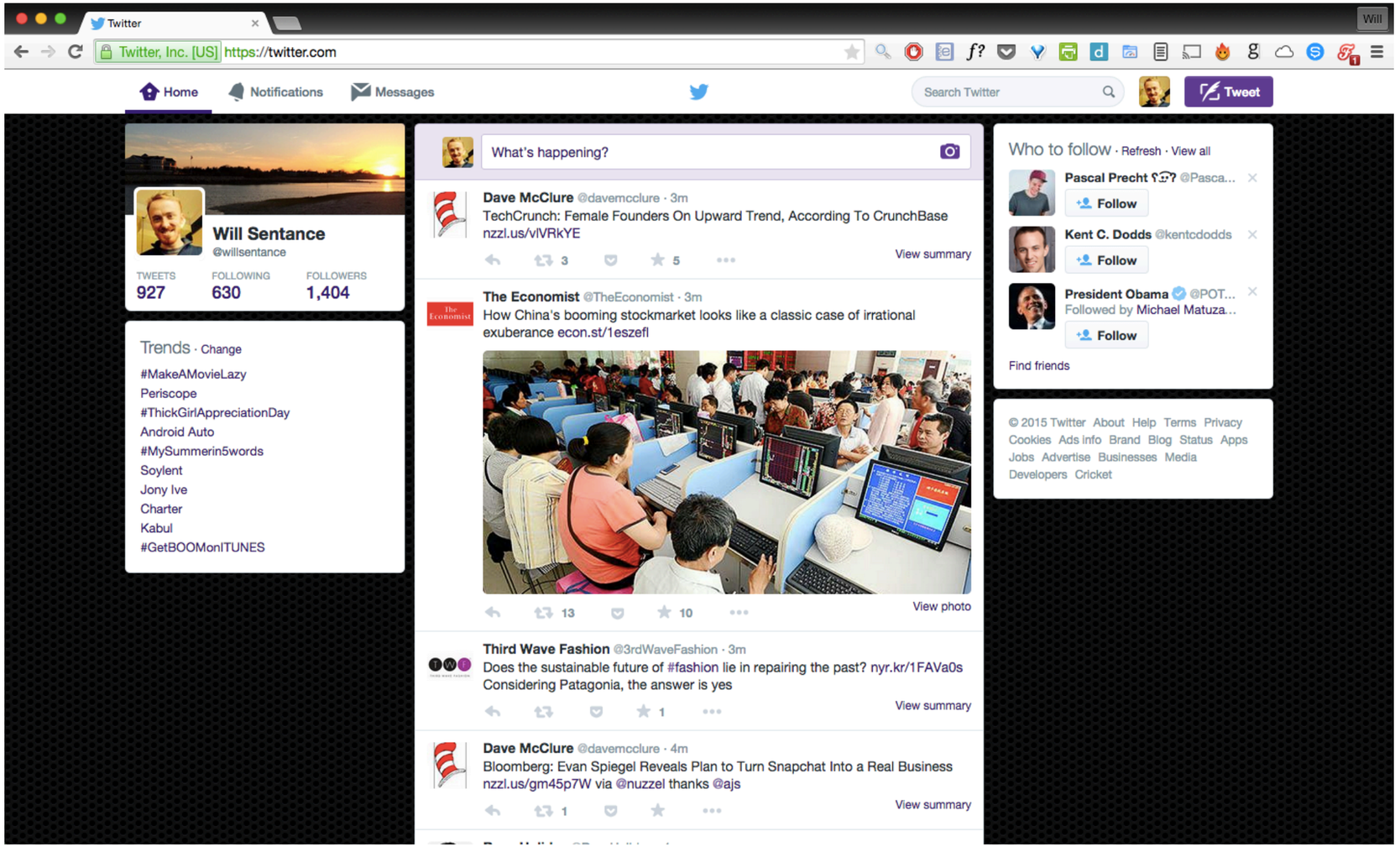
Servers

with Node

What happens when we open Twitter.com?



1. A request is formed by Chrome

— Request - I need to GET twitter.com's homepage
— Header - 'meta' info - "I'm from California", "I'm logged in as willsentance" etc
— Body - Empty (would only have something if we'd filled out a form and were sending that over)

Chrome begins the process of converting it into something the network will understand

2. The request for that page is sent out from your computer to twitter.com

2. Domain name system converts twitter.com to 199.16.156.0

3. 199.16.156.0 points to a particular computer online

http://twitter.com -> 199.16.156.0
http://google.com -> 216.68.216.46

http://twitter.com -> 199.16.156.0
http://google.com -> 216.68.216.46

4. The request arrives at Twitter's computer 199.16.156.0

5. Twitter's computer interprets the request

Twitter's engineers wrote the code to interpret the request

6. Twitter's computer forms a response

Header

— 'meta' data - today's date etc

Body

— an html file (here the code of Twitter's page)
— CSS, JS file
— an image
— some other data

7. The response is converted into a network message

8. The response is sent back to our computer

— Which also has a public address

— What's my ip: 45.59.229.41

Client and server

Our computer is the 'client'

Twitter's computer that 'serves' up a response to our request for a webpage is the server

HTTP (hyper-text transfer protocol)

- There are many different clients (browsers) so we have an agreed standard for how to communicate between any browser and any server
- HTTP is standard for webpages
- Others - FTP (for file sharing), SMTP (for sending emails)

REST

Four types of 'request'

GET (what we did to GET twitter's homepage)
PUT
POST
DELETE

Writing the code to interpret and form the response

Writing server code

What goes into a computer that's a server?

1. Communicate with the internet to receive requests
2. Understand network communications that use HTTP standard
3. Communicate with our computer's filesystem or a database
4. Can send out responses over the network
5. (Convert whatever language we want to code in into code that our computer understands)

We could write all this stuff ourselves, but fortunately a lot is done for us with a server side web framework

Node has all the code pre-written for us to do this

1. HTTP module
2. FTP module
3. Javascript V8 engine (like Google Chrome)

Your computer can be a server

1. Install the node 'app'
2. Create a file for our server code `myapp/server.js`
3. Run that code using our node app (in the command line)

```
node myapp/server.js
```

Let's write our first instructions that node will interpret

```
console.log("hello Codesmith")
```

Our first server

```
var http = require("http")
```

Making our server ready to receive requests from clients

```
http.createServer(function(request, response) {
  console.log(request)
}).listen(3000);
```

How do we try out our server?

Our computer (acting as a server) has a special address we can reach from our own machine

`localhost:3000`

Every time we open localhost:3000 we make a 'request' to our server

```
http.createServer(function(request, response) {
  console.log(request)
}).listen(3000);
```

We get access to it in our Javascript and can interpret it to form the right response to send back

You're going to go interpret the request in the Skill Builder!