**ANGULAR FRAMEWORK**

**AngularJS** is a very powerful JavaScript Framework, it works by first reading the HTML page, which has embedded into it additional custom tag attributes, it aims purpose is to simplify both development and testing of such applications by providing a frame work for the client-side model-view-architecture(MVC) and model–view-view-model (MVVM) architecture.

AngularJS is a basic system for element web applications. It gives you a chance to utilize HTML as your layout and gives you a chance to extend HTML structure to express your application's segments clearly.

Angular mainly functions on this domains-

* Angular Application.
* Angular Directives.
* Angular Services.
* Angular Controllers.
* Angular Filters

**Angular Application**: For the HTML document file we use the ng-app for bootstrapping an application. We use it for single page application here in it window object is not changing but contents in it do change.

**Angular Directives**: It is a custom attribute or an element defined inside a java script which is used for DOM manipulations.

**Angular Services**: Angular services are substitutional objects that are bind together using dependency injection, they are certain part of java script to write the business logics. It is also used for sharing information between controllers. They are classified into two types- 1. Singleton service 2. Class based service

**Angular Controllers:** Controllers mainly control the flow of data in the application. A controller is defined using ng-controller directive. It controls the view in the html page.

**NODE JS**

**What is Node.Js?**

Top of Form

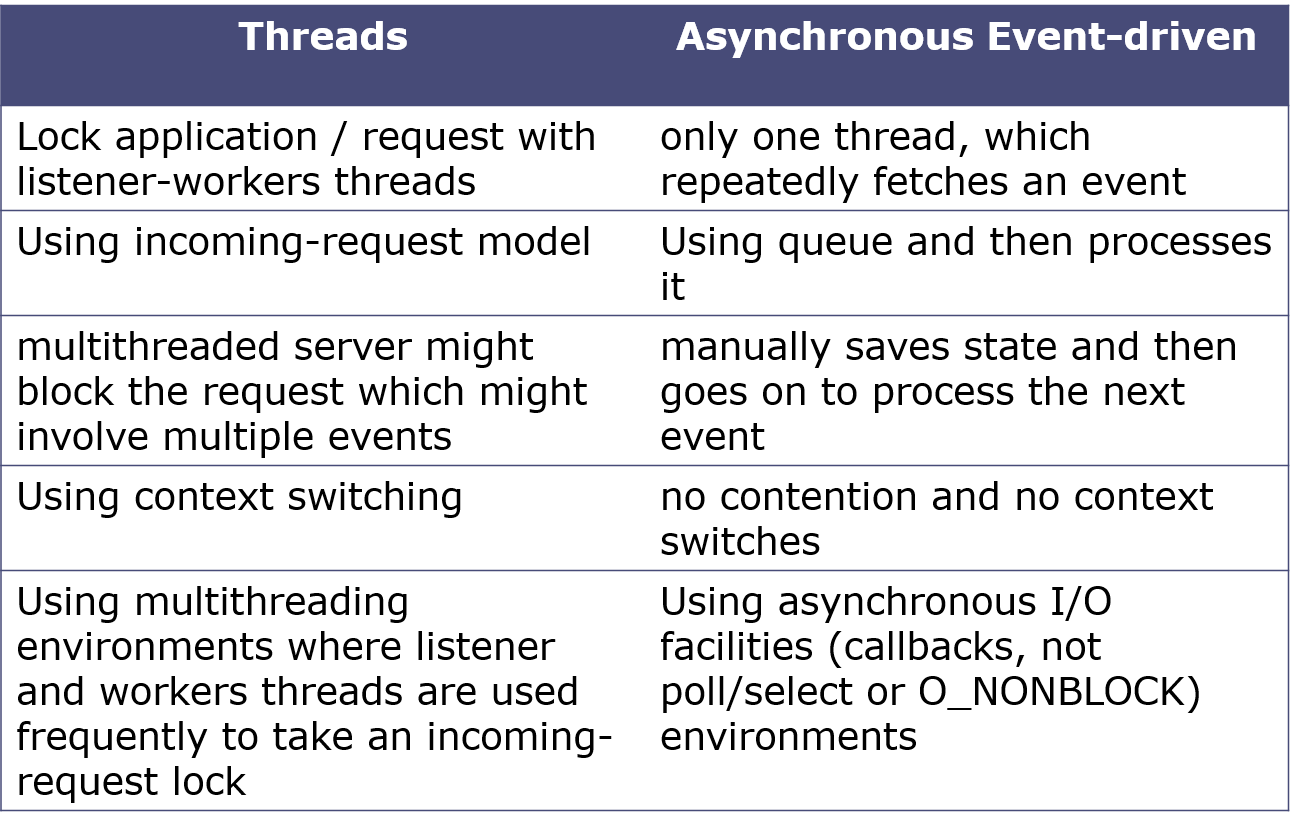
Bottom of Form

Node.js is a very powerful JavaScript-based framework platform built on Google Chrome's JavaScript Engine. It is used to develop input-output intensive web applications like video streaming sites, single-page applications, and other web applications.

**Concurrency: The Event Loop**

* Instead of threads Node uses an event loop with a stack.
* Alleviates overhead of context switching.

**Threads VS Event-driven**



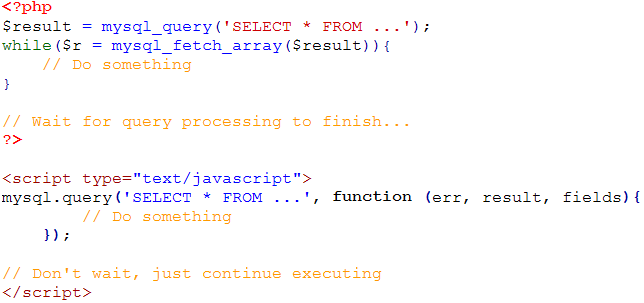
**Event Loop Example**

* Request for “index.html” comes in.
* Stack unwinds and ev\_loop goes to sleep.
* File loads from disk and is sent to the client.

**Non-blocking I/O**

* Servers do nothing but I/O.
* Scripts waiting on I/O requests degrades performance.
* To avoid blocking, Node makes use of the event driven nature of JS by attaching callbacks to I/O requests.
* Scripts waiting on I/O waste no space because they get popped off the stack when their non-I/O related code finishes executing.

**I/O Example**



**Consistency**

* Use of JS on both the client and server-side should remove need to “context switch”
  + Client-side JS makes heavy use of the DOM, no access to files/databases
  + Server-side JS deals mostly in files/databases, no DOM.
    - JSDom project for Node works for simple tasks, but not much else.

**Mongo dB**

MongoDB is an open-source document database, and leading NoSQL database. MongoDB is written in C++.

**Why should use MongoDB**

* Document Oriented Storage: Data is stored in the form of JSON style documents Index on any attribute.
* Replication & High Availability.
* Auto-Sharding.
* Rich Queries.
* Fast In-Place Updates.
* Professional Support by MongoDB.

**Where should use MongoDB?**

1. Big Data. 2) Content Management and Delivery. 3) Mobile and Social Infrastructure. 4) User Data Management. 5) Data Hub.

**What it does, how it works**

* MongoDB is a server process that runs on Linux, Windows and OS X.
  + It can be run both as a 32 or 64-bit application. We recommend running in 64-bit mode, since Mongo is limited to a total data size of about 2GB for all databases in 32-bit mode.
* Clients connect to the MongoDB process, optionally authenticate themselves if security is turned on, and perform a sequence of actions, such as inserts, queries and updates.
* MongoDB stores its data in files (default location is/data/db/), and uses memory mapped files for data management for efficiency.

**GIT commands**

|  |  |
| --- | --- |
| [**Tell GIT who you are**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-config) | git config --global user.name "Sam Smith"  git config --global user.email sam@example.com |
| [**Create a new local repository**](http://atlassian.com/git/tutorial/git-basics#!init) | git init |
| [**Check out a repository**](http://atlassian.com/git/tutorial/git-basics#!clone) | git clone /path/to/repository |
| git clone username@host:/path/to/repository |
| [**Add files**](http://atlassian.com/git/tutorial/git-basics#!add) | git add <filename>  git add \* |
| [**Commit**](http://atlassian.com/git/tutorial/git-basics#!commit) | git commit -m "Commit message" |
| git commit -a |
| [**Push**](http://atlassian.com/git/tutorial/remote-repositories#!push) | git push origin master |
| [**Status**](http://atlassian.com/git/tutorial/git-basics#!status) | git status |
| [**Connect to a remote repository**](http://atlassian.com/git/tutorial/remote-repositories#!remote) | git remote add origin <server> |
| git remote -v |
| [**Branches**](http://atlassian.com/git/tutorial/git-branches) | git checkout -b <branchname> |
| git checkout <branchname> |
| git branch |
| git branch -d <branchname> |
| git push origin <branchname> |
| git push --all origin |
| git push origin :<branchname> |
| [**Update from the remote repository**](http://atlassian.com/git/tutorial/remote-repositories) | git pull |
| git merge <branchname> |
| git diff  git diff --base <filename>  git diff <sourcebranch> <targetbranch> |
| git add <filename> |
| **Tags** | git tag 1.0.0 <commitID> |
| git log |
| git push --tags origin |
| [**Undo local changes**](http://atlassian.com/git/tutorial/undoing-changes) | git checkout -- <filename> |
| git fetch origin  git reset --hard origin/master |
| **Search** | git grep "foo()" |