Isomorphic Web Application

Demo using React

Isomorphic Web Application

Oliver N.

Software Engineer

Agenda

1. A brief history of web development
   - Server-rendered Multi-Page Application (MPA)
   - Client-rendered Single Page Application (SPA)
2. What is Isomorphic JavaScript?
3. Why on the Earth do we need it?
4. How can I build isomorphic app?
5. Stop talking. Show me the code!

TL;DR

1. Isomorphic JavaScript is the pattern of running JavaScript code on both server & client.
2. People are using it for production today.
   Ask Facebook, Yahoo, Asana, Airbnb, Rising Stack, …
3. This is not another talk about NodeJS!

once upon a time…

```
<?php
        echo “Hey, a web server is talking to you !!”;
        echo “How many {$item.name} do you want to buy?”
?>
<form>
        <input name=“your-name” />
        <input name=“quantity” />
</form>
```

then…

AJAX

Asynchronous Javascript And XML

then…

HTML

then…

then…

node JS™

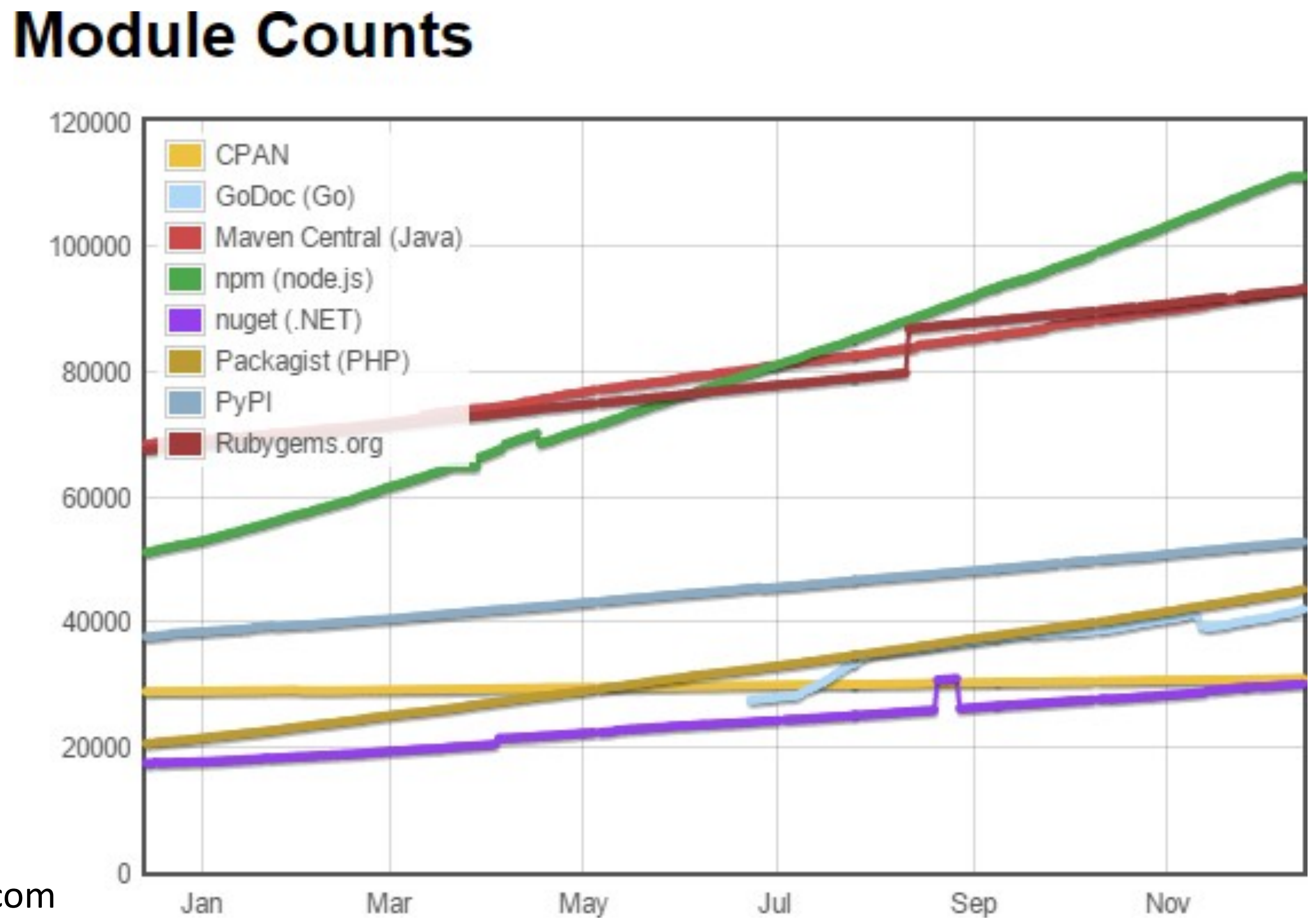
then…

AJAX
Asynchronous Javascript And XML

HTML 5

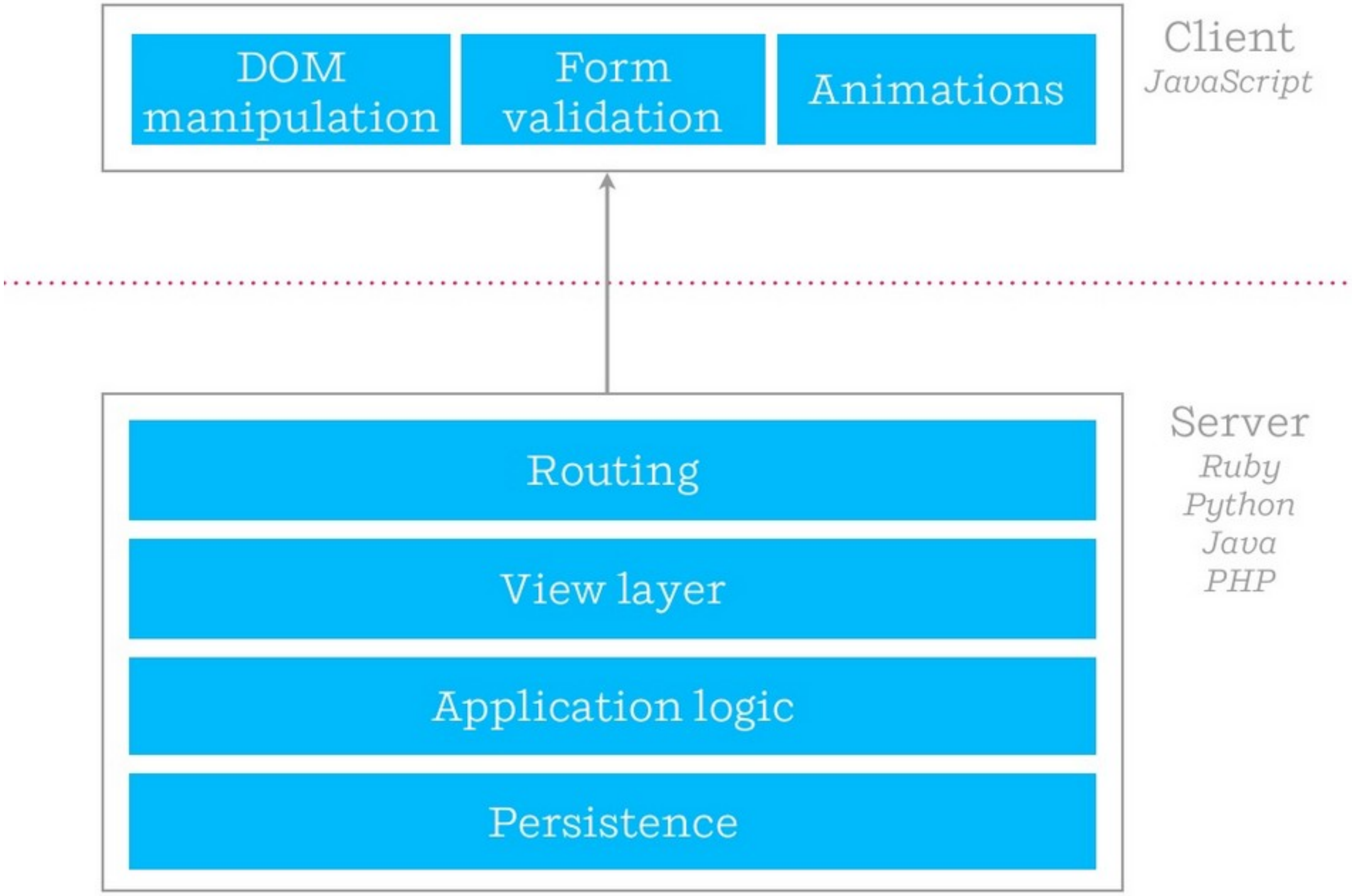
node JS ™

Today (Dec, 2014)



http://modulecounts.com

Traditional Multi-Page Application



https://www.slideshare.net/spikebrehm/jsconf-us-2014-building-isomorphic-apps

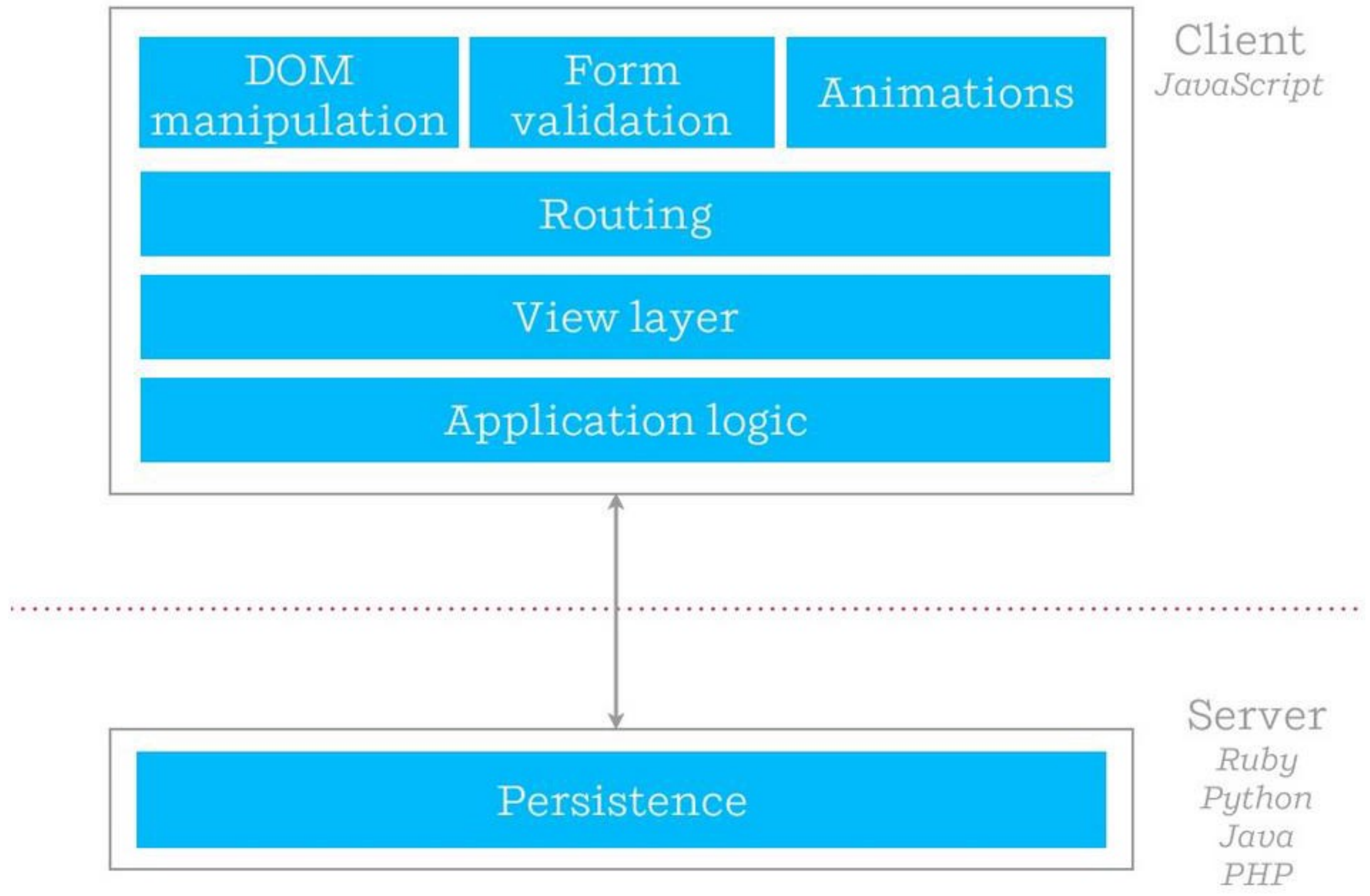
Traditional Multi-Page Application

- Server-rendered content,
  Improving user experience by using AJAX, JQuery, …
- Serving HTML on first-load is fast.
- Crawlers, screen-readers are happy with HTML.

But:

- Maintain UI render and logic in both client and server
- Duplicate application logic in (usually) two languages, two frameworks, two development stacks, two templates, …

Single Page Application



https://www.slideshare.net/spikebrehm/jsconf-us-2014-building-isomorphic-apps

Single Page Application

- Client-rendered content
- Separate application logic and data retrieval
- More interactivity, optimistic UI, offline, mobile

But:

- Not SEO-friendly.
  - Still have to pre-render pages on server for crawlers.
- Users have to wait a few seconds of blank page or loading spinner before seeing the content.

Multi-Page Application

OR

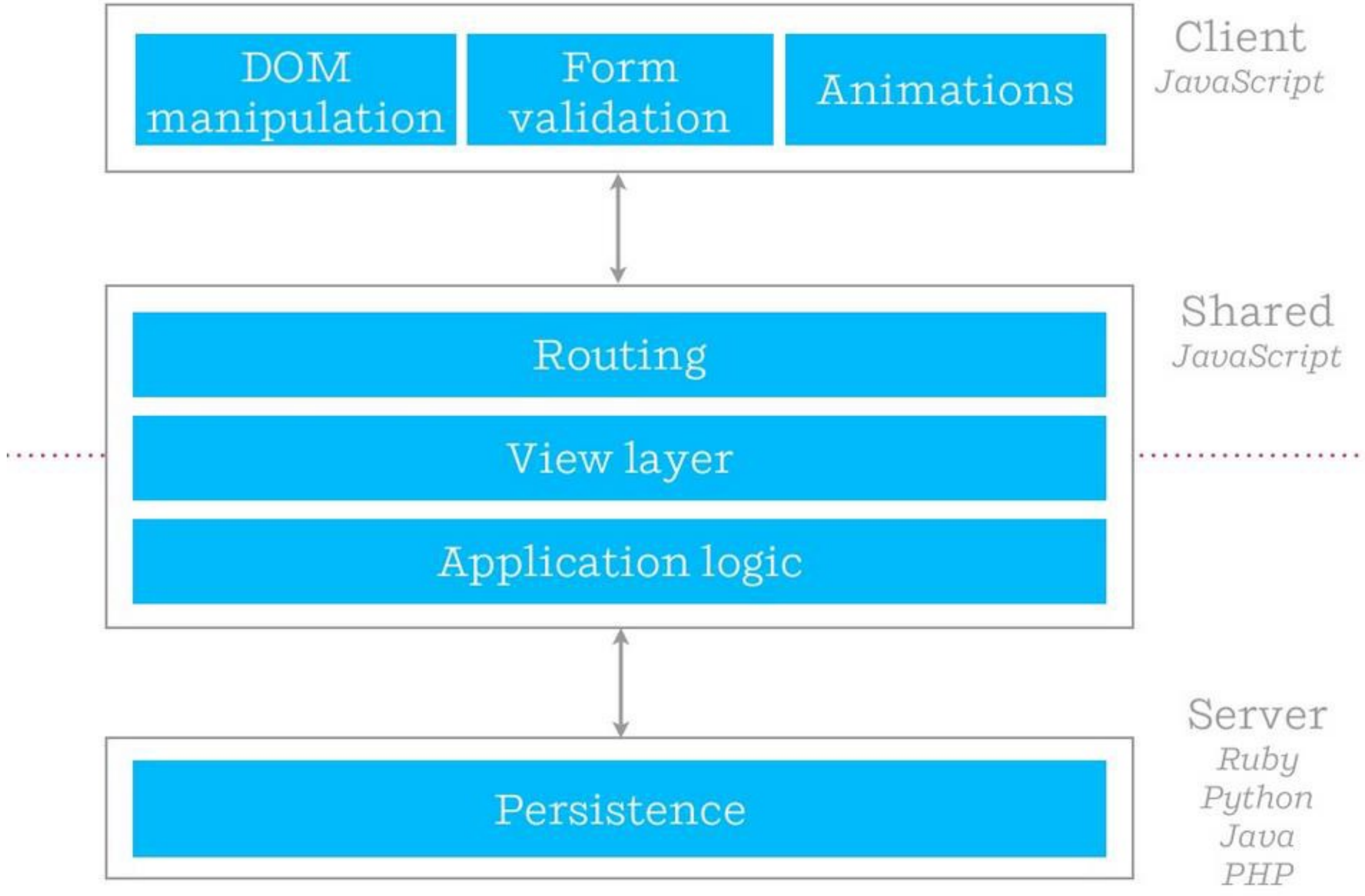
Single-Page Application

We have another option:

We have another option:

Isomorphic Web Application

The best of both worlds



https://www.slideshare.net/spikebrehm/jsconf-us-2014-building-isomorphic-apps

Isomorphic Web Application

Mix server-rendered and client-rendered content

- On first page load, serve real server-rendered HTML.
- Client-side JS app bootstraps on top of server-rendered HTML rather than bootstrapping into an empty div.
- From that point on, it's a client-side JS app.

Taking the best of both worlds

- Performance
  - MPA - Serving fully-formed HTML is fast.
  - SPA - AJAX, transport data instead of HTML.
- Connection quality
  - SPA can handle client state, request only needed data.
- SEO & Accessibility
  - MPA – crawlers & screen readers needs HTML. Google will always prefer MPA.
- UX
  - SPA - more interactivity, optimistic UI, offline first, client first, mobile first
- Maintainability

So, do I have to rewrite my app in NodeJs?
It’s terrible!

So, do I have to rewrite my app in NodeJs? It’s terrible!

No, you don’t have to. Just use your preferred stack plus a few code for NodeJs.

Are there anyone building isomorphic app in production?

Are there anyone building isomorphic app in production?

Facebook	*Instagram*	*RisingStack*
Asana	*Meteor*	*Yahoo! Mail (next version*)*

https://www.slideshare.net/rmsguhan/react-meetup-mailonreact

View layer shared

Entire app runtime synced between client & server

https://www.slideshare.net/spikebrehm/jsconf-us-2014-building-isomorphic-apps

METEOR

METEOR

But don’t use it.

Let’s build our own
Isomorphic application

Let’s build our own isomorphic application

- Understand Isomorphic JavaScript
    - Environment independent
    - Shimmed per environment
- Building blocks

Understand
Isomorphic JavaScript

Environment-independent

- Do not depend on environment-specific features

Browser: window, DOM

NodeJs: process, “fs”

Shimmed per environment

- Provide shims for accessing environment-specific features so we can use the same API

```
Browser:  xhr.open('GET', 'http://example.com')
NodeJs:   http.request({ host: 'example.com', path: '/' })
Shim:     superagent.get('http://example.com')
```

Most of your favourite JS libraries are Isomorphic

You can use these libraries on server or client

jquery

backbone

react

moment

superagent

…

underscore

handlebar

mithril

numeral

i18next

Building blocks

Building blocks: what do we need?

View:	Render HTML with or without browser’s DOM.
Routing:	Navigate through the application.
Communicating:	Send AJAX or HTTP requests.
Stores:	Store application states.
States:	Transfer application states from server to client.
Bundling:	Combine all our node modules to a single file that can run on browser.

Building blocks: which libraries can we use today?

View: react, handlebar

Routing: direction, flux-router-component

Communicating: superagent, fluxible-plugin-fetchr

Stores: flux, backbone model

States: express-state or implement your own

Bundling: browserify, web-pack

React

- Virtual DOM

```
Browser:  React.render( <App />, document.body )
NodeJs:   React.renderToString( <App /> )
```

Browserify

- require() node modules in browser.
- Provide shims for node-specific features that can run on browser.
- Bundle all modules to single script file.

Demo

using React

Thanks for your listening !!

Oliver N.

Software Engineer