

0 to API in node.js

key takeaways

- API first
- design APIs using tests.
- look into node if you haven't



last update: Sunday, 05-Jan-97 12:53:51

i am jm, owner & maintainer of [beats.com](#), & webmaster of [Goldendome.com](#).
[my man flynn's got his new EP out](#), and i'm diggin' it so far. (disclaimer: i'm not a record company, & the website's here. make sure you [contact eartube](#), and not me)

pager: 1-888-653-0480 (toll free)
(note that i'll only answer 1d calls if i recognize the phone #)
[e-page me](#) (up to 120 chars)
email: jm@goldendome.com
 (jm@beats.com for personal stuff)
[PGP Public Key Block](#)
[Geek Code](#)
my domain, [ill.beats.com](#).





••••• AT&T ⌂ 10:52 AM ⌂ 100% ⌂

fortune.com ⌂

FORTUNE SUBSCRIBE

All the magazines you love. Try it for FREE. > next issue

RETAIL LUXURY

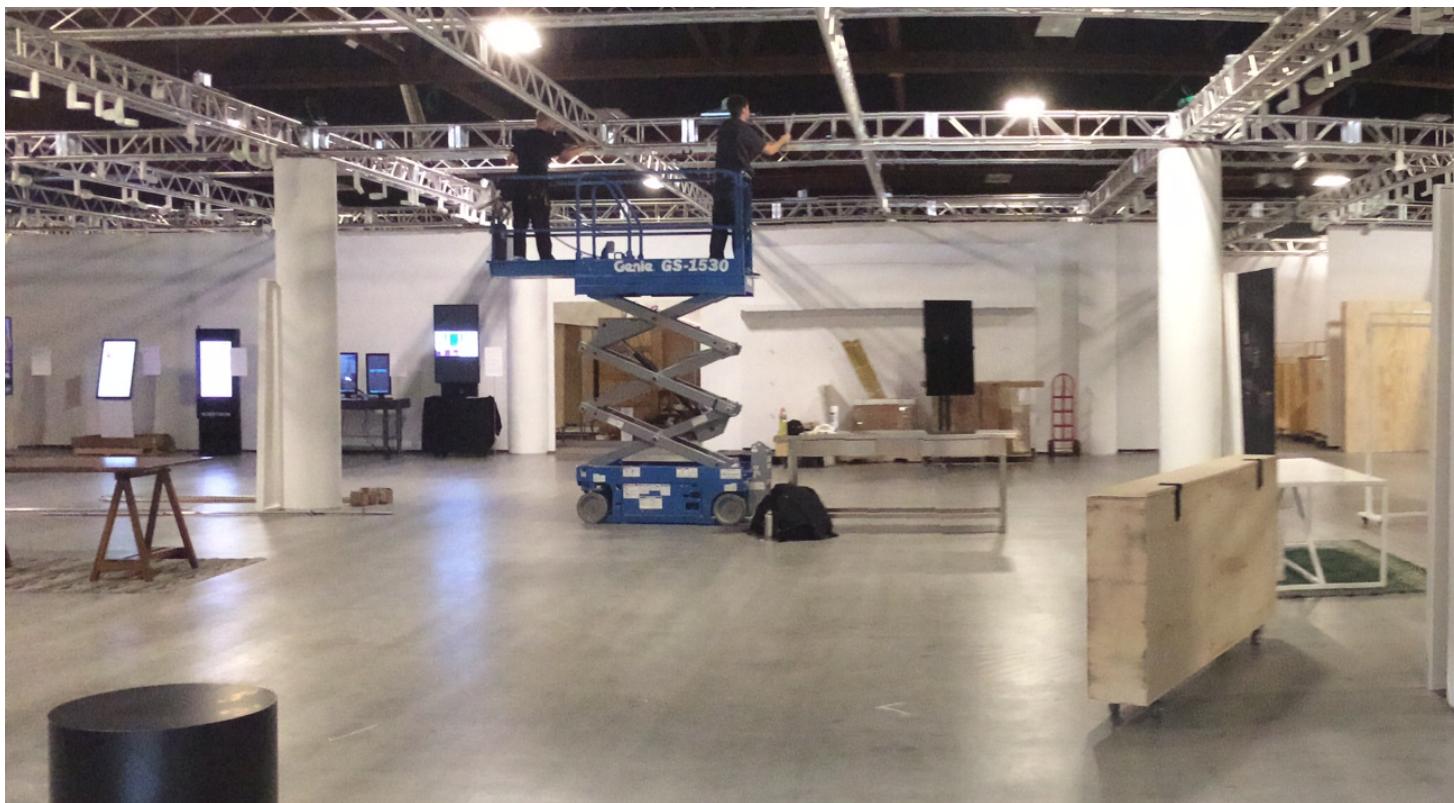
Nordstrom's latest tool in e-commerce battle? Shopping by text message

by Phil Wahba @philwahba

MAY 22, 2015, 10:35 AM EDT

✉ ⌂ f g+ in

< > ⌂ ⌂ ⌂



●●●○ AT&T 10:52 AM 100% ⚡

fortune.com

FORTUNE SUBSCRIBE

All the magazines you love. Try it for FREE. > nextissue

RETAIL LUXURY

Nordstrom's latest tool in e-commerce battle? Shopping by text message

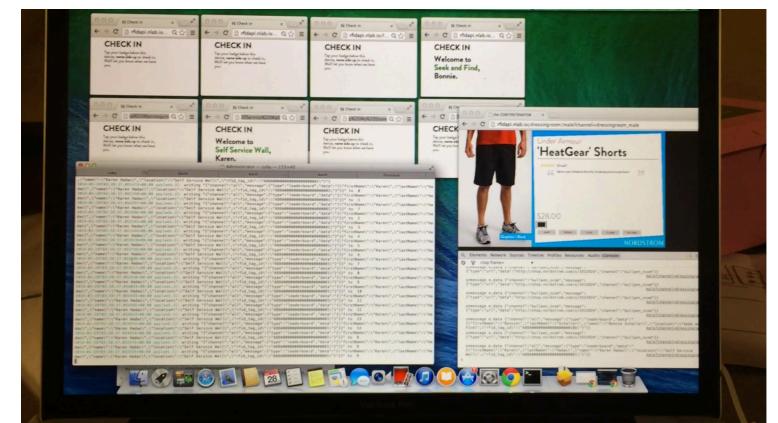
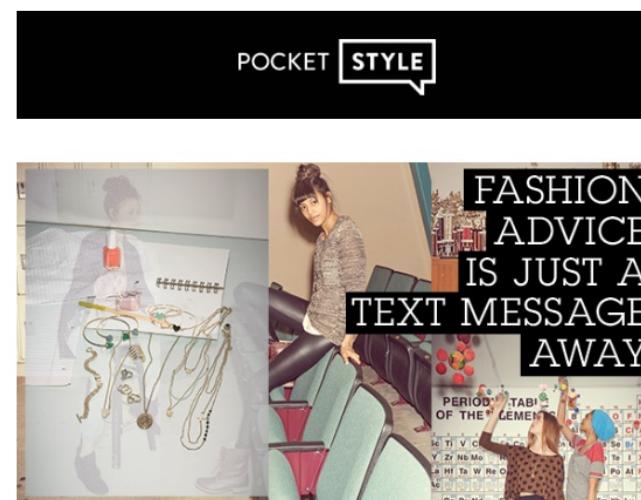
by Phil Wahba @philwahba

MAY 22, 2015, 10:35 AM EDT

[✉](#) [✉](#) [f](#) [g+](#) [in](#)

NORDSTROM

< > [↑](#) [↑](#) [↑](#) [↑](#)



●●●○ AT&T LTE 10:49 AM 100% ⚡

fortune.com

FORTUNE SUBSCRIBE

RETAIL E-COMMERCE

Nordstrom taps eBay's tech to build fitting room of the future

by Phil Wahba @philwahba

NOVEMBER 25, 2014, 7:00 AM EDT

[✉](#) [✉](#) [f](#) [g+](#) [in](#)

< > [↑](#) [↑](#) [↑](#) [↑](#)





ASP Active
Server Pages





ASP Active
Server Pages





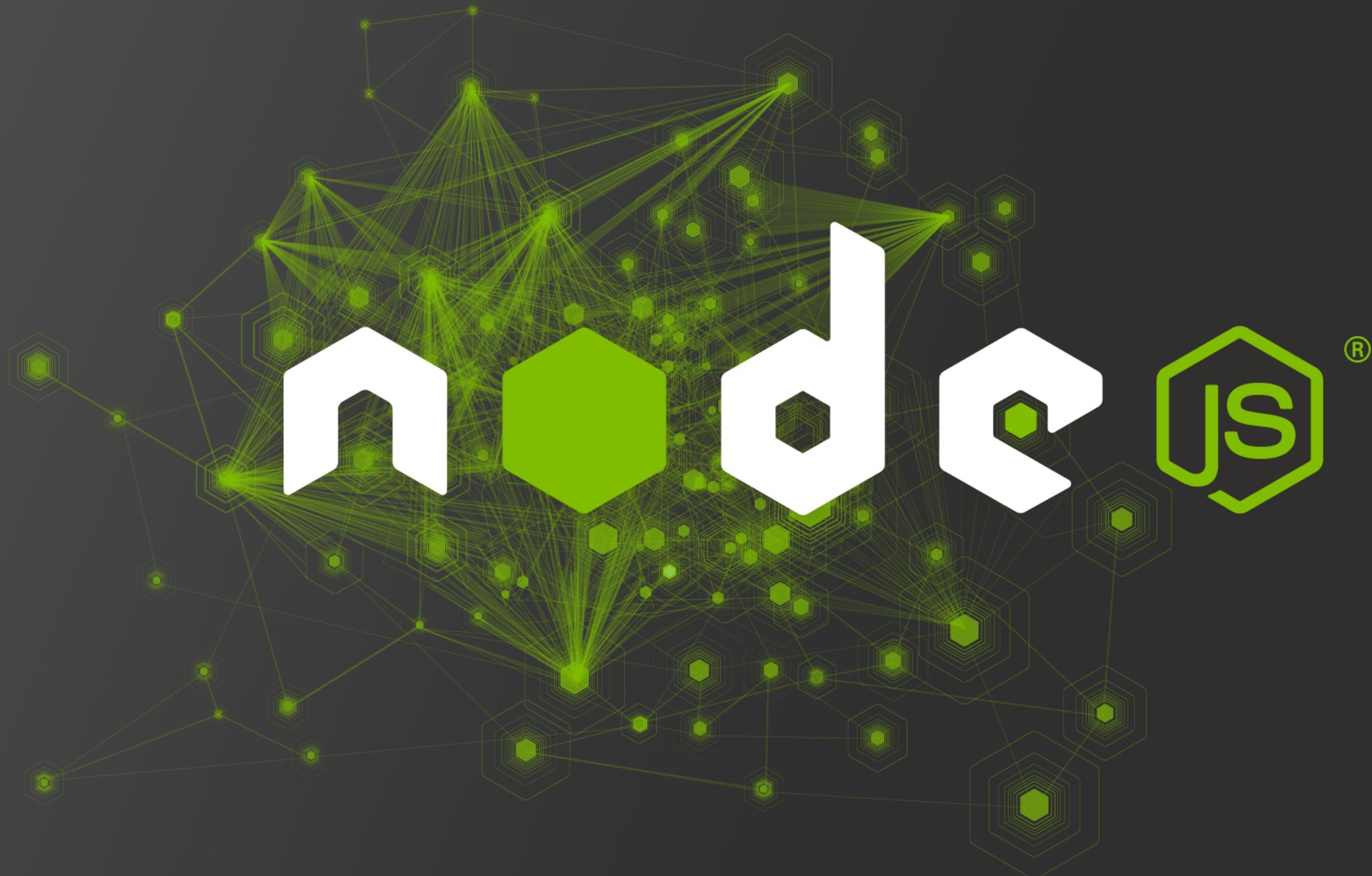
ASP Active
Server Pages





ASP Active
Server Pages



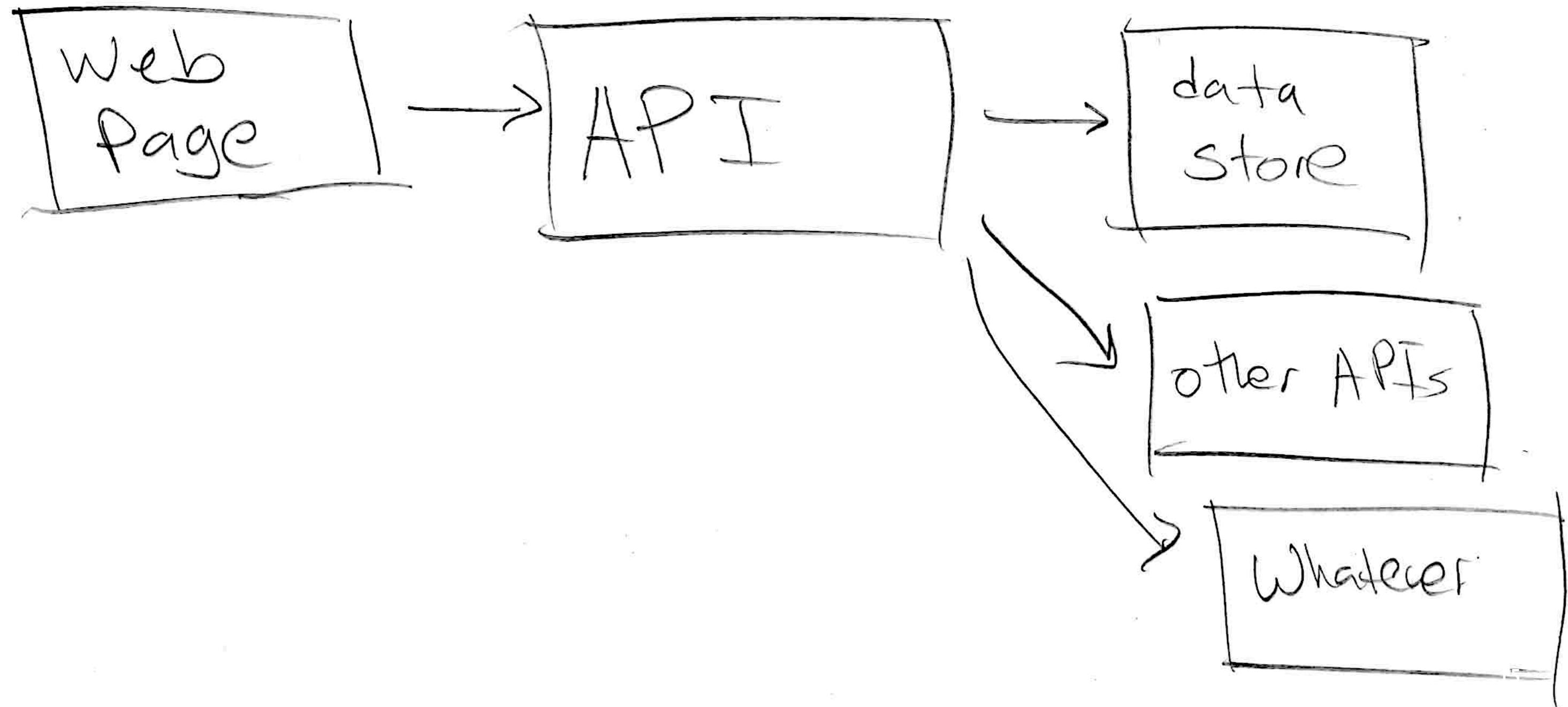


Why API, first?

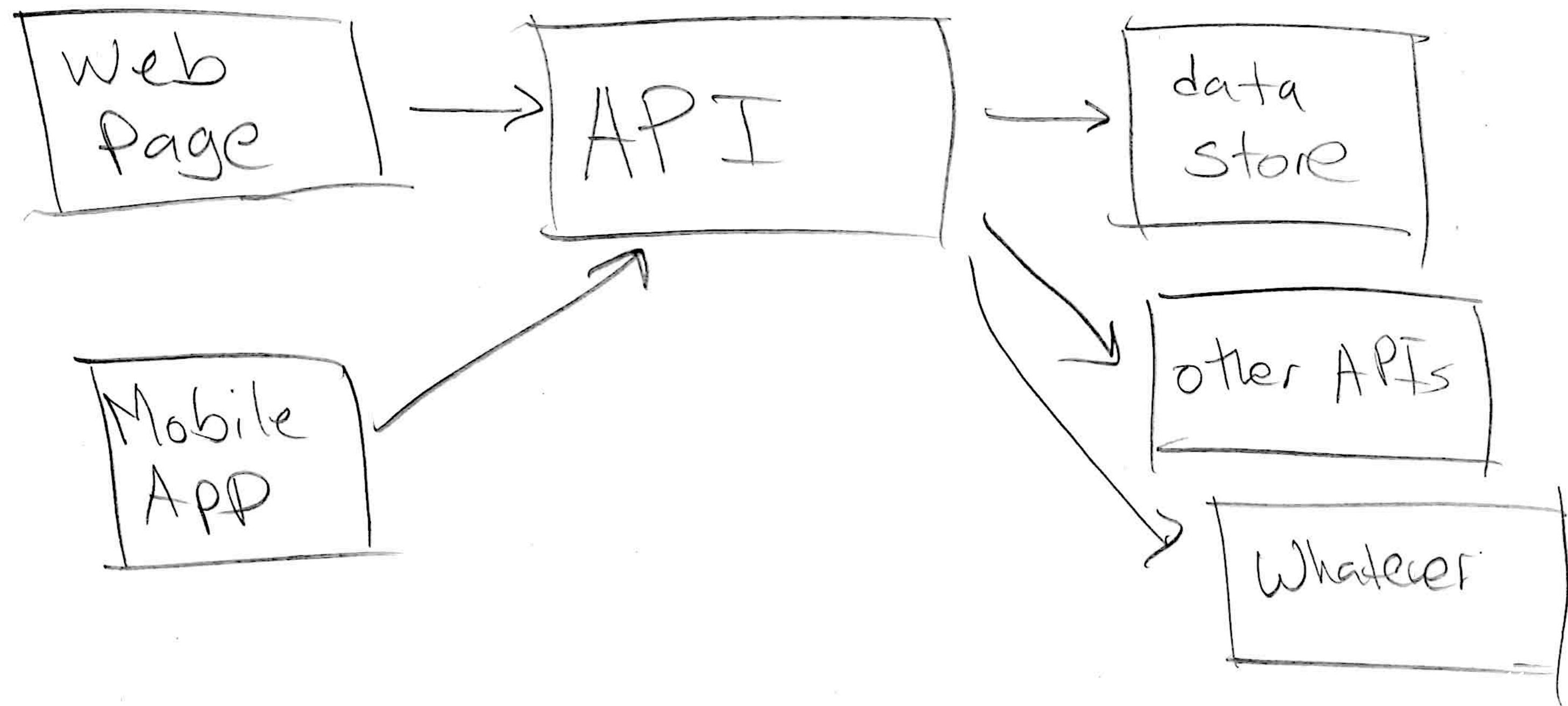
SPA - Single Page App



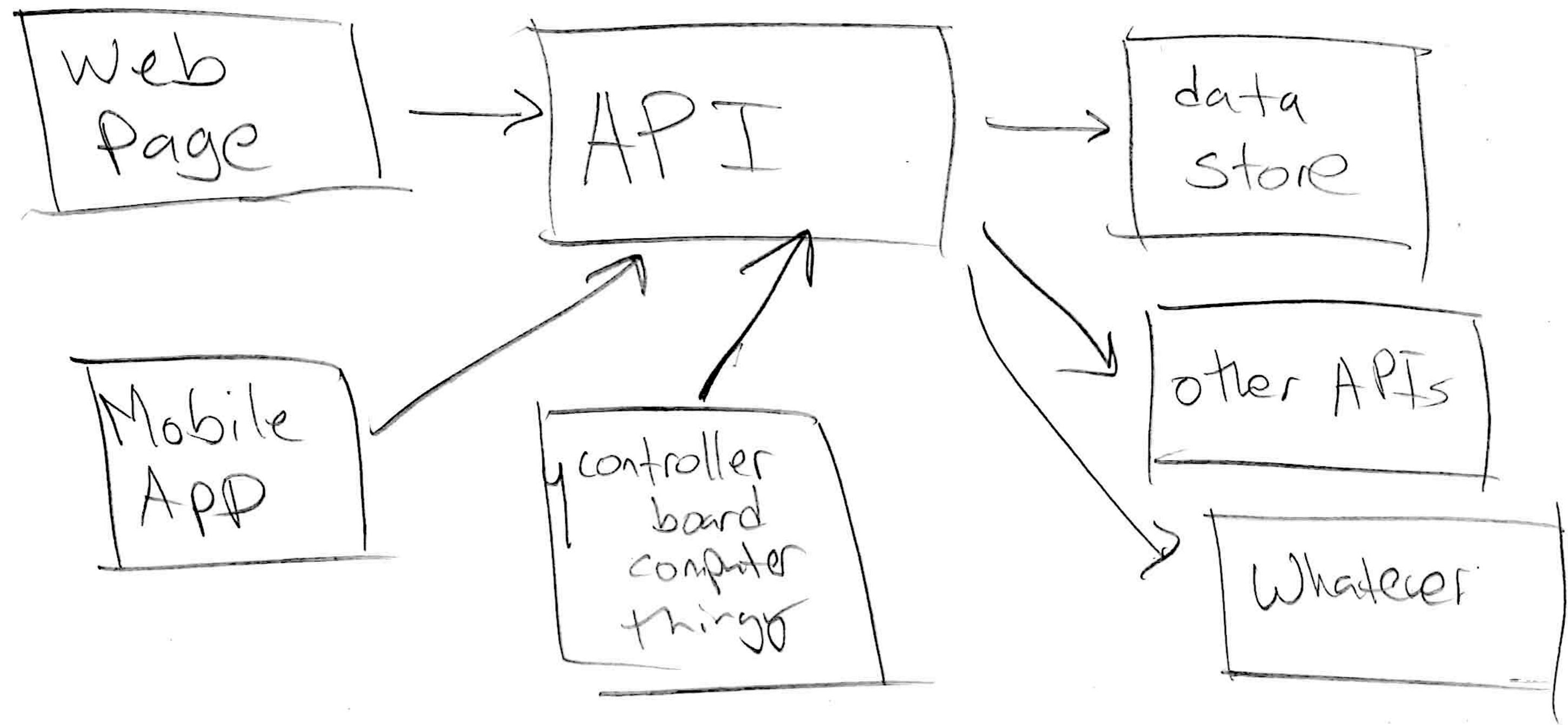
SPA - Single Page App



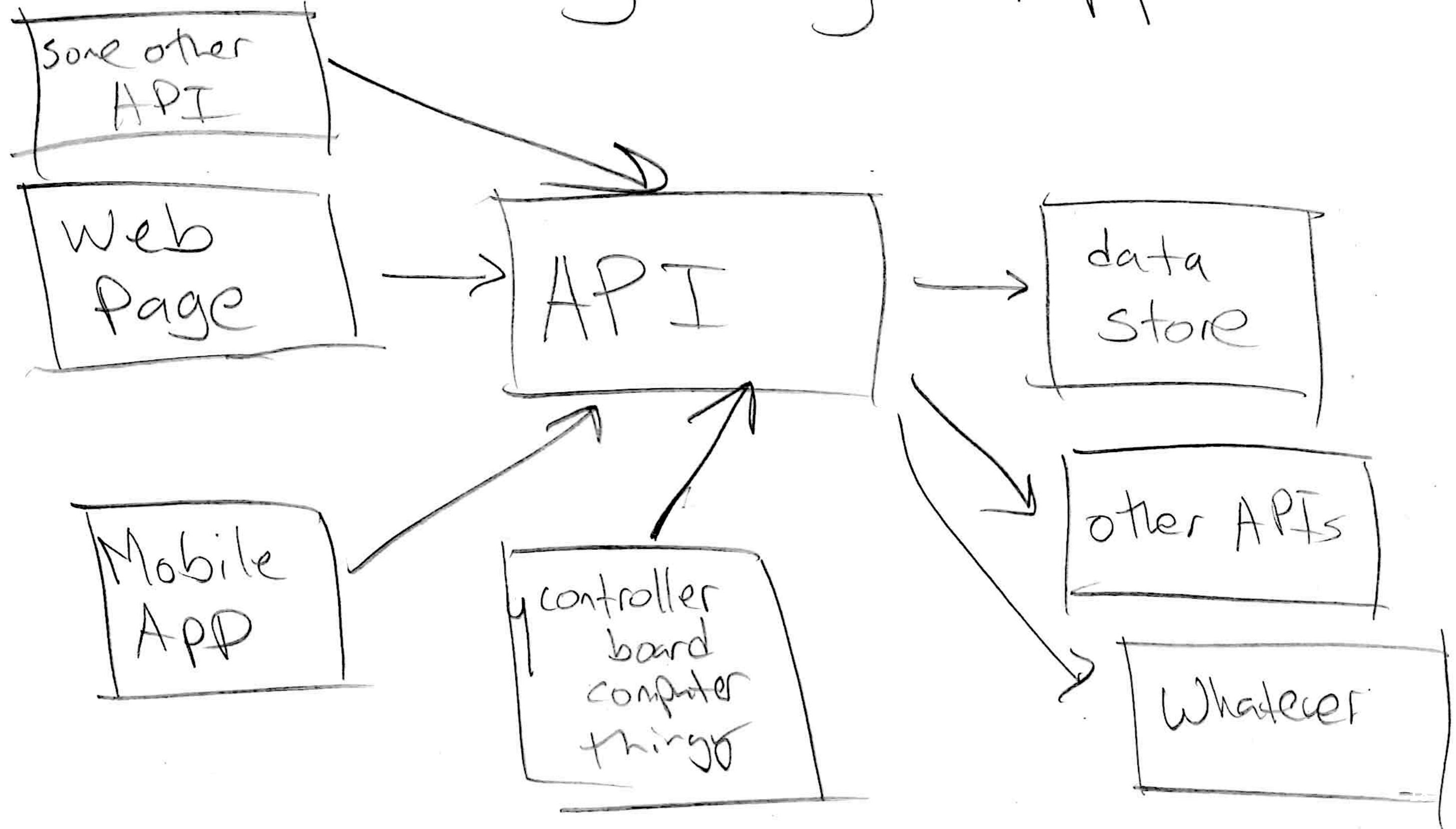
SPA - Single Page App



SPA - Single Page App



~~SPA~~ - Single Page App



Why node.js?

- minimal
- powerful
- additive, not subtractive (most of the time)
- it's javascript, with all its quirkiness

Let's start with an idea

You have an idea, or job to be done

- data you want available to a mobile or web site or app?
- gather data from multiple clients (signing up customers? weather sensor?)
- mashing up multiple APIs

Today's idea - flowers!

Click to go back, hold to see history Agriculture

Natural Resources Conservation Service

PLANTS Database

Home | About PLANTS | Team | Partners | What's New | NPDT | Help | Contact Us

You are here: Home / Advanced Search and Download

Advanced Search and Download

About the Advanced Search and Download

Part A: PLANTS Core Data

1. Distribution

PLANTS Floristic Area or Not

include:

Any
PLANTS Floristic Area
--North America
--Lower 48 U.S. States
--Alaska
--Canada

Display

Note: PLANTS Floristic Area or Not cannot be used with the next two search boxes.

State and Province

include:

Any
U.S. States
--Alabama
--Alaska
--Arizona
--Arkansas

Display

Note: County results are added to State and Province results. See About the Advanced Search and Download for details.

County Distribution
(Select a maximum of 256)

include:

Any
Alabama:Autauga
Alabama:Baldwin

Display

Search

Name Search

Scientific Name

State Search

Advanced Search

Search Help

PLANTS Topics

Alternative Crops

Characteristics

Classification

Cover Crops

Culturally Significant

Distribution Update

Documentation

Fact Sheets & Plant Guides

Introduced, Invasive, and Noxious Plants

Threatened & Endangered

Wetland Indicator Status

Image Gallery

50,000+ Plant Images

Download

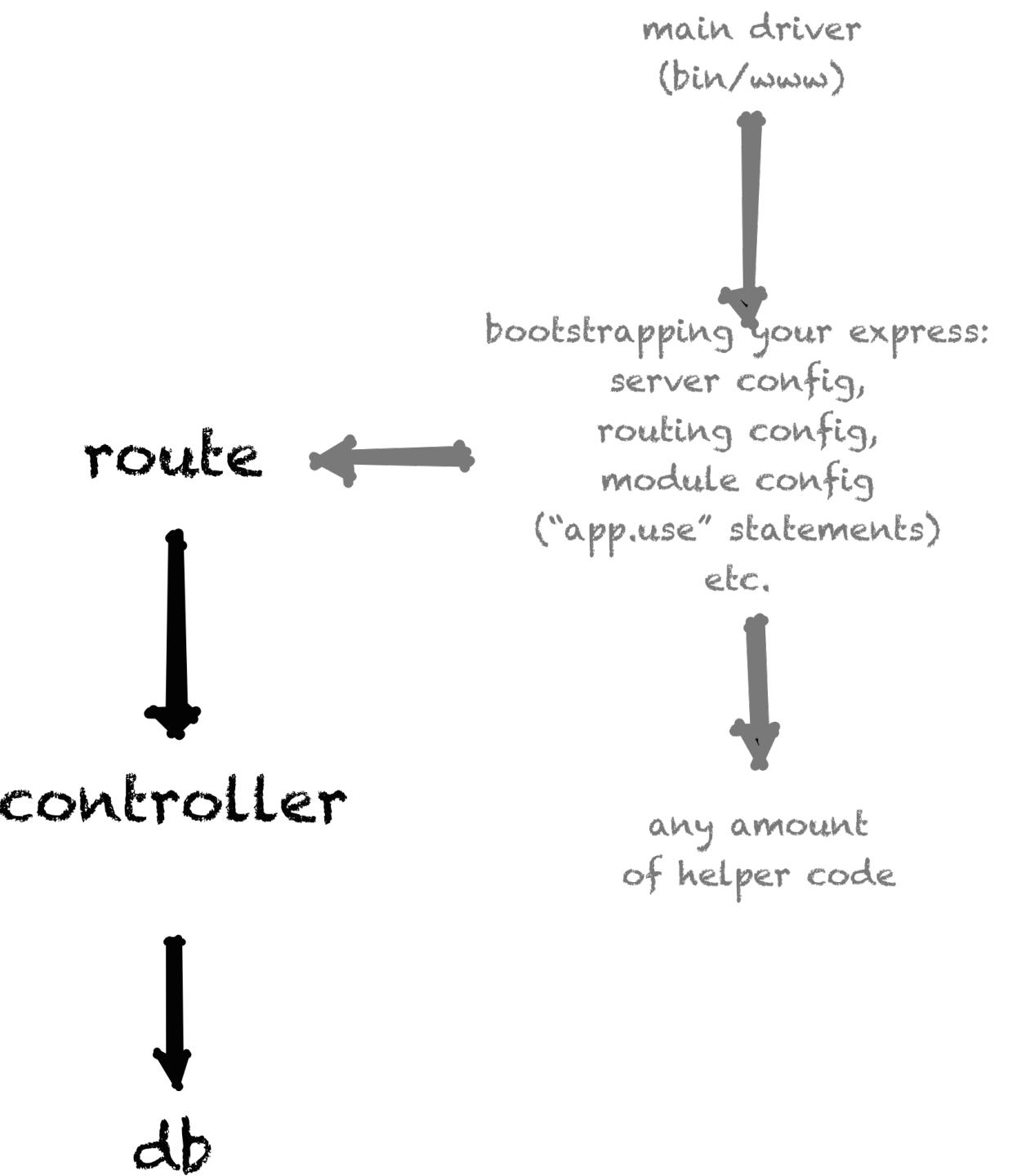
express

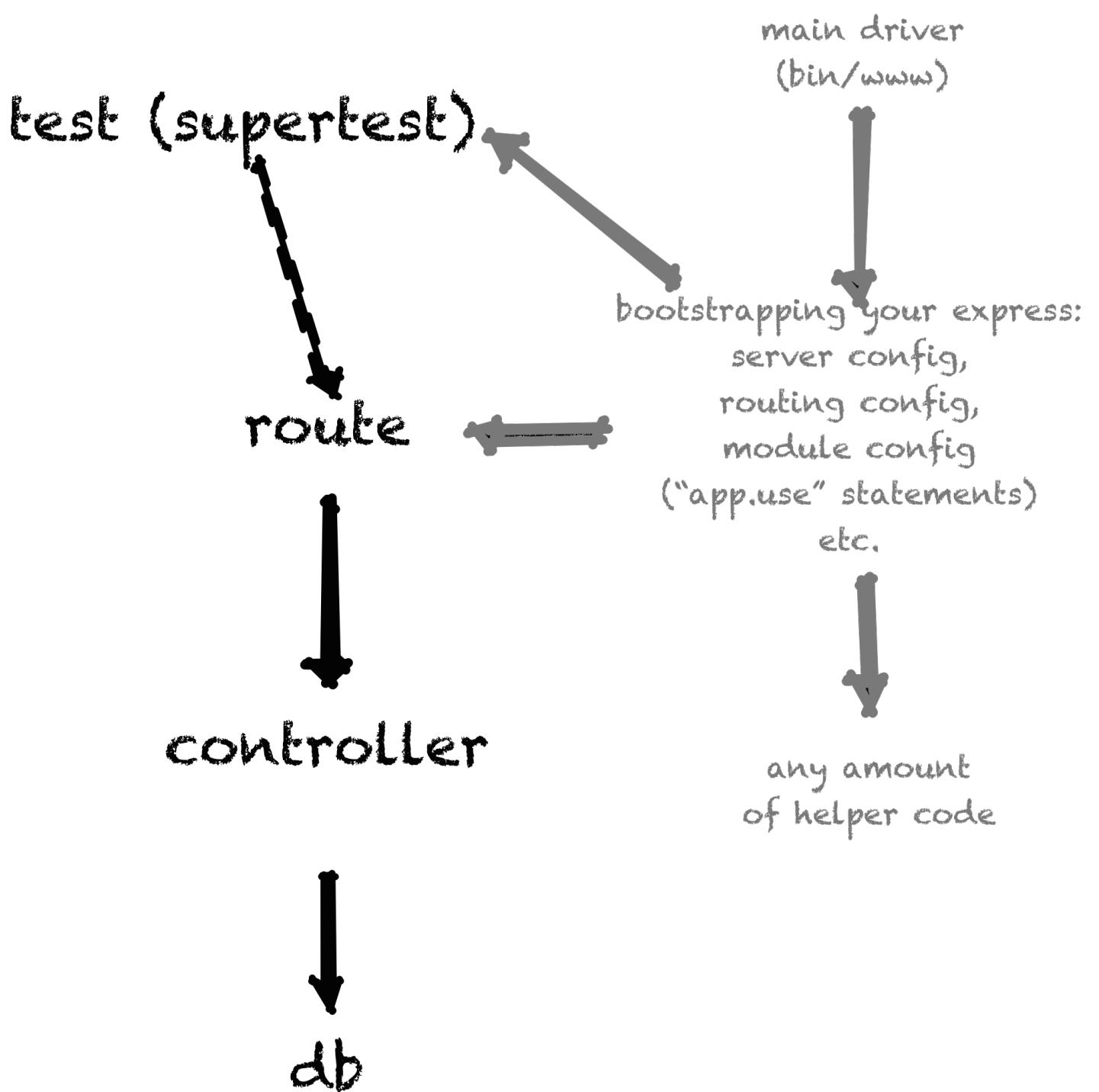
mocha

simple, flexible, fun



PostgreSQL





Create the project

```
% sudo npm install -g nodemon          #automatically restarts our server  
% sudo npm install -g express-generator #quickly gets us started. could use "npm init" but we're pressed for time  
% express ft --git                      #create 'scaffolding' for our project with git ignore file  
% cd ft                                #get in there...  
[get rid of 💩]
```

About REST

- Google "Roy Fielding Chapter 5"
- an *architectural style*
- Resources are nouns, Operations are verbs
- discoverable endpoints
- be consistent

A Resource in JSON

```
{  
  id: 7275,  
  name: "'oha kepau",  
  description: "this ",  
  create_dt: null,  
  user_id: null,  
  zones: "10a,11",  
  created_at: "2015-09-07T13:49:38.988Z",  
  updated_at: "2015-09-07T13:49:38.988Z",  
  color: "",  
  color_family: "purple",  
  states: "USA (HI)",  
  growth_season: "fall",  
  shopping_list_id: null,  
  garden_id: null,  
  duration: "Perennial",  
  scientific_name: "Clermontia hawaiiensis",  
  foliage_color: "green",  
  seed_color: null,  
  fruit_color: "violet",  
  shade_tolerance: "full sun"  
}
```

```
{  
  plants: [  
    {  
      id: 3849,  
      name: "'ahakea",  
      description: null,  
      create_dt: null,  
      user_id: null,  
      zones: null,  
      created_at: "2015-09-07T13:49:29.288Z",  
      updated_at: "2015-09-07T13:49:29.288Z",  
      color: "",  
      color_family: null,  
      states: "USA (HI)",  
      growth_season: "",  
      shopping_list_id: null,  
      garden_id: null,  
      duration: "Perennial",  
      scientific_name: "Bobea timonioides",  
      foliage_color: "",  
      seed_color: null,  
      fruit_color: "",  
      shade_tolerance: null  
    },  
    {  
      id: 3847,  
      name: "'ahakea lau nui",  
      description: null,  
      create_dt: null,  
      user_id: null,  
      [...]
```

Endpoints

GET /plants

#verb: GET noun: plants -- get a list of plants

GET /plants/:id

#verb: GET noun/resource: a specific plant with id

GET /plants?name=<term>

#verb: GET noun: a plant whose "name" matches the <term>

POST /plants

#verb/op: POST noun/resource(bucket): plants -- add a new plant

PATCH /plants/:id

#verb/op: PATCH noun/resource(bucket): plants -- update a specific plant

[. . .]



All your dreams

Come true

Hello world of tests, i guess

```
describe('basics', function(){
  it('should return 404 on /', function(done){
    request(app)
      .get('/')
      .set('Accept', 'application/json')
      .expect(404, done);
  })
});
```

Time for a real test - one assertion style

```
it('should return a list of plants',function(done) {
  request(app)
    .get('/v1/plants')                                // create a new HTTP request...
    .set('Accept', 'application/json')                // of type GET /v1/plants
    .expect('Content-Type',/json/)                   // set Accept header to ask for json response
    .expect(200, function(err,res) {                  // content type header should be json
      demand(err).be.empty();                         // need status 200 OK, and the following...
      demand(res.body).not.be.empty();
      demand(res.body.plants.length).equal(100);

      //get an example record
      var firstPlant = res.body.plants[0];
      demand(firstPlant).not.be.empty();
      demand(firstPlant.name).not.be.empty();
      demand(firstPlant.scientific_name).not.be.empty();
      done();
    })
});
```

Expect a specific result, use our JSON design

```
it('should return a particular plant',function(done){
  request(app)                                // create a new HTTP request...
  .get('/v1/plants/17590')                    // endpoint for single plant, very RESTful, yes
  .set('Accept', 'application/json')           // i want JSON back
  .expect('Content-Type','json/')             // so i should get JSON back
  .expect(200, {                                // check it: i can check for an entire object!
    name: 'Agardh lupine',
    id: 17590,
    description: null,
    create_dt: null,
    user_id: null,
    zones: null,
    created_at: '2015-09-07T13:50:06.654Z',
    updated_at: '2015-09-07T13:50:06.654Z',
    color: '',
    color_family: null,
    states: 'USA (CA)',
    growth_season: '',
    shopping_list_id: null,
    garden_id: null,
    duration: 'Annual',
    scientific_name: 'Lupinus agardhianus',
    foliage_color: '',
    seed_color: null,
    fruit_color: '',
    shade_tolerance: null },done);
});
```

Support querying

```
it('should return a particular plant via query',function(done){
  request(app)
    .get('/v1/plants/?name=Agardh%20lupine')
    .set('Accept', 'application/json')
    .expect('Content-Type',/json/)
    .expect(200, {    id: 17590,
                      name: 'Agardh lupine',
                      description: null,
                      create_dt: null,
                      user_id: null,
                      zones: null,
                      created_at: '2015-09-07T13:50:06.654Z',
                      updated_at: '2015-09-07T13:50:06.654Z',
                      color: '',
                      color_family: null,
                      states: 'USA (CA)',
                      growth_season: '',
                      shopping_list_id: null,
                      garden_id: null,
                      duration: 'Annual',
                      scientific_name: 'Lupinus agardhianus',
                      foliage_color: '',
                      seed_color: null,
                      fruit_color: '',
                      shade_tolerance: null },done);
});
```

deploy to the Internet!

```
% heroku login  
% heroku create ftapi-ex  
% heroku addons:create heroku-postgresql  
% vi .gitignore  
% git add .  
% git commit -am "net push"  
% git push heroku master
```

Internet



example client: iOS

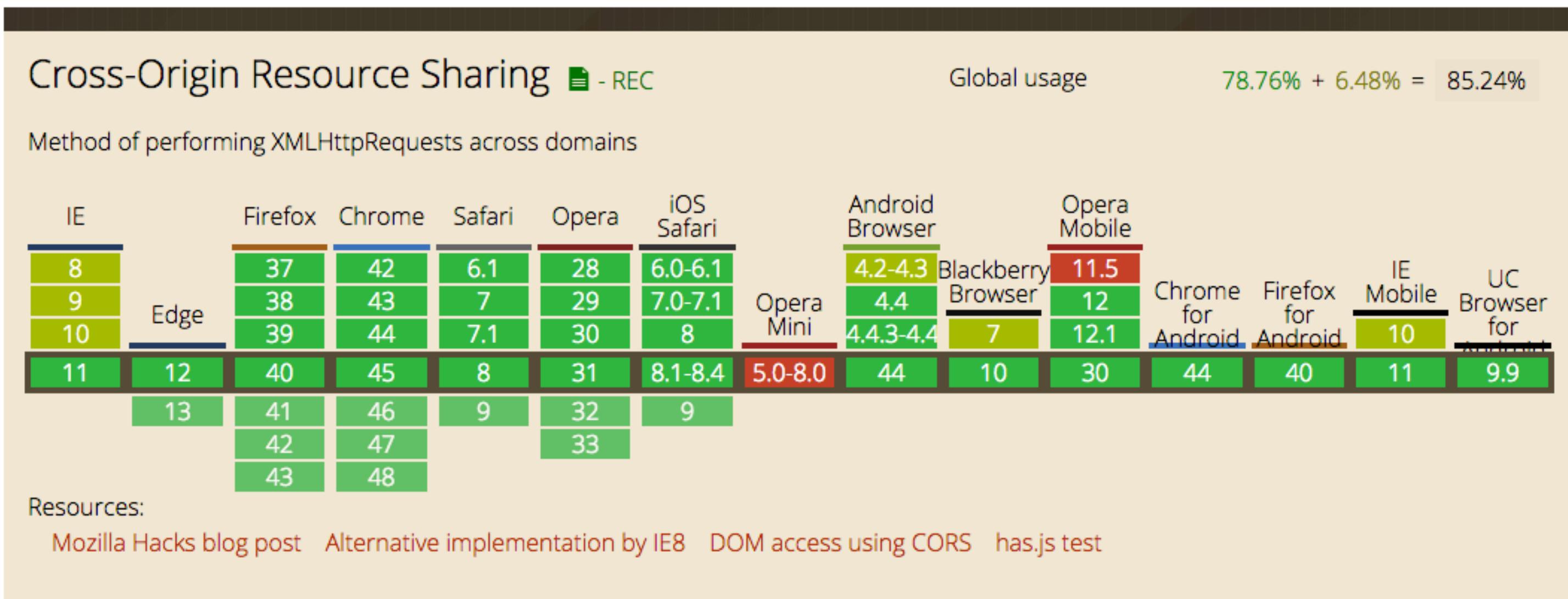
Security considerations

- CORS + jsonp



Security considerations

Browser support



Security considerations

- CORS + jsonp
- Secure your endpoints using PassportJS
 - passport-local + passport-http-bearer
 - jwt-simple
 - Sign your Tokens
- use HTTPS on the Internet, always.



Summary

- API first
- design APIs using tests.
- node is neat.



Web API Design: Crafting Interfaces that Developers Love (PDF)

The World's Most Misunderstood Programming Language

Javascript: Understanding the weird parts

ExpressJS crash course from StrongLoop

Token Authentication and Single Page Apps

Related Code Camp sessions

11:00 AM-12:00 PM TDD with TypeScript, AngularJS and Node.js

2:15 PM-3:15 PM Building Modern Web Front Ends on AWS, Linux and ReactJS

3:30 PM-4:30 PM Testing Node APIs With Mocha and SuperTest

Questions?

@jonmadison, <http://www.jonmadison.com>

me@jonmadison.com

jon madison on most of the things

