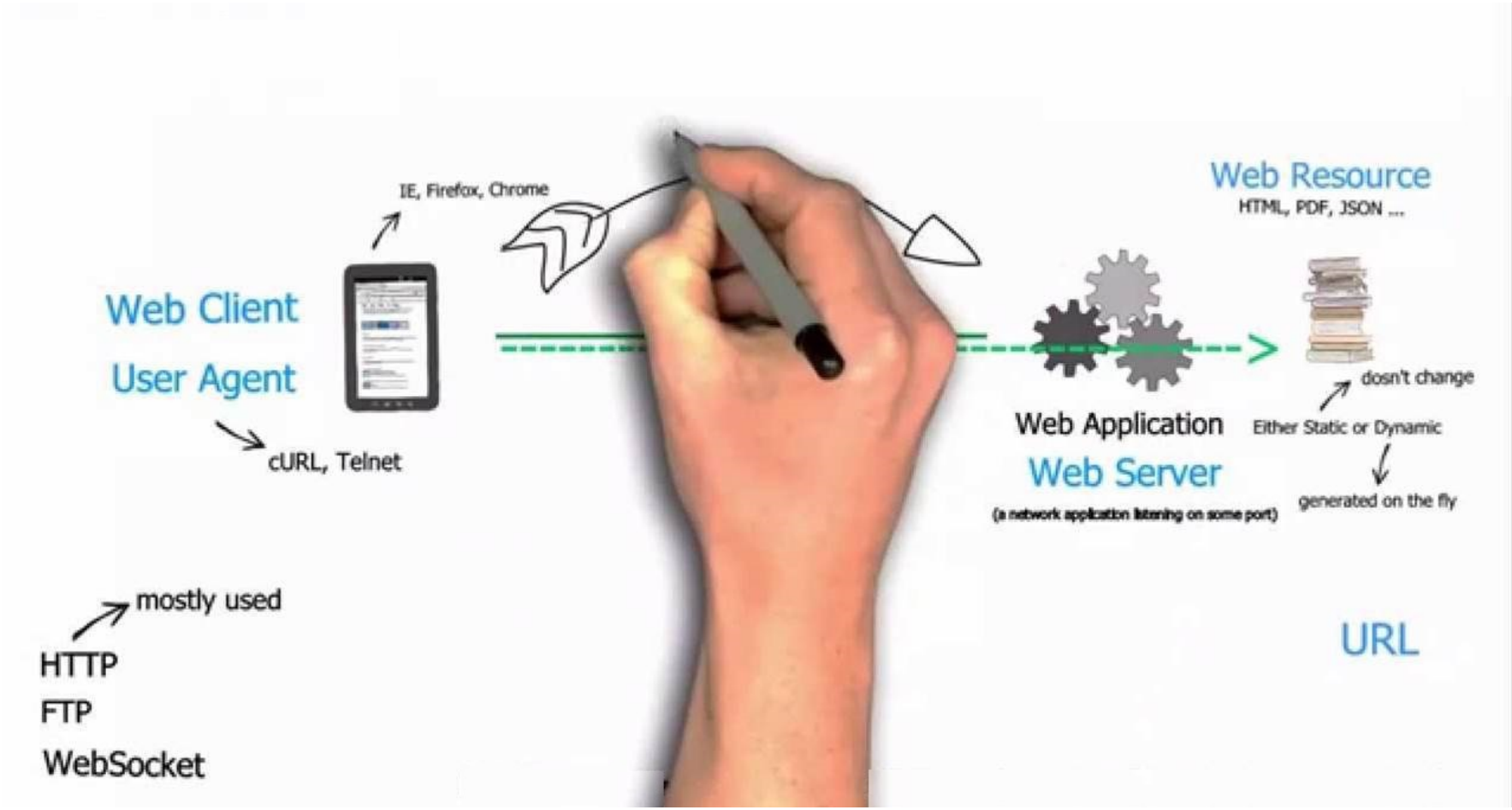
MEAN Stack

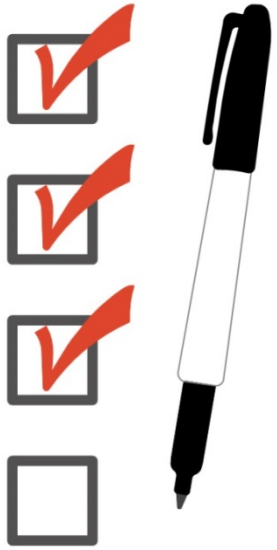
Structure of Web Application

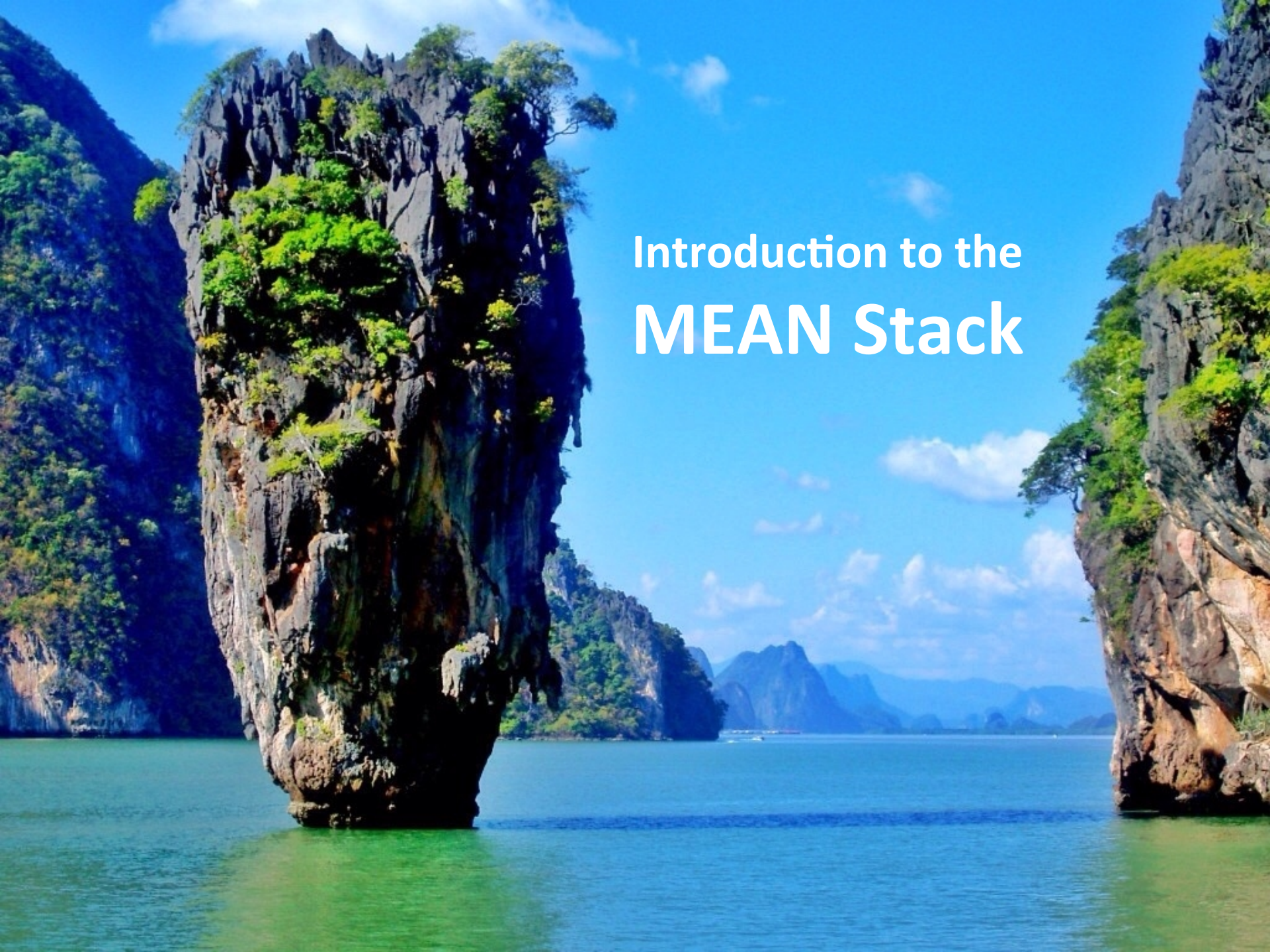


Web Application

Requirements for a modern web app

- ✓ Customers want fast web sites/fast response time
- ✓ No page reloads
- ✓ Enterprises want to go virtual
  - One box + Several Virtual Images = Shared Hardware
  - Systems with minimal memory footprint/overhead needed
- ✓ As many concurrent requests as possible
- ✓ Only load resources when needed (conditional loading)
- ✓ Most of the data must come from a slim REST-API
- ✓ Mobile friendly/Responsive UIs
- ✓ Automated build for backend AND frontend
  - Including check for coding conventions, testing,…
  - Integration in company's continuous integration platform




Introduction to the
MEAN Stack

What is MEAN stack?

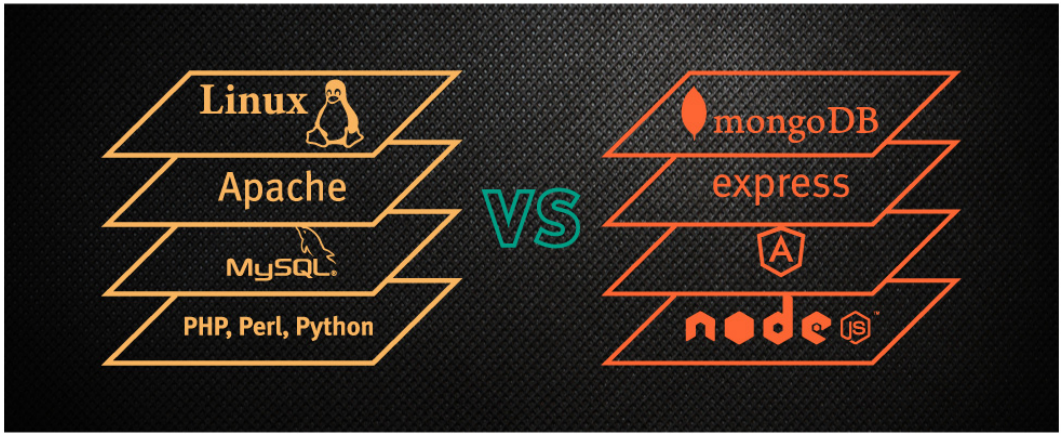
"MEAN is a fullstack JavaScript platform for modern web applications"
– Mean.io

Who uses MEAN stack?

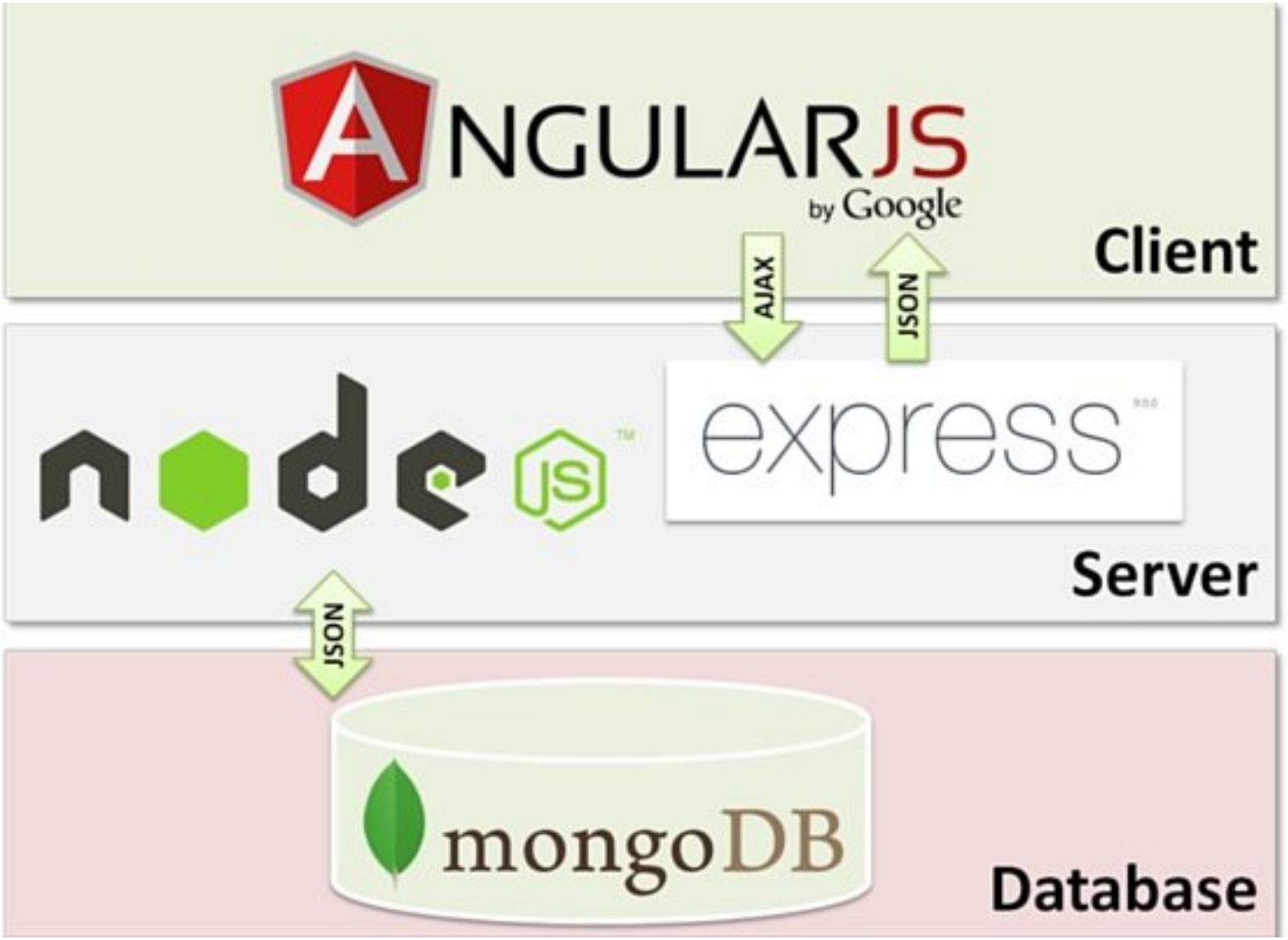
and many more…

How it is different from LAMP stack?

- ✓ Building web application with a LAMP approach
  - Server-side language: PHP, Perl, JSP, ASP…
  - Server-side container: Tomcat, Jboss, Jetty, Apache,…
  - Server-side templating: HAML, Moustache,…
  - Server-side MVC framework: Spring, Struts,…
  - Client-side JavaScript: jQuery, AngularJS, BackboneJS
  - **> 5 different technologies/languages**
- ✓ Building web applications with a MEAN approach
  - JavaScript and JSON
  - That's it!



The MEAN architecture



AngularJS

Our app's entire frontend

- ✓ Open source, maintained by Google
- ✓ Client Side MVC (MVVM) framework
- ✓ Excellent data bindings
- ✓ Very modular and extensible
- ✓ Great browser support (> IE8)
- ✓ Well documented
- ✓ Easy to test

Node.js

Lightweight web server framework

- ✓ Released in 2009 by Ryan Dahi
- ✓ Written in C/C++
- ✓ Built on Google Chrome's V8 JavaScript engine
- ✓ Extremely lightweight and efficient
- ✓ What JavaScript has done for the web browser, Node.js is doing for backend server
- ✓ Does module loading and Asynchronous IO
- ✓ Single threaded, One node process per CPU
- ✓ Easily scalable (just create a cluster)
- ✓ Great community and well maintained

Express

Create our routes and our API

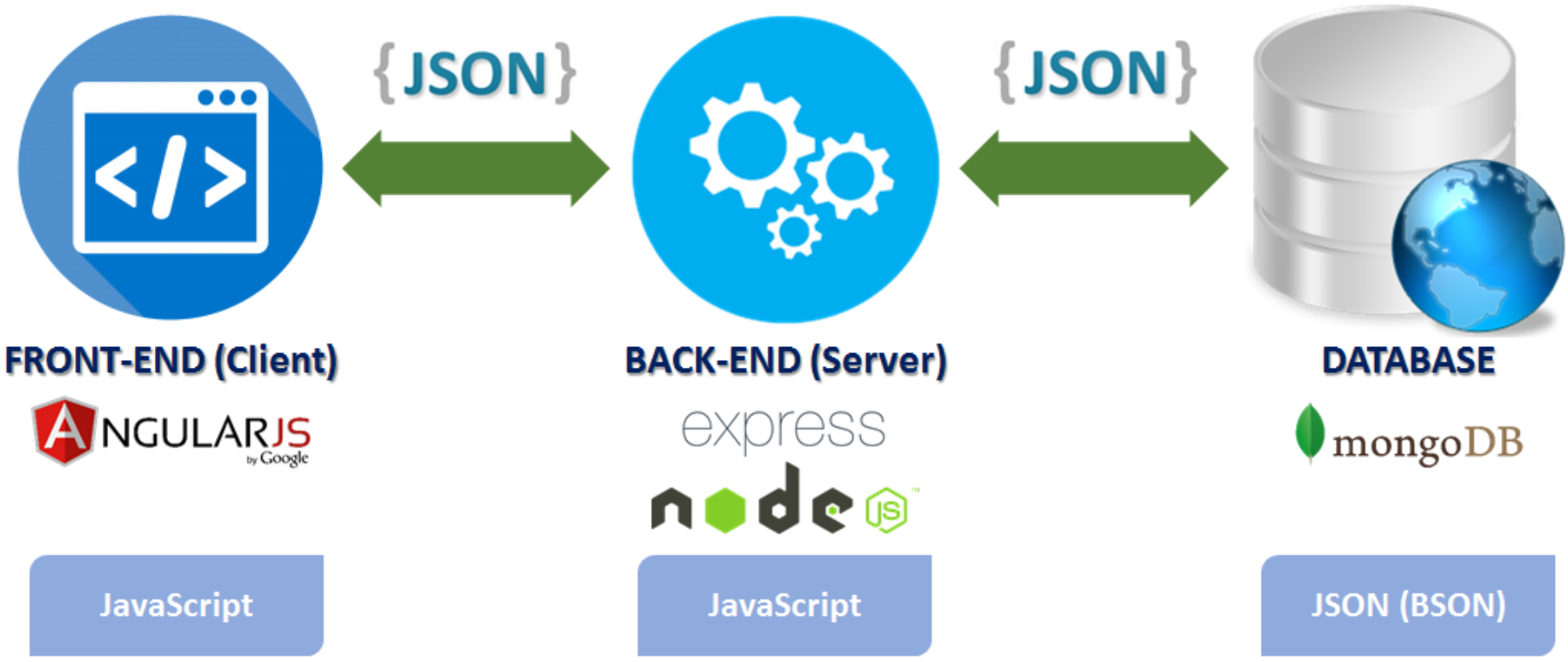
- ✓ A simple, minimal, robust and flexible web application framework for Node.js
- ✓ Gives everything we would expect to build a modern web server
- ✓ Abstracts away a lot of low level logic (e.g. for HTTP requests, SSL etc.)
- ✓ Helps organize your Node app into an MVC structure
- ✓ Easy to implement REST API and session management
- ✓ Middleware, routing, templating, static-files, cookies, mime-types and much much more

MongoDB

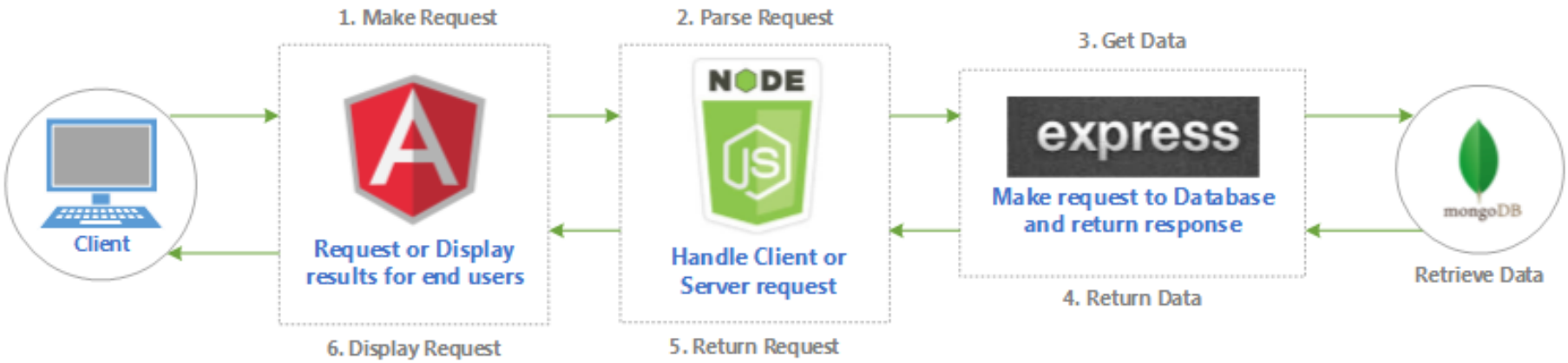
Our data store

- ✓ A simple and scalable document based top NoSQL Database
- ✓ Open Source, maintained by MongoDB
- ✓ JSON like syntax and model persistence
- ✓ Key-value stores
- ✓ Replication and High-Availability
- ✓ Flexible Schemas, Full Index support, Query Selectors, Sharding and much much more

MEAN Stack Language and Format Usage



MEAN Stack Concepts



Advantages of MEAN Stack over conventional Technology Stack

FRONT-END

BACK-END

DATABASE

CONVENTIONAL TECHNOLOGIES	HTML Rendering Decorative JS	Authentication Authorization Validation Business Logic HTML Generation	Data Storage Data Retrieval Business Logic
MEAN	HTML Rendering Decorative JS Business Logic HTML Generation	Authentication Authorization Validation	Data Storage Data Retrieval

MEAN Stack (Full Stack) Development Team

Front End

Back End

Database

Infrastructure

=

Full Stack

So why Go MEAN?

- ✓ It is 100% free, Open Source and Web Standards
- ✓ JavaScript all the way down
- ✓ Consistent models across stack (backend to frontend and back)
- ✓ Consistent best practices across stack
- ✓ Use a uniform language throughout your stack
  - JavaScript (the language of the web)
  - JSON (the data format of the web)
  - No conversion needed for the database
- ✓ Very low memory footprint/overhead
- ✓ Leverage JavaScript's popularity
- ✓ Huge community

Value of MEAN stack

- ✓ Universal end-to-end workflow

- ✓ Focus on speed of development

- ✓ Flexible pieces

MEAN Security

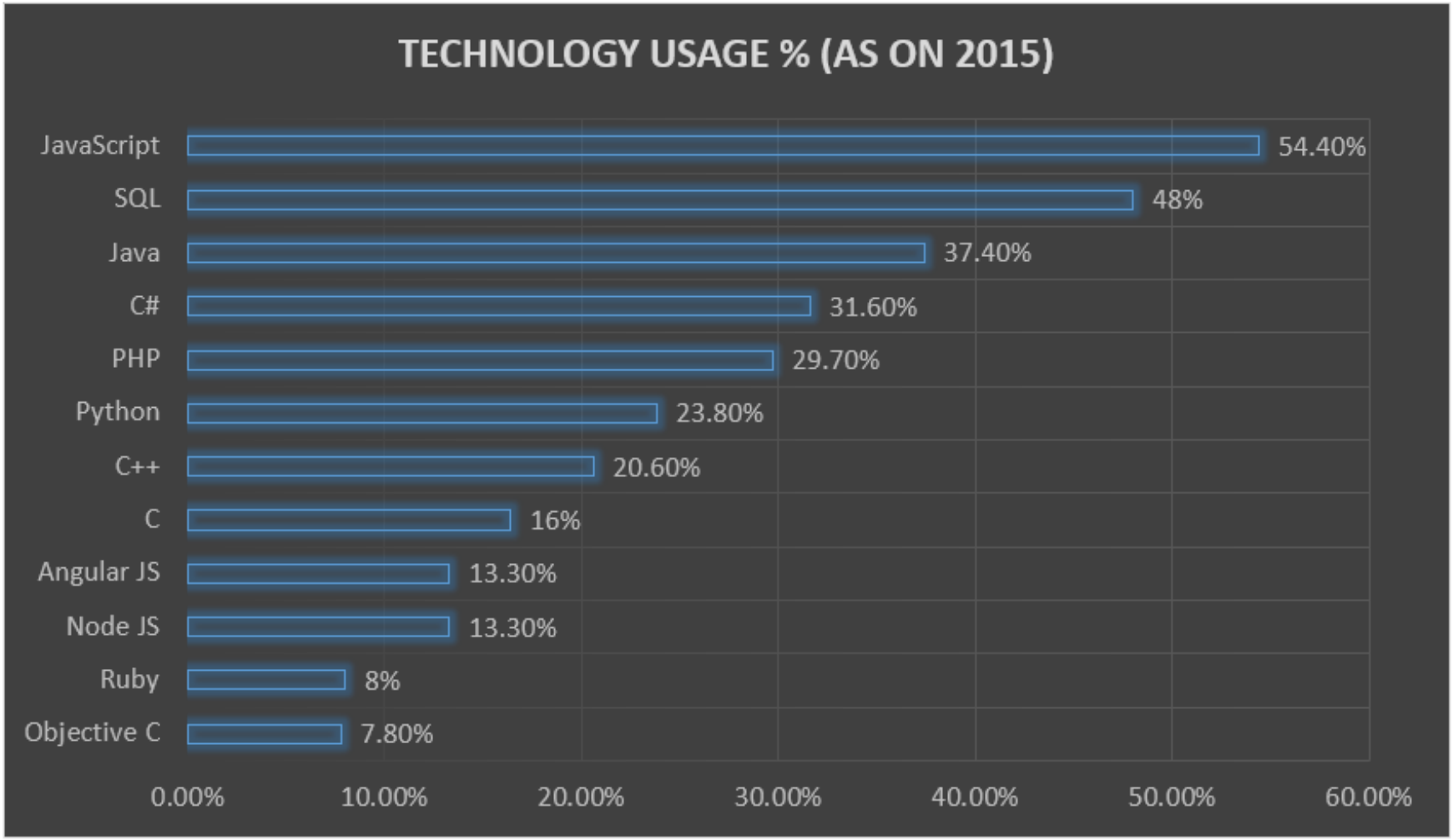
- ✓ It's as insecure as you code it ☺
- ✓ It's JavaScript so you have to be careful
  - Always use "strict" mode
  - Always declare variables
  - Always throw all errors
  - Always handle all callbacks (e.g. Success AND Error)
  - Use patterns for visibility (e.g. Module pattern)
  - Do static code analysis (JSHint/JSLint)
  - Always encode untrusted data

- ✓ Use Angular sanitizer ($sanitize) to sanitize an HTML string by stripping all potentially dangerous tokens
- ✓ And of course – never run any application as root! ☺
- ✓ Test, Test and Test

What's bad about this "MEAN" approach

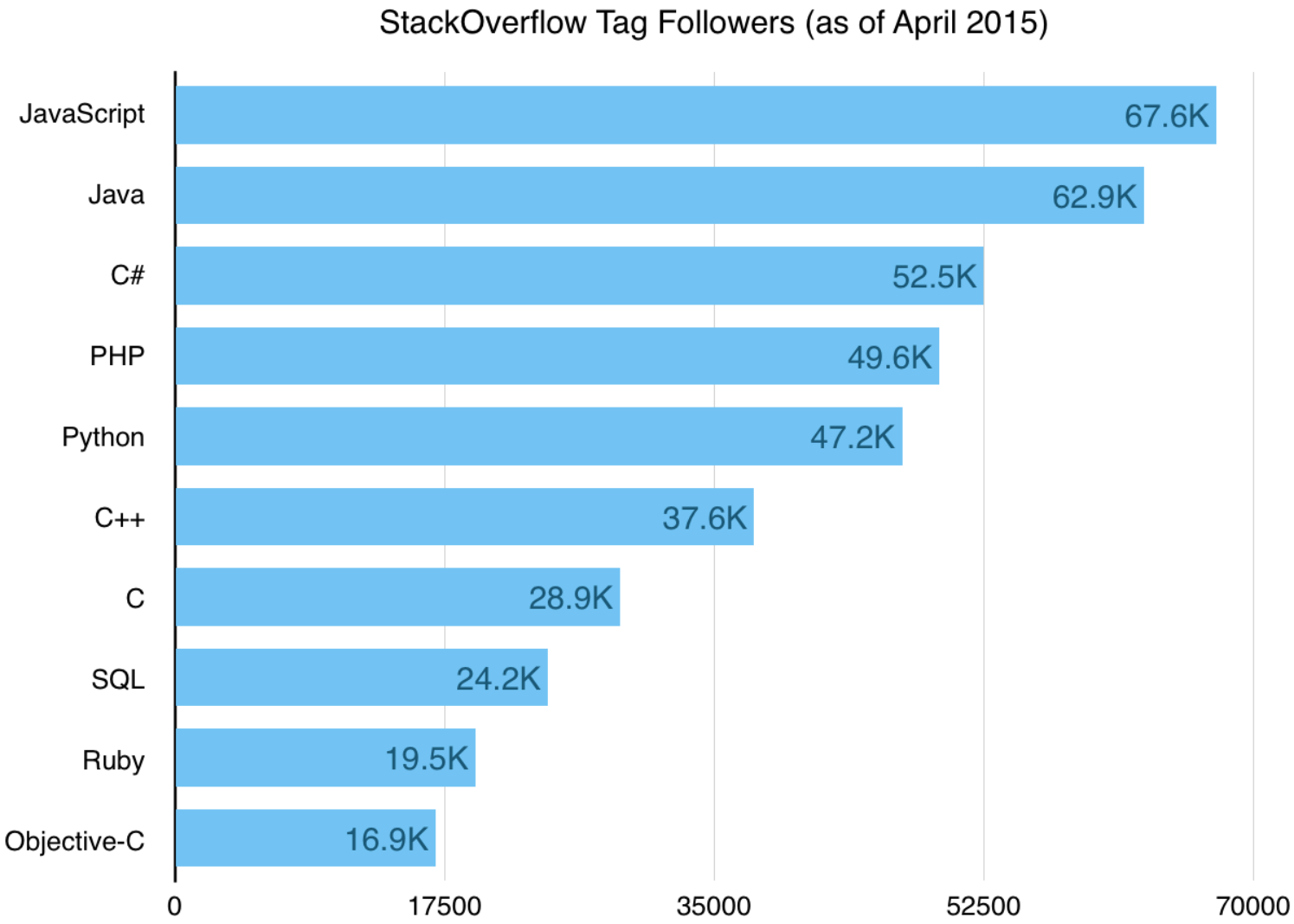
- ✓ For those who don't like JavaScript – Its going to be killing you ☺
- ✓ There is still no general JS coding guidelines
- ✓ SPA need a new way of thinking
  - The browser is your platform
- ✓ MongoDB is not as robust as an SQL Server is
  - This security is what they sacrifice to gain speed
- ✓ We need to take care of rollbacks our self (2-Phase-Commit)
  - ACID -> Possible but part of the driver/client application
- ✓ It's hard to find specialists that are no hipsters
  - Have the big picture/architecture for enterprise in mind
- ✓ It make sense to also have a look at Grunt, Yeoman and Bower in order to create good developer experience
- ✓ Once you have created the first application with this technology, it's hard to go back to the old approach

Future of MEAN Stack



Source: http://stackoverflow.com/research/developer-survey-2015

Future of MEAN Stack



Checkout http://stackoverflow.com/research/developer-survey-2016 for latest survey results