

- Create the HTTP Server and Socket:

```
var app = require('http').createServer(handler);
var io = require('socket.io')(app);
```

- Create the Twitter Connection and setup the stream

```
var Twit = require('twit')
var T = new Twit({
    consumer_key:         'REQUIRED'
  , consumer_secret:      'REQUIRED'
  , access_token:         'REQUIRED'
  , access_token_secret:  'REQUIRED'
});


var stream = T.stream('statuses/sample');
```

EPSRC
Engineering and Physical Sciences
Research Council

Step Two – Start Sending the Data

- Start the Stream and emit to the socket

```
stream.on('tweet', function (tweet) {
  console.log(tweet);
io.emit('tweets',tweet);
});
```

- Now We’re ready to receive it on the front end….

Step Three - Receiving the Data on the Client

- Receive the data from the socket

```
var socket = io.connect(‘serverAddress');
socket.on('tweets', function (tweet) {
   var image_url = "";
         if(tweet.entities.media[0]){
          var url = tweet.entities.media[0];
              url = url.media_url;
              image_url = url;
       }

     // Here we can get some of the images in the Tweets
     if (image_url && image_url != "") {
       var div = $("#collageContainer");
       var img = new Image(image_url);
       img.onload = function () {
                div.append($("<img src='"+image_url+"' height='100%'>"));
                if (div.children().length > 25000) {
                   div.children()[0].remove();
                     }
       };
     }
  });
};
```

//Connect to the Stream….

//Do some processing of the stream

Step Four – Show some data in the browser

- Call the JavaScript method and add a div tag to an HTML Page

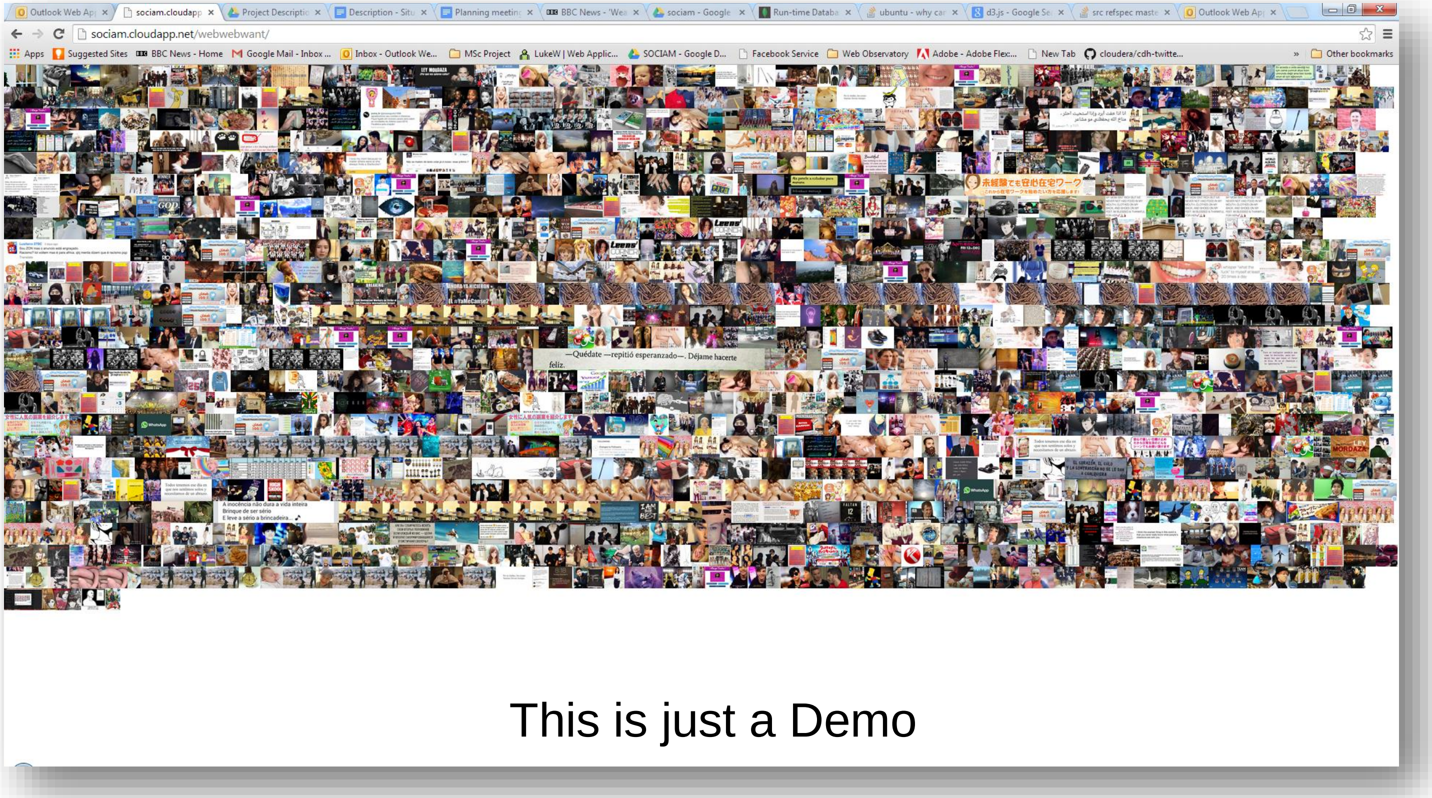
```
<body onload='main()'>

        <div id="collageContainer" class="thumbimg"> </div>

</body>
```

- And… You’re done☺

Step Five – Now Go Modify your Code!



This is just a Demo