Application frameworks

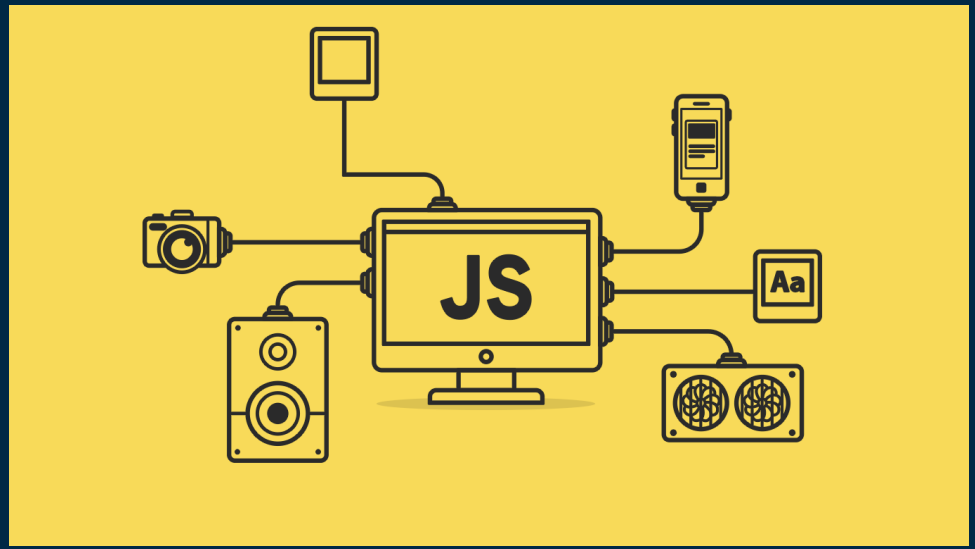
JavaScript, Version controlling and NoSQL

Overview

- JavaScript
- Version controlling
- NoSQL

JavaScript

- Introduction
- Classes, objects and prototype
- How ‘this’ acts
- Strict notation
- Function closure
- Callbacks and promises



JavaScript

- By default, JavaScript programs run using a single thread. Though there are ways to create new threads JavaScript is considered as Single threaded language.
- JavaScript does not wait for I/O operations to get completed, instead it continues the execution of the program. This is called as non-blocking I/O.
- JavaScript is asynchronous because of the NIO nature.
- JavaScript is dynamically typed. It determines variable types, ordering etc. in runtime.
- JavaScript support OOP as well as functional programming (Multi-paradigm).
- JavaScript has an eventing system which manages it’s asynchronous operations.

Classes and objects

- In JavaScript, a constructor function is used with the ‘new’ key keyword when creating a Object.
- Constructor function is a just another function.
- When function is used with ‘new’ keyword that function acts as a Class.
- Recently JavaScript introduced ‘class’ keyword, but it is not yet adopted by all JavaScript engines.
- Another way of creating a object is using object literals (‘{}’). These objects are considered to be singleton.
- JavaScript supports static methods and variables.
- When ‘new’ keyword is used, new object is created and it assigned as ‘this’ for the duration of call to the constructor function.

Prototypes

- JavaScript functions has a reference to another object called prototype. It is somewhat similar to class definition in other languages.
- It is really another object instance.
- In JavaScript prototype object is used when creating objects, for inheritance and adding methods to a JavaScript class.
- Because of the flexibility in JavaScript there are multiple ways to create classes as well as to extend classes. Prototypes are the recommended way of doing so.
- Function that is being used to create objects is called a constructor function.
- Object instance also has a prototype it is basically the object instance from which object is being created. Object '__proto__' is where object get its properties inherited from.
- Functions prototype is used to inherit properties to object instances.

‘this’ in JavaScript

- Unlikely other languages in JavaScript ‘this’ keyword acts differently.
- Inside a object ‘this’ refers to object itself.
- In global context ‘this’ refers to global object (in browser it is the window object). This behaviour will get changed in strict mode.
- If a function which is using ‘this’ keyword is being passed to another object then ‘this’ will refers to that object, but not to the original object where function was declared at first place.
- This behaviour is very noticeable in callback and closures.

Strict notation

- Restricted mode of JavaScript.
- Purpose it make it easier to write secure JavaScript.
- Strict mode make bad practices in JavaScript to errors.
- Keep developer away from using syntaxes that will get invalidate with future JavaScript developments.
- For example it does not allows to create variables without the var keyword (Variable have to be declared).
- Another example would be it will stop referring to the window object as ‘this’ from outside object instances.

Closure

- JavaScript closure is a function which returns another function.
- In JavaScript closure is used to encapsulate variables into a function and restrict access to it from the outside.
- JavaScript creates an environment with all the local variables from the outer function when the inner function is created. Closure is the combination of this environment and the inner function.

Callback and promises

- JavaScript is asynchronous. All I/O operations in JavaScript is implemented to be asynchronous by nature.
- Reason for this is JavaScript being a single threaded language if an I/O operations holds the thread till it get completed JavaScript won’t perform well as a programming language.
- But asynchronous operation introduce difficulty when we need to do synchronous processing using data.
- This is solved by using callbacks and promises.
- Callback is a function that is being passed to an async task and on completion the function will be executed.
- Promise is an object that is being returned from async tasks. Promise have properties to deal with async operations synchronously.

Callback and promises…

- Nested callbacks passed into sequence of async tasks is referred to a ‘callback hell’.
- Promise object was introduced to solve this problem.
- Promise object has a set of properties, methods and mechanism of chaining to handle complex async tasks nicely.

Version controlling

- What and why?
- Terminology
- Best practices
- GIT

What?

- Managing changes to a source.
- Changes are identified using a revision number.
- Each revision has its timestamp as well as the person who done the change.
- Revisions can be restored, compared and merged.
- “Management of multiple revisions of the same unit of information”

Why?

- Easier backups and centralized source code repository.
- Easy collaborative development.
- Overview of changes performed to a file.
- Access control.
- Conflict resolvement.

Terminology

- Repository
  - Central location where all the files are being kept. Usually a directory with set of files.
- Trunk
  - Also referred to as master branch. This is where the most stable code is being placed which is referred as the production code.
- Stage
  - Mark files for tracking changes.
- Commit
  - Create a snapshot of the changes being made to the files.

Terminology…

- Branch
  - Copy of the master branch taken at a given point. All the feature developments and bug fixes will be done in a branch. Usually it is allowed to have multiple branches at the same time.
- Checkout
  - Mark/unlock file for changing.
- Merge
  - Combining branches together to update the master branch.
- Merge conflict
  - Merge conflicts occur when merging a file which has been changed in two separate branches or places. Changes that interfere other changes.

Best practices

- Use a source control system.
- Always make sure to have the latest version of the file.
  - In distributed source control system advice is to get the latest source code at least start of the day.
- Checkout only what you need.
- Merge code with the development branch at least once per day.
- Always make sure code is working as expected and it is not causing any other code to break.
- Follow a formal review process when merging.

Git

- Most popular version control system.
- Distributed version control system.
  - Client get a complete clone of the source code. In a disaster situation full source along with all history can be restored from a client.
- Free and open source.
- Multiple branches and tags.
  - Feature branches, role branches (production).
- Faster comparing to other systems (works on a linux kernel and written in C).
- Support multiple protocols
  - HTTP, SSH
- Staging area, local commits and stashing.
  - Staging area - Mark files to be committed.
  - Local commit - Commit code locally without pushing into the remote branch.
  - Stashing - Keep file changes in Stash and apply them in a later.

Git commands

- Git init
- Git clone
- Git add
- Git stage
- Git commit
- Git push
- https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html

Git: Interactive Learning

Following are two good interactive demos for learning git.

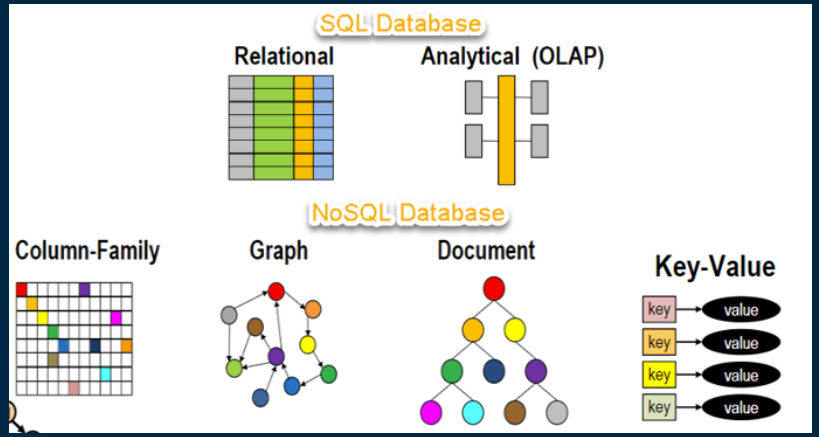
The fundamentals are found in [1] and advanced branching demo is in [2].

[1] https://try.github.io

[2] http://pcottle.github.io/learnGitBranching/

NoSQL

- What?
- Why not SQL?
- Strengths and weaknesses
- MongoDB



What?

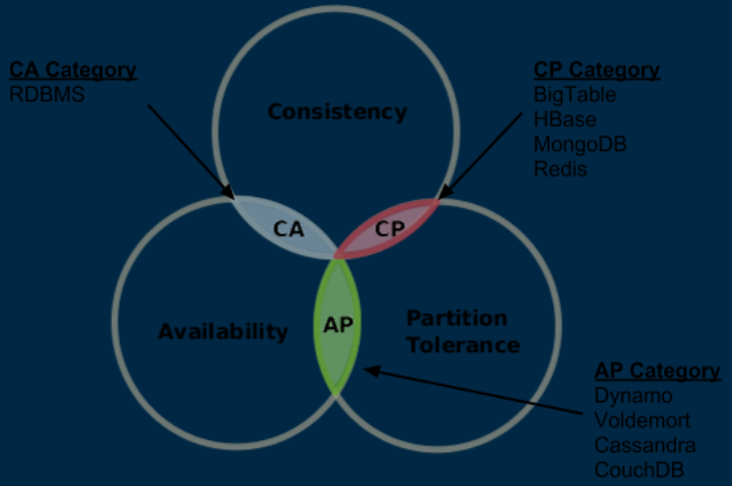
- Non relational (mostly), Schema free.
- Distributed.
- Horizontally scalable.
- Easy replication.
- Eventually consistent.
- Open source (mostly).
- No transaction support.

Why?

- Remove the burden of data structures mismatch between application in-memory and relational databases.
- Integrate databases using services. (ElasticSearch).
- Relational databases not designed to run efficiently on clusters.
- Aggregate oriented databases, are easier to manage inside clusters and based on the domain driven design. (Order details inside the order).
  - But inter aggregate relationships are harder to manage.

CAP theorem

- Consistency - Every read receives the most recent write.
- Availability - Every request receives a response (no guarantee that it is the most recent write).
- Partition tolerance - System will continue to operate despite number of messages lost among network nodes.



Types

- Key-Value - Stores data as key value pairs.
  - Ex: Redis, Riak, Memcached.
- Document - Stores data as documents (JSON, BSON, XML) in maps or collections.
  - Ex: MongoDB
- Column Family - Store data in column families as rows that have many columns associated with.
  - Ex: Cassandra
- Graph - Store entities(nodes) and relationships(edges) between them and represent it in a graph.
  - Ex: Neo4j

MongoDB

- NoSQL document database.
- Strong query capabilities with aggregations using JavaScript.
- Use SpiderMonkey JavaScript engine.
- High availability with replica sets.
- Reads and writes on primary by default.
- Eventually consistent on secondary instances.
- In built file storage called Grid File System.

MongoDB queries

- Insert
- Find
- Update
- Remove

Any Questions?