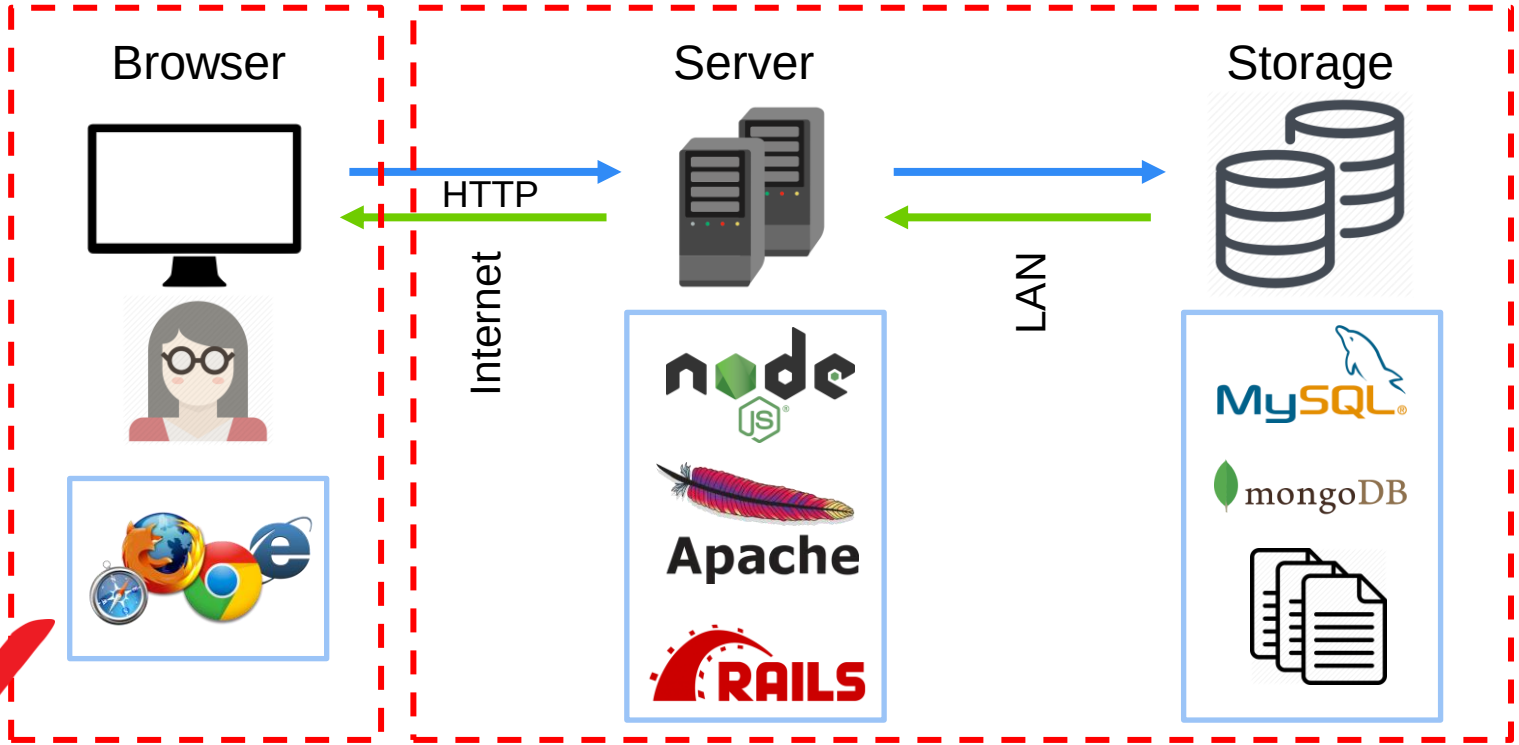
Australian National University

Full Stack Web Dev Overview

Comp1710/6780 Week 9

Dr Sabrina Caldwell
with thanks to Dr Xuanying Zhu
(xuanying.zhu@anu.edu.au)
School of Cybernetics
The Australian National University

Full Stack Web Application Architecture



Outline

01	Front End Web Development 
02	Back End Web Development
03	Common Stacks

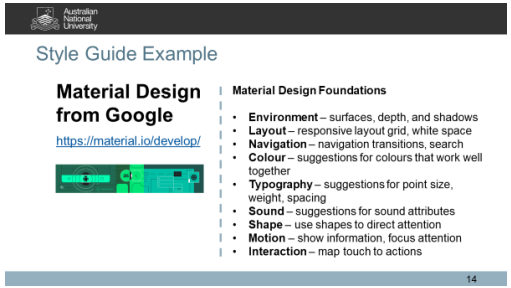
(This will be partially covered today, and finished on Thursday)

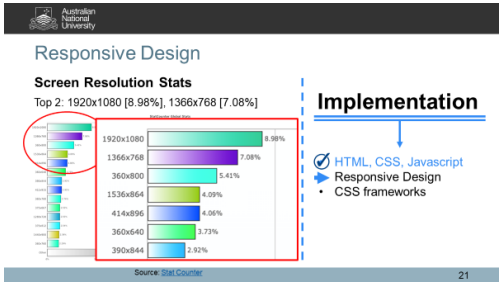
Good Front-end Applications

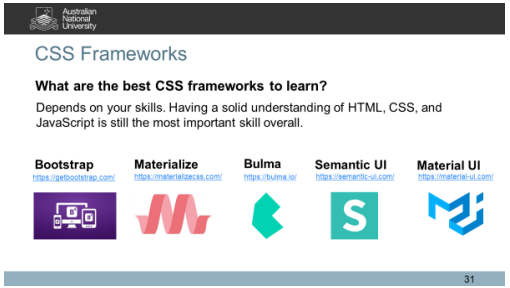
Design + Implementation

- Good & Bad Design
- Style Guide
  - e.g. Material Design

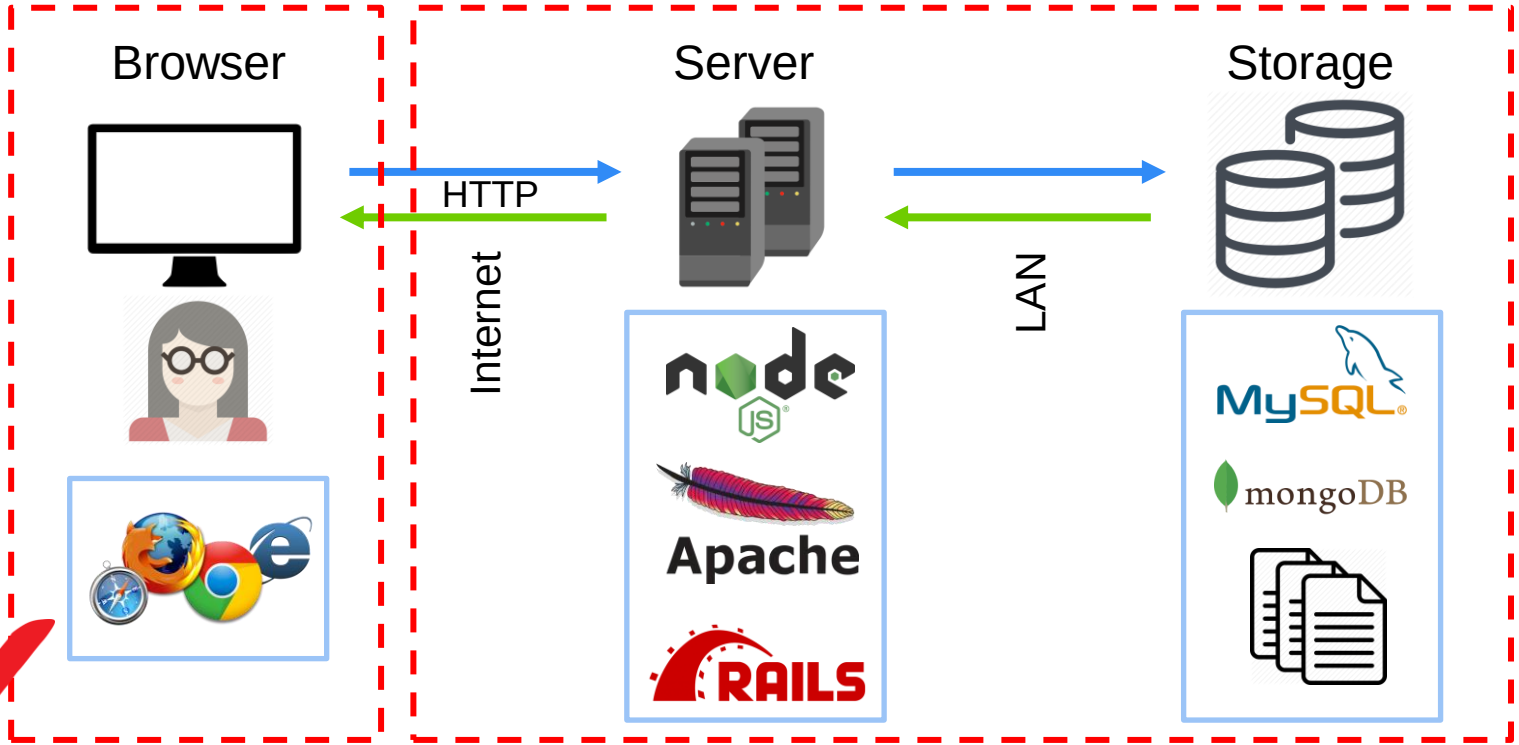
- HTML, CSS, Javascript
- Responsive Design
- CSS frameworks



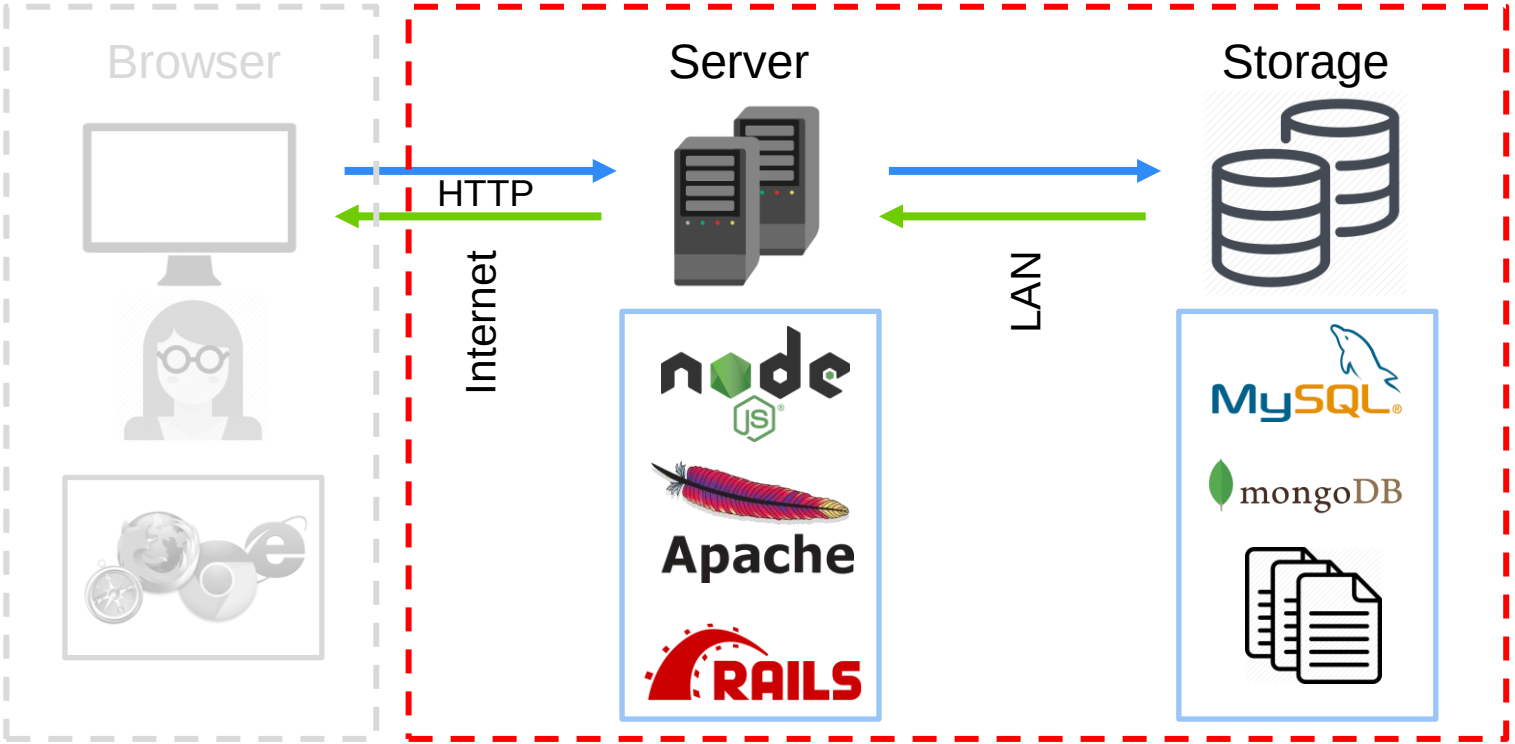




Full Stack Web Application Architecture

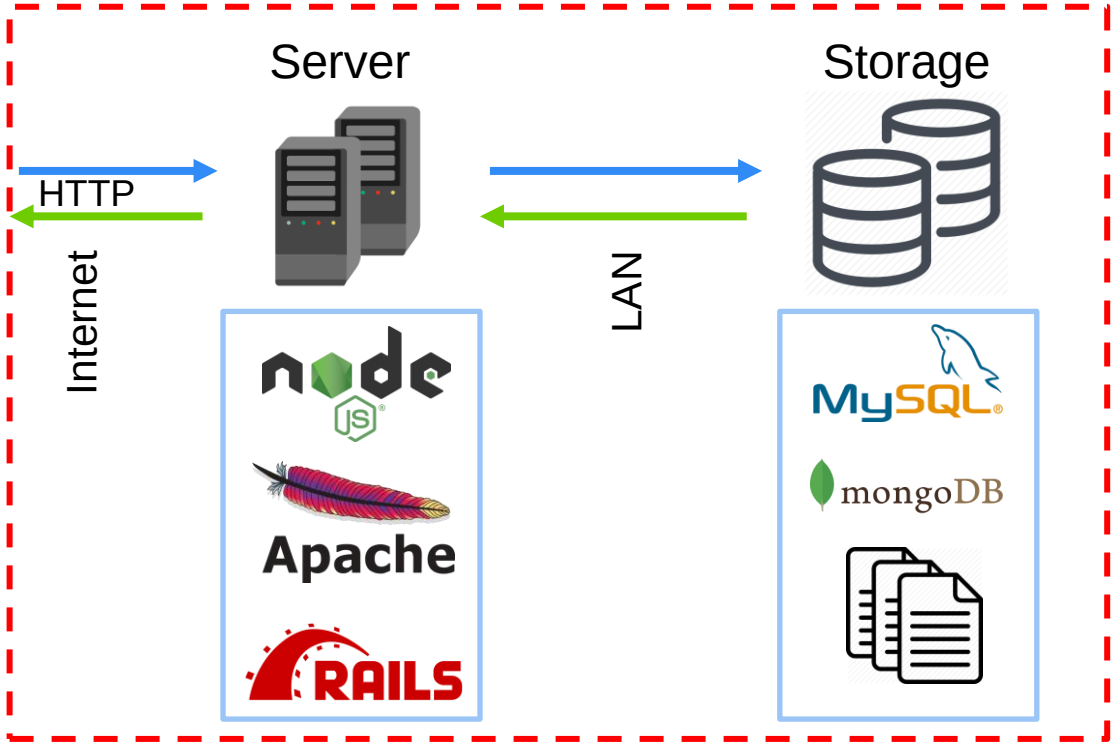


Full Stack Web Application Architecture

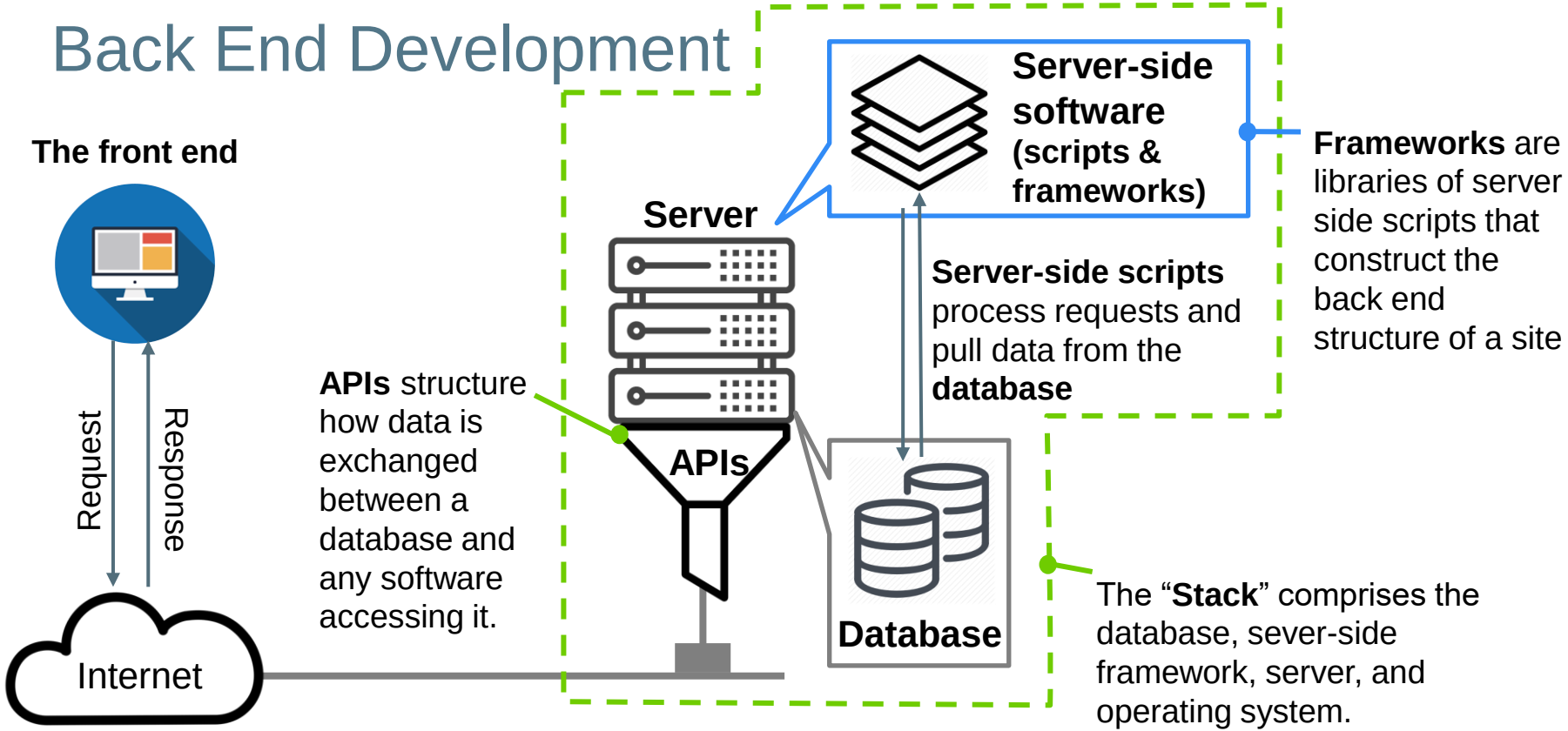


Back End Development

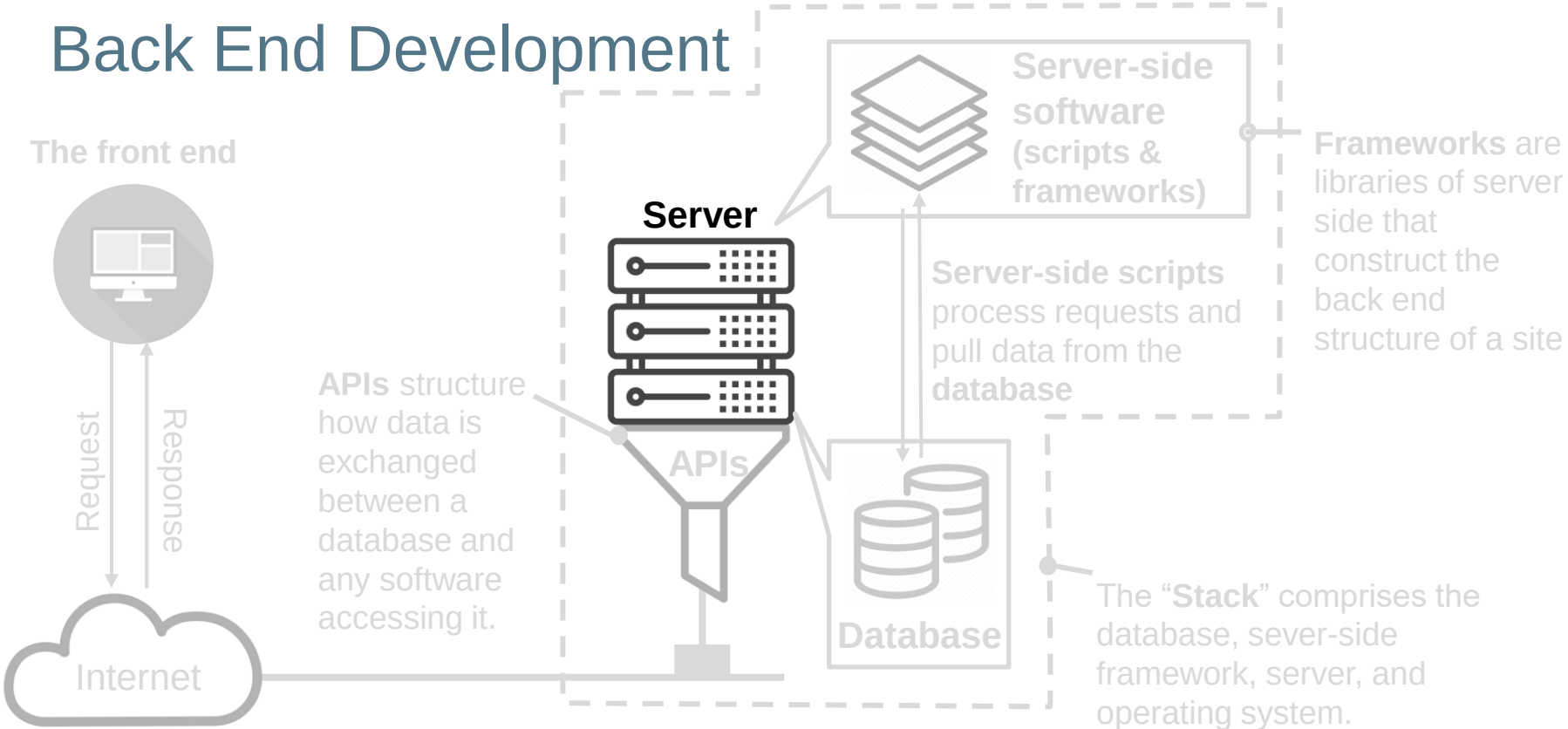
- Also called “Server-side” programming
- It is everything that happens on the server and databases
- Everything is behind the scenes



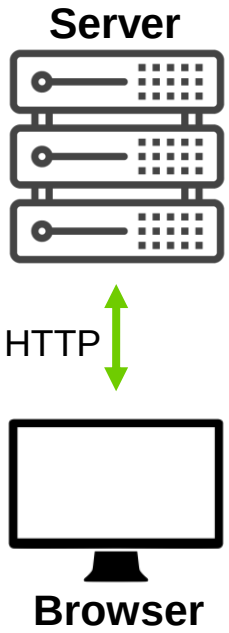
Back End Development



Back End Development



Server

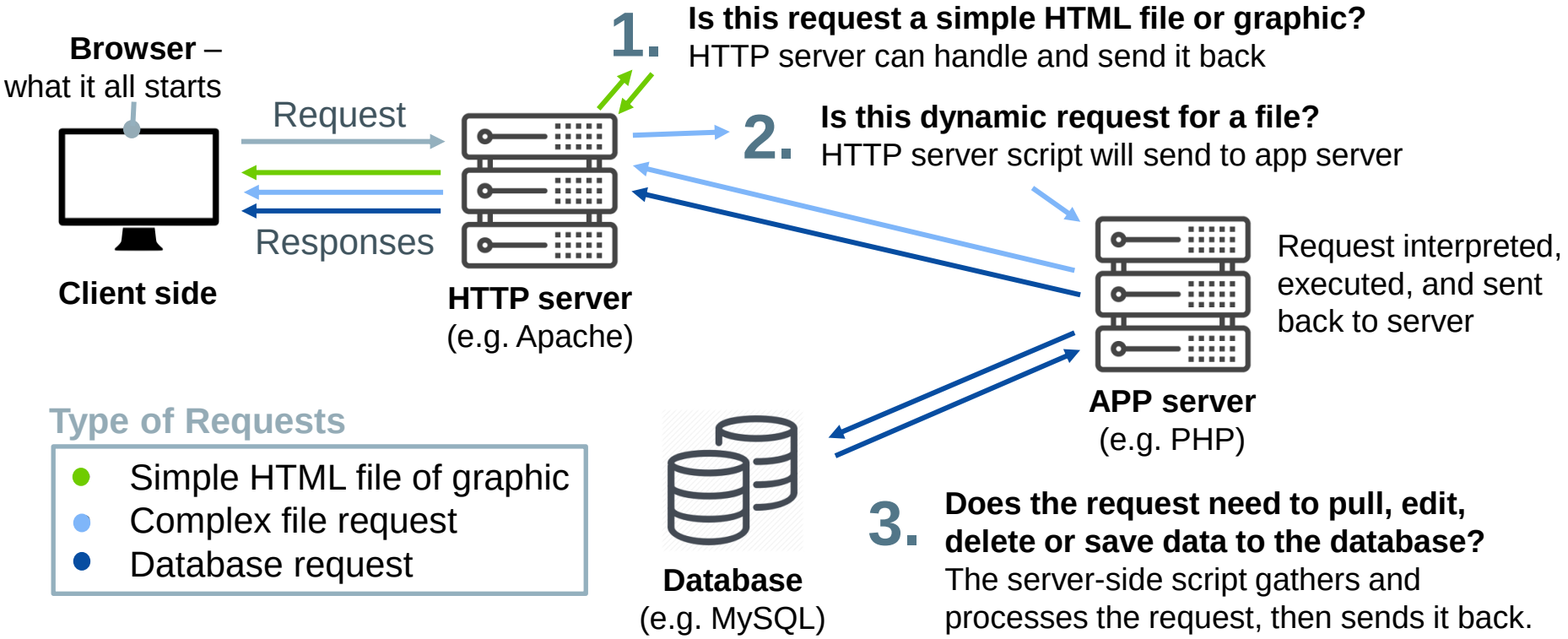


What do servers do?

Browsers: **send** HTTP requests and **get** HTTP responses

Servers: **get** HTTP requests and **send** HTTP responses

Server: Behind the Scenes



Other servers

Mail servers

Sending and storing emails network.
e.g. Microsoft Exchange Server

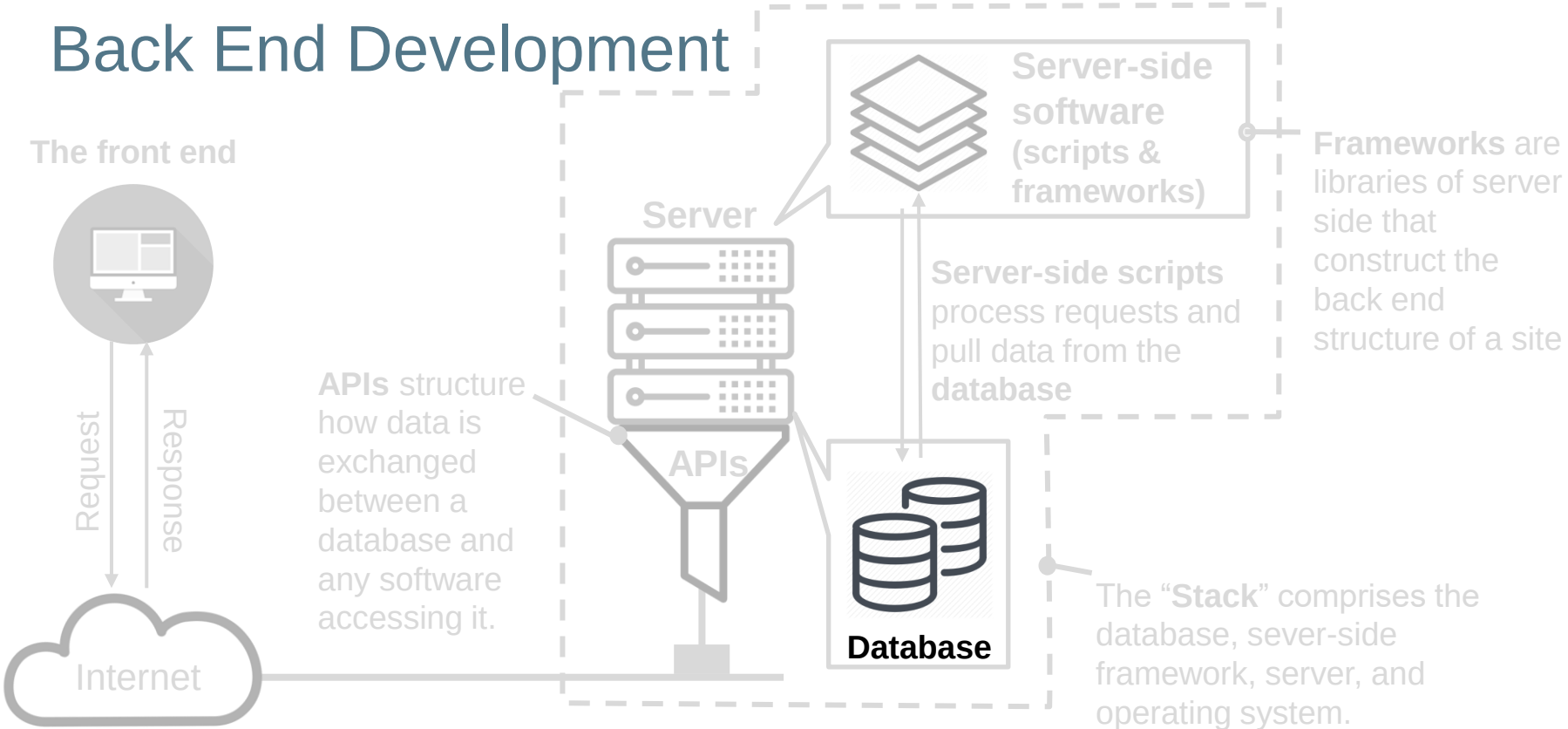
Proxy servers

These servers improve speed, security and performance between local network and the web by filtering requests and also providing cached versions of site pages to reduce network workload.
e.g. Nginx

FTP servers

All about uploading and sending files.
e.g. Filezilla FTPS

Back End Development



Database / Storage

Database

Databases store, organise and process information so that we can easily find what we need.

Properties

- Always available – Fetch correct app data and store updates
  - Even if many request come in concurrently – Scalable
  - Even if pieces fail – Reliable / fault tolerant
- Provide a good organisation of storing data
  - Quickly generate data for view
  - Handle app evolving over time

Database Type 1: Relational Database

Database

Relational Database

- Data is organised as a series of **tables** (also called **relations**)
- A table is made of **rows** (all called **records**)
- A row is made of a fixed (per table) set of typed **columns**

Relational Database Management System (RDBMS):

- **CRUD** functions: **C**reate, **R**ead, **U**pdate and **D**elete data
- Build-in programming language: **SQL** (Structured Query Language)
- e.g. MySQL, PostgreSQL, Microsoft SQL Server, Oracle

Ideal for

Organising and retrieving structured data

Database Type 2: NoSQL Database

Database

NoSQL Database

- NoSQL = “Not only SQL.”
- These databases are non-relational and distributed, addressing the issue that most modern data from the web is not structured.

How do NoSQL databases work?

Document-oriented: If a blog used a NoSQL database, each file could store data for a blog post: social likes, photos, text, links etc

Pros: Flexible & Ease of access (execute queries without SQL)
Cons: Require extra processing effort and more storage

Ideal for

Organising and retrieving inconsistent/incomplete data

SQL or NoSQL

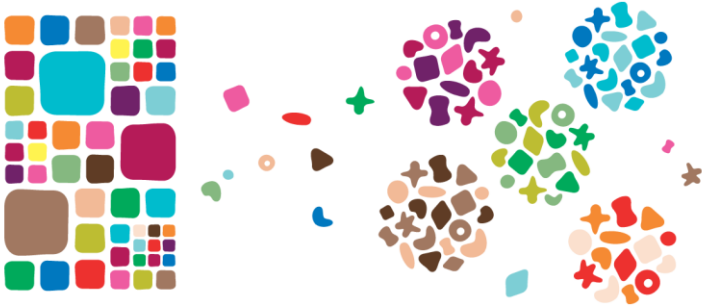
Database

Reasons to use a SQL database

Your data is structured and unchanging.

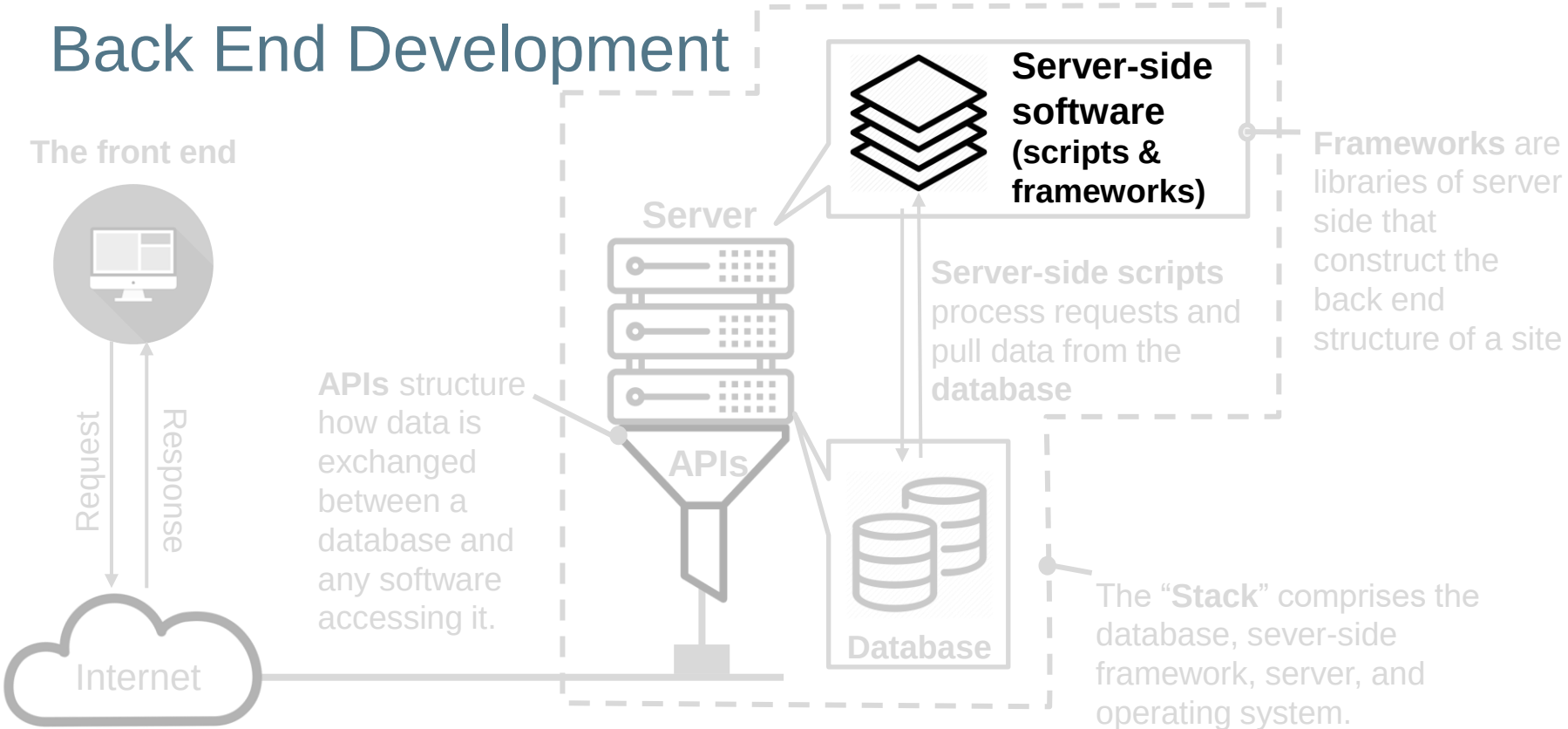
Reasons to use a NoSQL database

1. Storing large volumes of data that have little or no structure.
2. Making most of cloud computing and storage
3. Rapid development



Source: SQL vs NoSQL

Back End Development

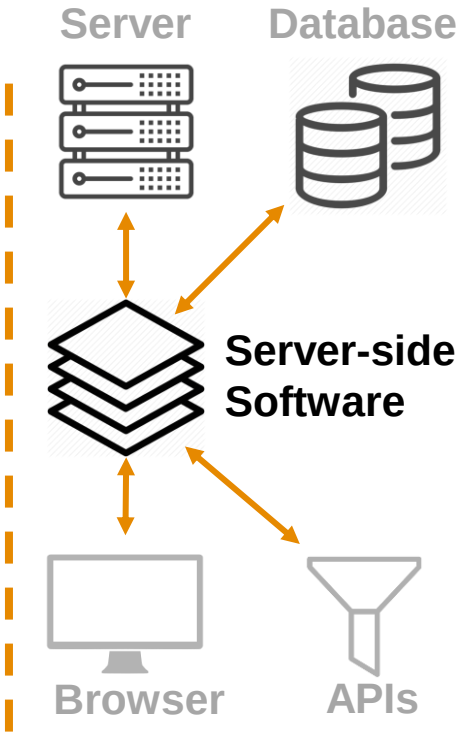


Server Side Software

- Also called “back-end web application”

Tasks

1. Provide application services based on business logic
2. Create the communication channel between browsers, server and storage system.
3. Build application programming interfaces (APIs), which control what data and software a site can share with other apps

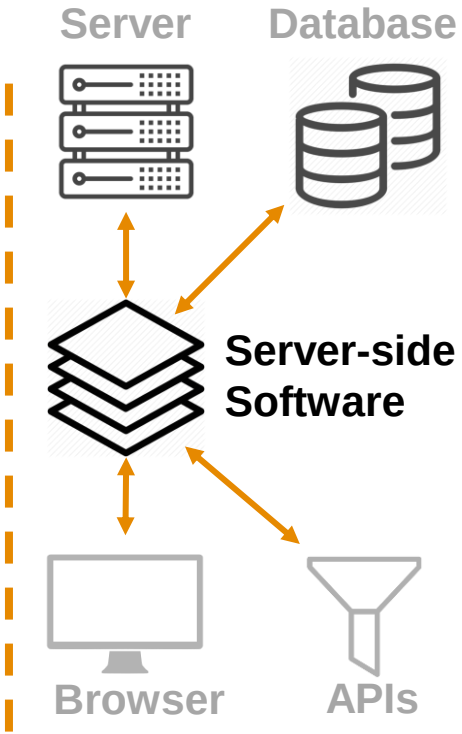


Server Side Software

- Also called “back-end web application”

Tasks

1. Provide application services bases on business logic
2. Create the communication channel between browsers, server and storage system.
3. Build application programming interfaces (APIs), which control what data and software a site can share with other apps



Server Side Software - Frameworks

A framework is a standardized set of concepts, practices and criteria for dealing with a common type of problem.

Why frameworks?
They boost performance, extend capabilities, and offer coding shortcuts that developers don't need to start from scratch.

Example:
When you're making a sandwich, it's much easier to buy pre-made, sliced bread from the store than it is to bake it on your own from scratch. Frameworks are your site's sliced bread—they speed up the process.

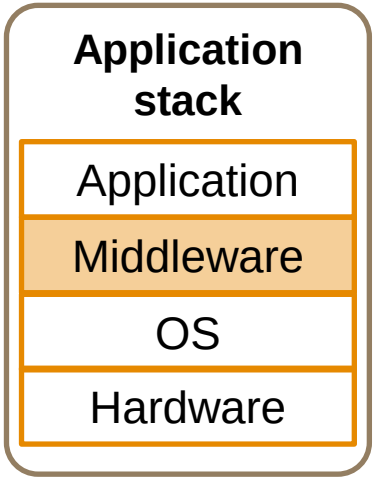
Server-side Software

Server Side Software - Middleware

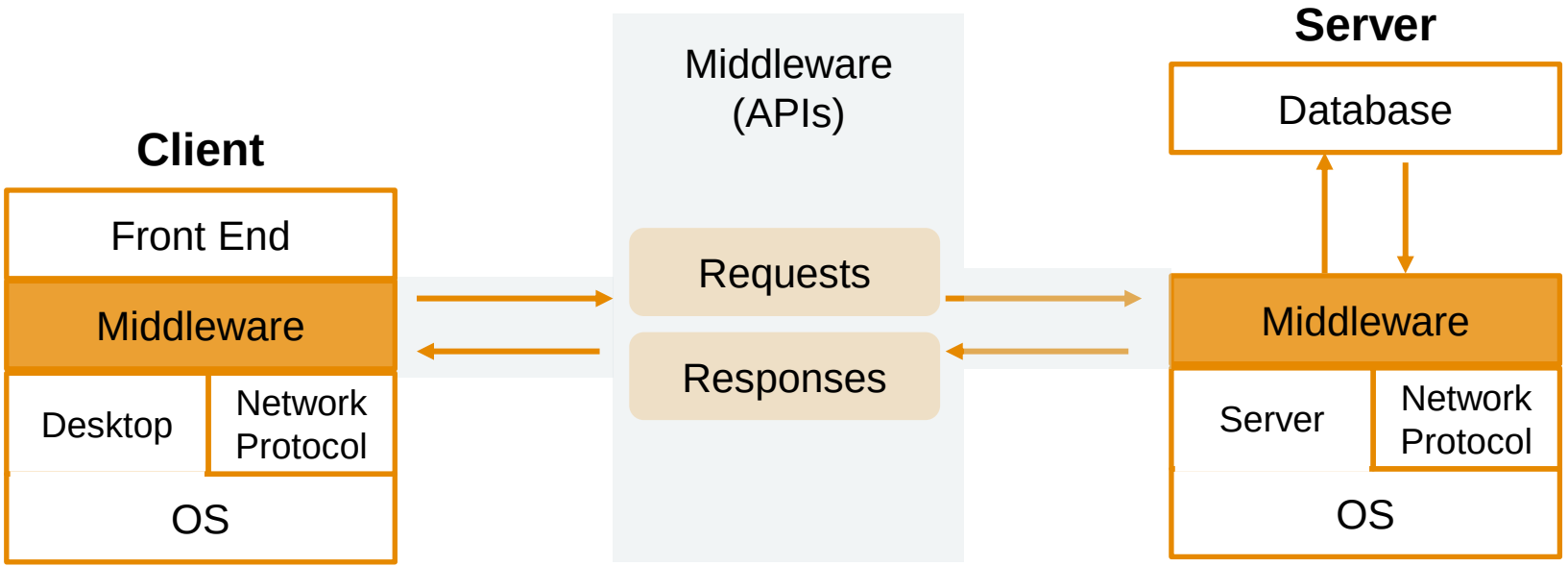
Middleware is any software that glues an application and its network.

Pros

- It facilitates client-server connectivity, forming a middle layer in the application that acts as “glue” between the app(s) and the network.
- It ties together complex systems and keeps all of the business’ software linked and able to communicate smoothly.
- It lets cloud applications and on-premise applications “talk” and provides services like data integration and error handling.



Server Side Software - Middleware

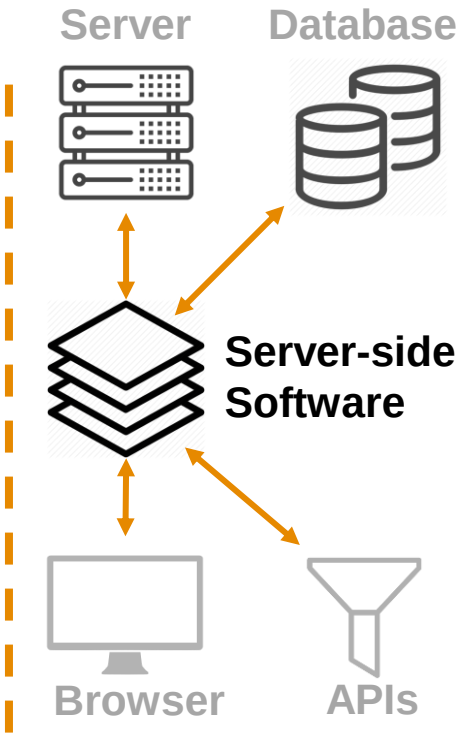


Server Side Software

- Also called “back-end web application”

Tasks

1. Provide application services bases on business logic
2. Create the communication channel between browsers, server and storage system.
3. Build application programming interfaces (APIs), which control what data and software a site can share with other apps



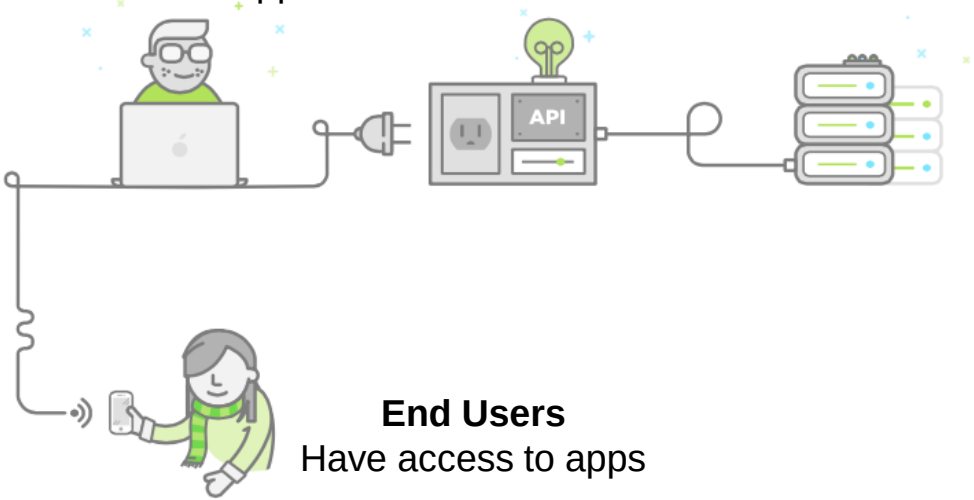
Server Side Software - APIs

An interface that allows two applications to talk to each other.

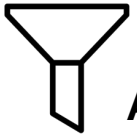
Developers
will access assets through API to build Mobiles or Web Apps

API
provides universal access to sharable assets

Assets
Data and software leveraged by developers and 3rd services

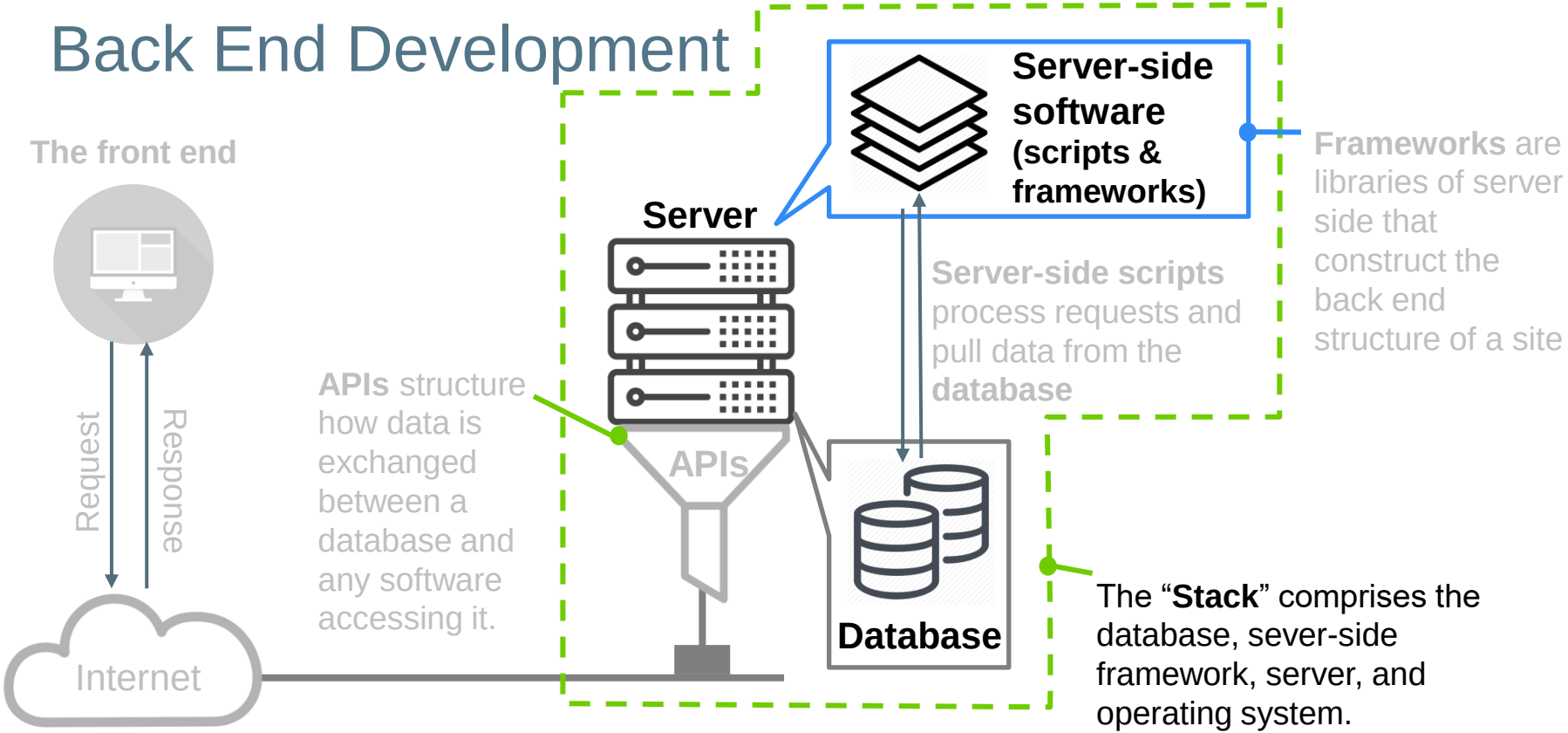


End Users
Have access to apps



APIs

Back End Development



Outline

01	Front End Web Development	
02	Back End Web Development	
03	Common Stacks	

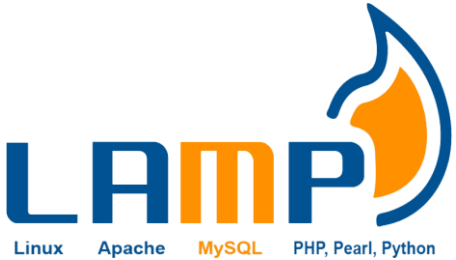
(This will be partially covered today, and finished on Thursday)

Common Stacks - LAMP

L: Linux operating system
A: Apache web server
M: MySQL database
P: PHP/Python/Perl application software

Pros

- Flexible, customizable, easy to develop and deploy
- A huge support community since it's open-source.
- Great for organizing massive amounts of structured data.



Variations

- WAMP: Windows/Apache/MySQL/PHP
- MAMP: Mac OS X/Apache/MySQL/PHP
- LAPP: Linux/Apache/PostgreSQL/PHP

Common Stacks – MVC Stacks

Django Stack: Python / Django / Apache / MySQL

- Rapid development, Simplify deploying Django software

Ruby Stack: Ruby / Ruby on Rails / Apache / MySQL

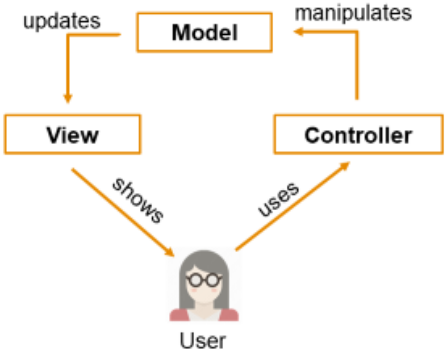
- Rapid development

Model-View-Controller (MVC) Pattern

Model:
- Manages application's data
- Connects the View and the Controller

View:
- Displays the web pages

Controller:
- Handles users' interactions
- Fetches models and controls views



Common Stacks - MEAN

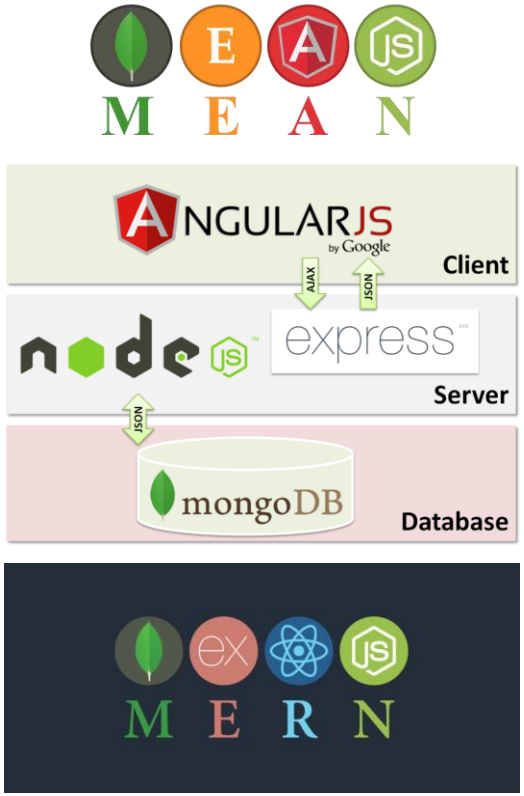
M: MongoDB
E: Express.js
A: AngularJS
N: Node.js

Pros

- Supports the MVC pattern
- Language uniformity.
- Document-based NoSQL database: more flexibility with semi-structured data.

Variation

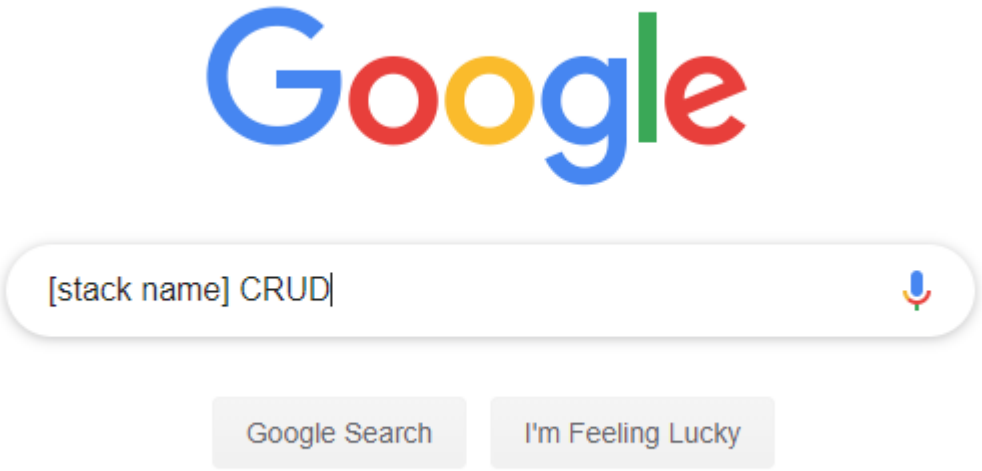
- MERN: MongoDB / Express.js / React.js / Node.js



Source: MEAN

Tips: a way to learn stacks

Google: stack name + “CRUD”



Tips: a way to learn stacks

- LAMP: https://www.taniarascia.com/create-a-simple-database-app-connecting-to-mysql-with-php/

- MVC stacks:
  - Django: https://rayed.com/posts/2018/05/django-crud-create-retrieve-update-delete/
  - Laravel: https://itsolutionstuff.com/post/laravel-57-crud-create-read-update-delete-tutorial-example-example.html
  - Ruby on Rails: https://medium.com/@nancydo7/ruby-on-rails-crud-tutorial-899117710c7a

- MEAN: https://appdividend.com/2018/11/04/angular-7-crud-example-mean-stack-tutorial/
- MERN: https://codingthesmartway.com/the-mern-stack-tutorial-building-a-react-crud-application-from-start-to-finish-part-1/

Outline

01	Front End Web Development	
02	Back End Web Development	
03	Common Stacks	

(This will be partially covered today, and finished on Thursday)

Full Stack Web Application Architecture