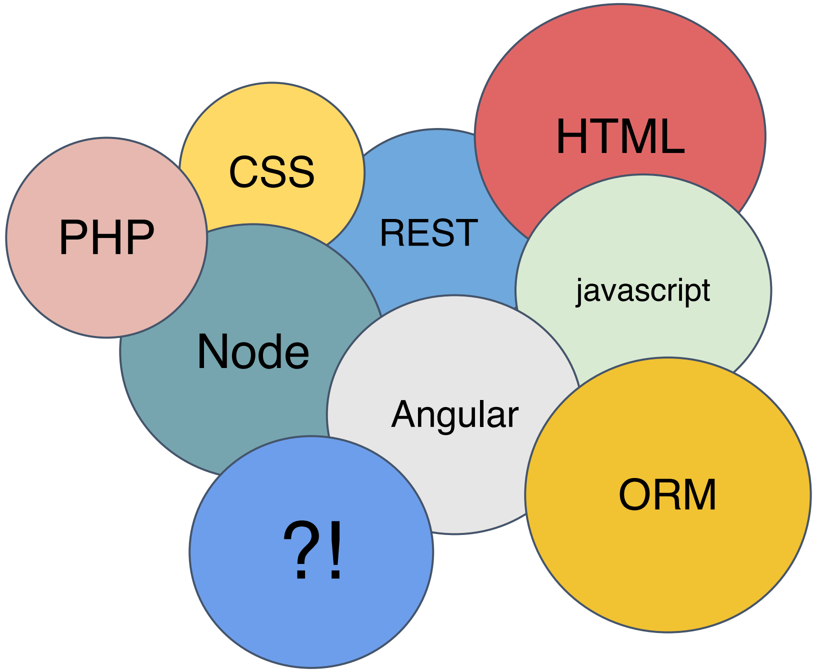
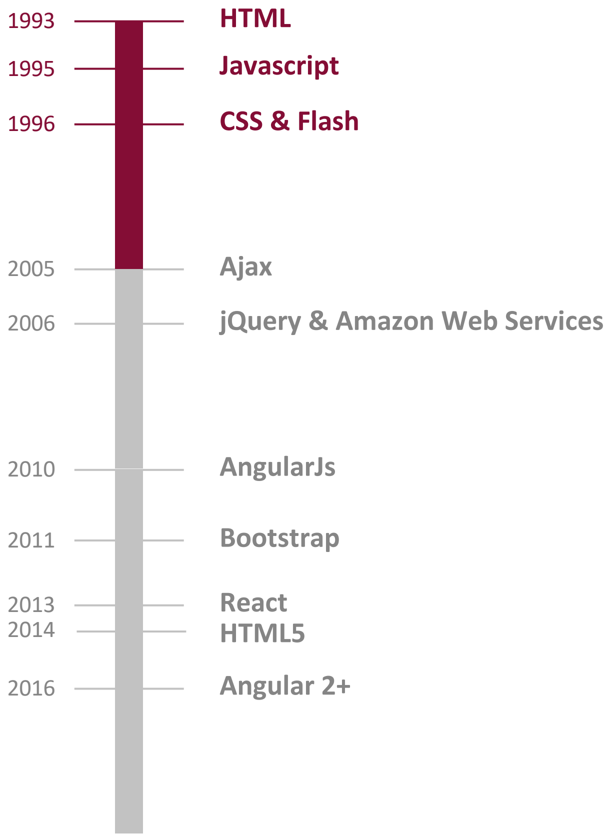
Practical web development basics



Prepared by Nitish Patkar

Important milestones in the history of web development



1993 HTML

1995 Javascript

1996 CSS & Flash

2005 Ajax

2006 jQuery & Amazon Web Services

2010 AngularJs

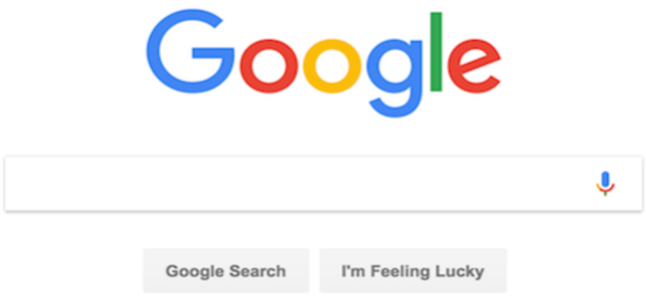
2011 Bootstrap

2013 React

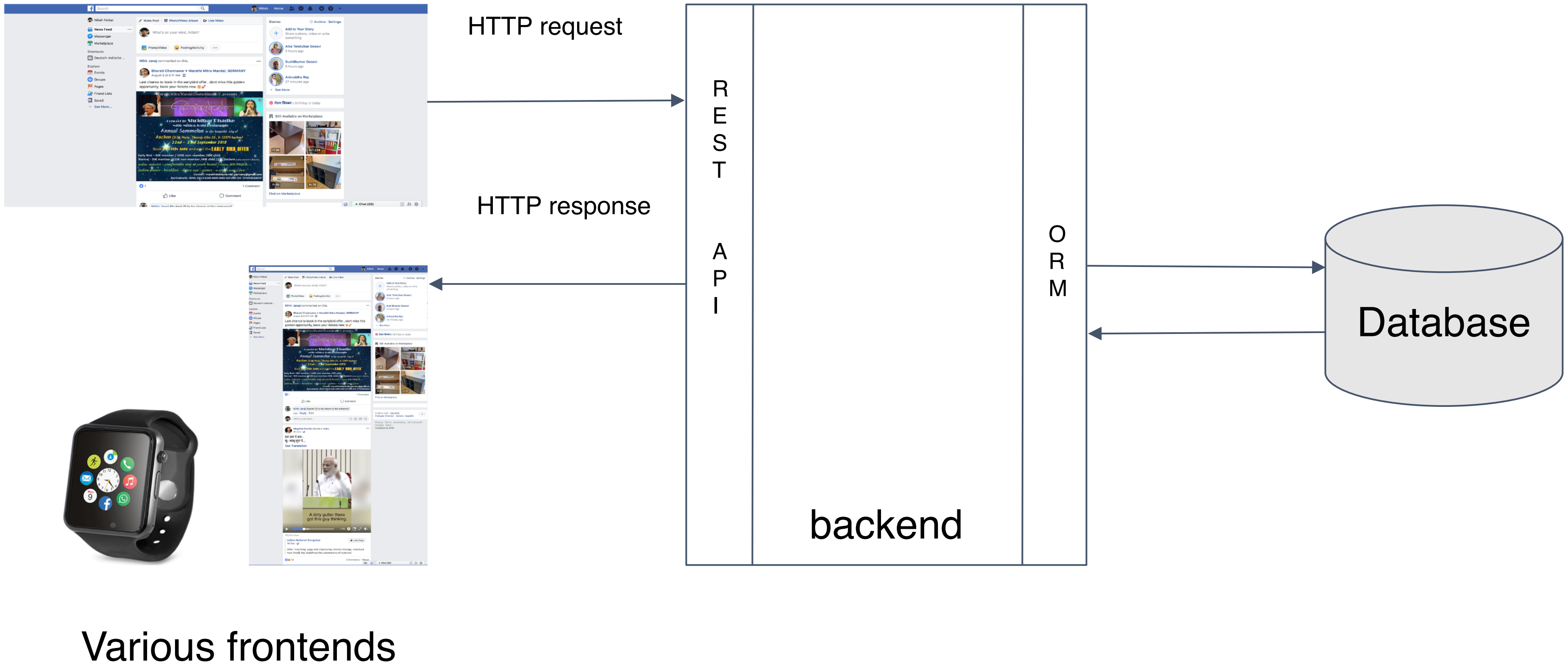
2014 HTML5

2016 Angular 2+

As a web developer you cannot live without



Simplified view on today's web development



Frontend: key responsibilities and main concepts

- It is what your users see- can be accessed on mobile as an app, or as a mobile website, or on laptops etc.
- Accepts request from users and forwards it to the backend, subsequently receives response from backend and shows it to the user

Main concepts to get familiar with-

- HTML, CSS, JavaScript, Typescript
- HTTP requests
- Ajax (or asynchronous calls to backend)
- JavaScript frameworks and libraries (e.g. Angular, React, Vue.js)
- Styling frameworks (e.g. Bootstrap, Material)
- Package manager (e.g. NPM)
- Single Page Applications (SPAs)
- Developer tool in browser for debugging
- Responsive design
- Browser compatibility
- …

Frontend: key responsibilities and main concepts

- It is what your users see- can be accessed on mobile as an app, or as a mobile website, or on laptops etc.
- Accepts request from users and forwards it to the backend, subsequently receives response from backend and shows it to the user

Main concepts to get familiar with-

- HTML, CSS, JavaScript, Typescript
- HTTP requests
- Ajax (or asynchronous calls to backend)
- JavaScript frameworks and libraries (e.g. Angular, React, Vue.js)
- Styling frameworks (e.g. Bootstrap, Material)
- Package manager (e.g. NPM)
- Single Page Applications (SPAs)
- Developer tool in browser for debugging
- Responsive design
- Browser compatibility
- …

Backend: key responsibilities and main concepts

- ❖ Handles business logic for your application- you write a single backend application which is typically consumed by various frontends
- ❖ Coordinates with frontend and database

Main concepts to get familiar with-

- REST API
- JSON data format
- Programming languages (e.g. PHP, Python, Java)
- Backend frameworks (e.g. Django Rest, .NET core, Ruby on Rails, Express)
- Object Relational Mapping (ORM)
- Request routing
- Security concerns
- Architecture (e.g. MVC, MVVM)
- Session management
- ...

Backend: key responsibilities and main concepts

- Handles business logic for your application- you write a single backend application which is typically consumed by various frontends
- Coordinates with frontend and database

Main concepts to get familiar with-

- REST API
- JSON data format
- Programming languages (e.g. PHP, Python, Java)
- Backend frameworks (e.g. Django Rest, .NET core, Ruby on Rails, Express)
- Object Relational Mapping (ORM)
- Request routing
- Security concerns
- Architecture (e.g. MVC, MVVM)
- Session management
- ...

Database: key responsibilities and main concepts

- Stores user and application data
- With the availability of ORMs, developers deal less and less with databases directly, but it is still good to know

Main concepts to get familiar with-

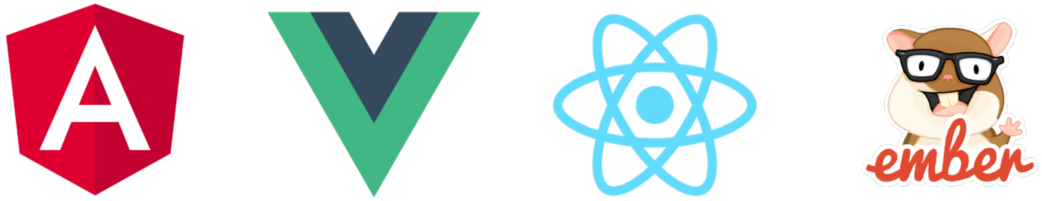
- Types of databases (e.g. Relational, non-relational)
- Database venders (e.g. MySQL, PostgreSQL, MongoDB)
- Database backups
- Database query languages (e.g. SQL)
- Scripting languages (e.g. PL/SQL)
- ...

Frameworks

- One can do web development without frameworks, but using them simplifies many routine tasks and speeds up the development, also makes sure applications are scalable
- Choosing a correct framework depends on multiple factors such as programming language preference, framework documentation, community support etc.
- There are different frameworks for frontend development, backend development, and also for styling

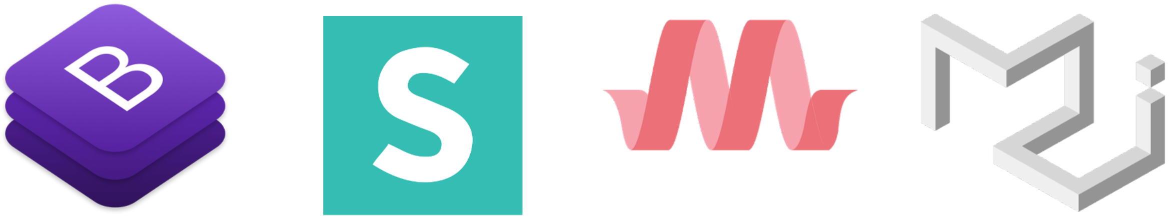
Frontend (JavaScript) frameworks

- Enforces developers to write HTML, CSS and JavaScript in a certain way to make the application easier to develop and maintain
- Easy handling of Ajax calls, HTTP requests
- Most popular frontend frameworks and libraries:



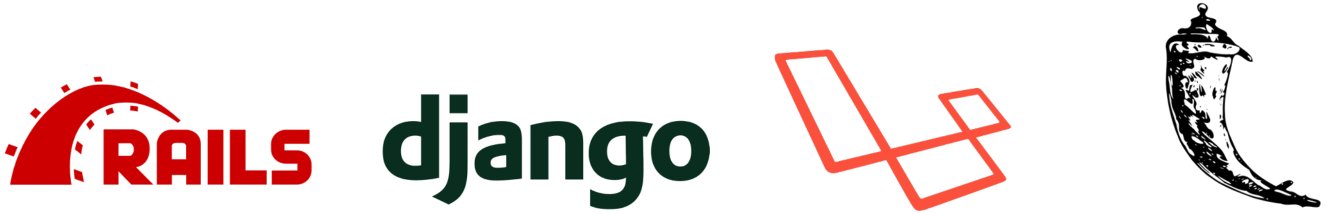
Styling (CSS) frameworks

- Provides ready to use CSS classes, components and grid system based on proven design standards, so that developers do not have to write CSS from scratch
- Most popular CSS frameworks:



Backend frameworks

- They provide tools and libraries that simplify common web development tasks, including routing URLs to appropriate handlers, interacting with databases through ORM, supporting sessions and user authorization, formatting output (e.g. HTML, JSON, XML), and improving security against web attacks.
- Most popular backend frameworks:



	Frontend	Backend
Concepts to get familiar with	HTTP requests, responsive design, asynchronous calls, SPAs, NPM,...	HTTP requests routing, REST API, ORM, token & session management, CSRF, ...
Languages used	HTML, CSS, Javascript, Typescript	Python, Java, javascript,...
Popular frameworks	Angular, React, Vue.js,..	Django, .NET core, Laravel, Express,...
Popular styling frameworks	Bootstrap, Material UI, Saas,...	-

Project scaffolding

- Helps to quickly set up a project skeleton (hopefully) with best practices
- One can scaffold a project by using CLI tools such as angular cli for angular projects, or by cloning a repository from GitHub, GitLab etc.
- Best known scaffolding tools:

Yeoman

Slush

Project management tools

- Helps to organize development work within teams, manage user stories, deadlines, generate project analytics and many more tasks
- Best known project management tools:

Trello Confluence JIRA

Communication tools

- Helps to communicate with other team members through dedicated channels, direct messages and rich content such as video, images, Gif etc.
- Provide search functionality and integrations to other applications such as confluence, GitLab etc.
- Best known communication tool:

slack

CI/CD pipeline

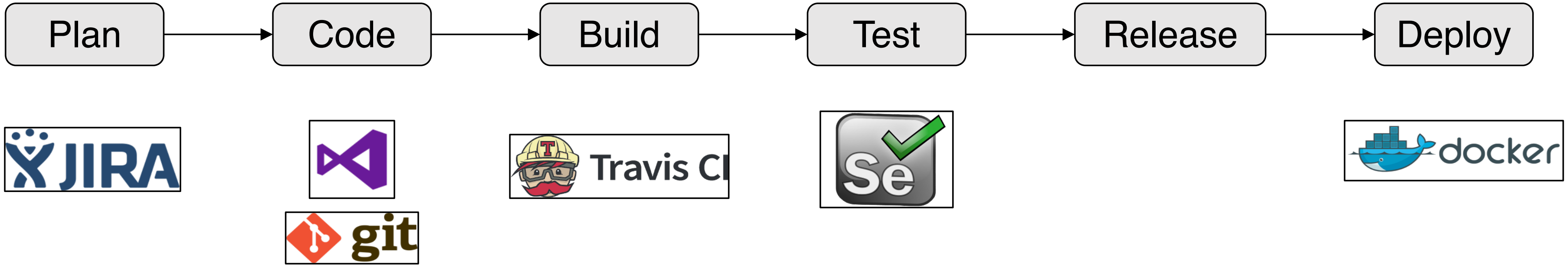
- Continuous Delivery (CD) is a software strategy that enables organizations to deliver new features to users as fast and efficiently as possible

- The pipeline breaks down the software delivery process into stages. Each stage is aimed at verifying the quality of new features from a different angle

- There is no standard pipeline however a typical process will look like following:

- A pipeline procedure is triggered when code is committed to a repo hosted somewhere like GitHub. Next comes notification to a build system, such as Jenkins. The build system compiles the code and runs unit tests. If your pipeline is built well, and unit tests go smoothly, at this point integration tests are executed. After integration testing is done, you can create images and push them to a registry service, such as Docker Hub. From there, they can be easily deployed on cloud services such as AWS or Google cloud.

CI/CD pipeline: example



Plan → Code → Build → Test → Release → Deploy

Other end-to-end solutions

GitLab

Jenkins

End.