Imp links

<https://docs.mongodb.com/manual/reference/method/js-collection/>

<https://docs.mongodb.com/manual/reference/method/js-collection/>

<http://www.allinterview.com/>

1. What is node

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

1. Why the Name of node is node

Node is a point in a network or diagram at which lines or pathways intersect. Node is a single-threaded, single-process system which enforces shared-nothing design with OS process boundaries. It has rather good libraries for networking. I believe this to be a basis for designing very large distributed programs. The “nodes” need to be organized: given a communication protocol, told how to connect to each other. In the next couple months we are working on libraries for Node that allow these networks.

1. Difference between node and java

|  |  |
| --- | --- |
| Java | Node |
| Synchonous | Async |
| Use multithreading | Single Threaded |
| Process based | Event based |
| Cpu intensive applications can be made with java | Cpu intensive applications can not be made |
| slow | node is faster than java |

1. How node is single threaded

Node works on single thread it achieves the fast performance with single thread with the use of eventloop and callbacks

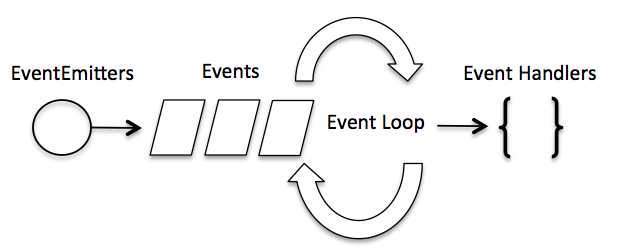
and event loop keeps on eye for any incoming request

as soon as new request is coming it throw the perticular event

and keep it in queue for processing.

Node.js uses events heavily and it is also one of the reasons why Node.js is pretty fast compared to other similar technologies. As soon as Node starts its server, it simply initiates its variables, delcares functions and then simply waits for event to occur.

In an event-driven application, there is generally a main loop that listens for events, and then triggers a callback function when one of those events is detected.



1. How node is fast with single thread

Because of event loop and callbacks

1. How node works(Threads concept)

Node JS Platform does not follow Request/Response Multi-Threaded Stateless Model. It follows Single Threaded with Event Loop Model. As Node JS follows this architecture, it can handle more and more concurrent client requests very easily.

**Single Threaded Event Loop Model Processing Steps**:

* Clients Send request to Web Server.
* Node JS Web Server internally maintains a Limited Thread pool to provide services to the Client Requests.
* Node JS Web Server receives those requests and places them into a Queue. It is known as “Event Queue”.
* Node JS Web Server internally has a Component, known as “Event Loop”. Why it got this name is that it uses indefinite loop to receive requests and process them. (See some Java Pseudo code to understand this below).
* Event Loop uses Single Thread only. It is main heart of Node JS Platform Processing Model.
* Even Loop checks any Client Request is placed in Event Queue. If no, then wait for incoming requests for indefinitely.
* If yes, then pick up one Client Request from Event Queue
  1. Starts process that Client Request
  2. If that Client Request Does Not requires any Blocking IO Operations, then process everything, prepare response and send it back to client.
  3. If that Client Request requires some Blocking IO Operations like interacting with Database, File System, External Services then it will follow different approach
     + Checks Threads availability from Internal Thread Pool
     + Picks up one Thread and assign this Client Request to that thread.
     + That Thread is responsible for taking that request, process it, perform Blocking IO operations, prepare response and send it back to the Event Loop
     + Event Loop in turn, sends that Response to the respective Client.

1. What is event loop

https://docs.mongodb.com/manual/reference/method/js-collection/

An event loop is an internal loop in that processes system and user events

To process and handle external events and to convert them into callback invocations an event loop is used. So, at I/O calls, node.js can switch from one request to another.

Node js is a single threaded application but it support concurrency via concept of event and callbacks. As every API of Node js are asynchronous and being a single thread, it uses async function calls to maintain the concurrency. Node uses observer pattern. Node thread keeps an event loop and whenever any task get completed, it fires the corresponding event which signals the event listener function to get executed.

1. What is loopback

LoopBack is a framework for [creating APIs](https://www.toptal.com/back-end/job-description) and connecting them with backend data sources. Built on top of Express, it can take a data model definition and easily generate a fully functional end-to-end REST API that can be called by any client.

LoopBack comes with a built-in client, API Explorer. We’ll use this since it makes it easier to see the results of our work, and so that our example can focus on building the API itself.

1. What is callback

Callback is an asynchronous equivalent for a function. A callback function is called at the completion of a given task. Node makes heavy use of callbacks. All APIs of Node are written is such a way that they supports callbacks. For example, a function to read a file may start reading file and return the control to execution environment immediately so that next instruction can be executed. Once file I/O is complete, it will call the callback function while passing the callback function, the content of the file as parameter. So there is no blocking or wait for File I/O. This makes Node.js highly scalable, as it can process high number of request without waiting for any function to return result.

1. What is callback hell

When there are so many Callback functions inside callback that becomes callback hell. This is not a good coding as code becomes unreachable due to this.

1. How call back hell is removed from node

Using promises

Using modular approach

Using generators

* Don't nest functions. Give them names and place them at the top level of your program
* Handle **every single error** in every one of your callbacks. Use a linter like [standard](http://standardjs.com/) to help you with this.

**Modularization:** Create reusable functions and place them in a module to reduce the cognitive load required to understand your code. It’s like splitting your code into small pieces

1. What is promises

Promises are used to avoid callback hell

A Promise object represents a value that may not be available yet, but will be resolved at some point in the future. It allows you to write asynchronous code in a more synchronous fashion. For example, if you use the promise API to make an asynchronous call to a remote web service you will create a Promise object which represents the data that will be returned by the web service in future. The caveat being that the actual data is not available yet. It will become available when the request completes and a response comes back from the web service. In the meantime the Promise object acts like a proxy to the actual data. Furthermore, you can attach callbacks to the Promise object which will be called once the actual data is available.

If everything is successful, the **promise** is fulfilled by calling resolve() . In case of an error, reject() is called with an Error object. This indicates that the **promise** is rejected.

## The API

To get started, let’s examine the following code which creates a new Promise object.

if (window.Promise) { // Check if the browser supports Promises

var promise = new Promise(function(resolve, reject) {

//asynchronous code goes here

});

}

We start by instantiating a new Promise object and passing it a callback function. The callback takes two arguments,resolve and reject, which are both functions. All your asynchronous code goes inside that callback. If everything is successful, the promise is fulfilled by calling resolve(). In case of an error, reject() is called with an Error object. This indicates that the promise is rejected.

Now let’s build something simple which shows how promises are used. The following code makes an asynchronous request to a web service that returns a random joke in JSON format. Let’s examine how promises are used here.

if (window.Promise) {

console.log('Promise found');

var promise = new Promise(function(resolve, reject) {

var request = new XMLHttpRequest();

request.open('GET', 'http://api.icndb.com/jokes/random');

request.onload = function() {

if (request.status == 200) {

resolve(request.response); // we got data here, so resolve the Promise

} else {

reject(Error(request.statusText)); // status is not 200 OK, so reject

}

};

request.onerror = function() {

reject(Error('Error fetching data.')); // error occurred, reject the Promise

};

request.send(); //send the request

});

console.log('Asynchronous request made.');

promise.then(function(data) {

console.log('Got data! Promise fulfilled.');

document.getElementsByTagName('body')[0].textContent = JSON.parse(data).value.joke;

}, function(error) {

console.log('Promise rejected.');

console.log(error.message);

});

} else {

console.log('Promise not available');

}

Another Example

function dieToss() {

return Math.floor(Math.random() \* 6) + 1;

}

console.log('1');

var promise = new RSVP.Promise(function(fulfill, reject) {

var n = dieToss();

if (n === 6) {

fulfill(n);

} else {

reject(n);

}

console.log('2');

});

promise.then(function(toss) {

console.log('Yay, threw a ' + toss + '.');

}, function(toss) {

console.log('Oh, noes, threw a ' + toss + '.');

});

console.log('3');

This snippet prints output similar to the following:

1

2

3

Oh, noes, threw a 4.

Or, if we get lucky, we see:

1

2

3

Yay, threw a 6.

In the previous code, the Promise constructor callback contains the asynchronous code used to get data the from remote service. Here, we just create an Ajax request to <http://api.icndb.com/jokes/random> which returns a random joke. When a JSON response is received from the remote server, it is passed to resolve(). In case of any error, reject() is called with an Error object.

When we instantiate a Promise object we get a proxy to the data that will be available in future. In our case we are expecting some data to be returned from the remote service at some point in future. So, how do we know when the data becomes available? This is where the Promise.then() function is used. This function takes two arguments: a success callback and a failure callback. These callbacks are called when the Promise is settled (i.e. either fulfilled or rejected). If the promise was fulfilled, the success callback will be fired with the actual data you passed to resolve(). If the promise was rejected, the failure callback will be called. Whatever you passed to reject() will be passed as an argument to this callback.

Try this [Plunkr](http://plnkr.co/edit/ilf9xtDqrimWxZd77yLI?p=preview) example. Simply refresh the page to view a new random joke. Also, open up your browser console so that you can see the order in which the different parts of the code are executed. Note that a promise can have three states:

* pending (not fulfilled or rejected)
* fulfilled
* rejected

The Promise.status property, which is code-inaccessible and private, gives information about these states. Once a promise is rejected or fulfilled, this status gets permanently associated with it. This means a promise can succeed or fail only once. If the promise has already been fulfilled and later you attach a then() to it with two callbacks, the success callback will be correctly called. So, in the world of promises, we are not interested in knowing when the promise is settled. We are only concerned with the final outcome of the promise.

## Chaining Promises

It is sometimes desirable to chain promises together. For instance, you might have multiple asynchronous operations to be performed. When one operation gives you data, you will start doing some other operation on that piece of data and so on. Promises can be chained together as demonstrated in the following example.

function getPromise(url) {

// return a Promise here

// send an async request to the url as a part of promise

// after getting the result, resolve the promise with it

}

var promise = getPromise('some url here');

promise.then(function(result) {

//we have our result here

return getPromise(result); //return a promise here again

}).then(function(result) {

//handle the final result

});

The tricky part is that when you return a simple value inside then(), the next then() is called with that return value. But if you return a promise inside then(), the next then() waits on it and gets called when that promise is settled.

## Handling Errors

You already know the then() function takes two callbacks as arguments. The second one will be called if the promise was rejected. But, we also have a catch() function which can be used to handle promise rejection. Have a look at the following code:

promise.then(function(result) {

console.log('Got data!', result);

}).catch(function(error) {

console.log('Error occurred!', error);

});

This is equivalent to:

promise.then(function(result) {

console.log('Got data!', result);

}).then(undefined, function(error) {

console.log('Error occurred!', error);

});

Note that if the promise was rejected and then() does not have a failure callback, the control will move forward to the next then() with a failure callback or the next catch(). Apart from explicit promise rejection, catch() is also called when any exception is thrown from the Promise constructor callback. So, you can also use catch() for logging purposes. Note that we could use try...catch to handle errors, but that is not necessary with promises as any asynchronous or synchronous error is always caught by catch().

As the [Promise.prototype.then()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise/then) and [Promise.prototype.catch()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise/catch) methods return promises, they can be chained—an operation called composition.

1. What is singleton

A Singleton is an object which can only be instantiated one time. Repeated calls to its constructor return the same instance and in this way one can ensure that they don't accidentally create, say, two Users in a single User application

The **Singleton** Pattern limits the number of instances of a particular object to just one. **This single instance is called the singleton**. **Singletons** are useful in situations where system-wide actions need to be coordinated from a single central place. An example is a database connection pool

1. What is opensource

Its **source** code is available with a license (It’s generally free) in which the copyright holder provides the rights to study, change, and distribute the **software** to anyone and for any purpose. Meaning that other developers can see how it works and add to it. Examples of open source products include Open Office, the internet browser [Mozilla Firefox](http://en.wikipedia.org/wiki/Mozilla_Firefox), Wikipedia, the [GNU/Linux](http://en.wikipedia.org/wiki/Linux) operating system and its derivative [Android](http://en.wikipedia.org/wiki/Android_(operating_system)), an operating system for mobile devices .the idea behind open source software is that users are effectively co-developers, suggesting ways to improve it and helping to hunt out bugs and problems. This means that if you wish, you can modify it to your own needs, port it to new operating systems and share it with others.

1. What is events

Much of the Node.js core API is built around an idiomatic asynchronous event-driven architecture in which certain kinds of objects (called "emitters") periodically emit named events that cause Function objects ("listeners") to be called. All objects that emit events are instances of the EventEmitter class. These objects expose an eventEmitter.on() function that allows one or more functions to be attached to named events emitted by the object. Typically, event names are camel-cased strings but any valid JavaScript property key can be used.

When the EventEmitter object emits an event, all of the functions attached to that specific event are called *synchronously*. Any values returned by the called listeners are *ignored* and will be discarded.

1. What is eventdriven programming, event emitter class

In computer **programming**, **event**-**driven programming** is a **programming**paradigm in which the flow of the **program** is determined by **events** such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs/threads.

 EventEmitter class lies in **events** module. It is accessibly via following syntax:

// Import events module

var events = require('events');

// Create an eventEmitter object

var eventEmitter = new events.EventEmitter();

When an EventEmitter instance faces any error, it emits an 'error' event. When new listener is added, 'newListener' event is fired and when a listener is removed, 'removeListener' event is fired.

EventEmitter provides multiple properties like **on** and **emit**. **on** property is used to bind a function with the event and **emit** is used to fire an event.

1. How node is different from other frameworks

here is two important things that make Node.js different to existing server-side frameworks,**asynchronous events** and the use of **JavaScript** as a programming language.

## Asynchronous Events

While most of the existing server side frameworks use a synchronous architecture, Node.js uses an**asynchronous** architecture, which JavaScript can handle well. This means that the server **reacts to events** and sends events (messages) to e.g. the database. This style of programming is very different to a synchronous style, and may be hard to use with other languages. Node.js employs an asynchronous style with asynchronous IO and can scale well.

## JavaScript

JavaScript is the programming language that web applications are using on the client. Using the same language on the server-side means that the developer can apply his JavaScript knowledge both on the client and the server, and use the same functions as needed.

1. Advantages of node

Following are some of the important features that make Node.js the first choice of software architects.

* **Asynchronous and Event Driven** All APIs of Node.js library are asynchronous that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
* **Very Fast** Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
* **Single Threaded but Highly Scalable** - Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
* **No Buffering** - Node.js applications never buffer any data. These applications simply output the data in chunks.
* **License** - Node.js is

1. Applications of node

Following are the areas where Node.js is proving itself as a perfect technology partner.

* I/O bound Applications
* Data Streaming Applications
* Data Intensive Real time Applications (DIRT)
* JSON APIs based Applications
* Single Page Applications

It is not advisable to use Node.js for CPU intensive applications.

1. What is the use of libuv libraray

**libuv** is a [software library](https://en.wikipedia.org/wiki/Software_library) that provides asynchronous event notification.

Node achieve asynchronous even loop programming paradigm on the basis of libuv libraray

Event loop works behind the scene on libuv library**.**

1. Why to use sudo in npm install

Its provides the master priveliges.

1. What is JSON

JavaScript Object Notation

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate

1. why async.waterfall and how event loop works with async function.
2. What is asynchronous

Asynchronous means non blocking.

Except your code everything will run in parallel.

If code is requesting i/o it will not block the code the rest code will run.

This is achieved with events and callbacks in node .

1. What is javascript

JavaScript is the programming language of HTML and the Web.

1. What is closures in js

A **closure** is an inner function that has access to the outer (enclosing) function's variables—scope chain. The **closure** has three scope chains: it has access to its own scope (variables defined between its curly brackets), it has access to the outer function's variables, and it has access to the global variables.

You create a closure by adding a function inside another function.  
**A Basic Example of Closures in JavaScript:**

function showName (firstName, lastName) {

var nameIntro = "Your name is ";

// this inner function has access to the outer function's variables, including the parameter

function makeFullName () {

return nameIntro + firstName + " " + lastName;

}

return makeFullName ();

}

showName ("Michael", "Jackson"); // Your name is Michael Jackson

Closures are used extensively in Node.js; they are workhorses in Node.js’ asynchronous, non-blocking architecture. Closures are also frequently used in jQuery and just about every piece of JavaScript code you read.

1. What is rest

**RESTful** Web Services are REST architecture based web services. In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web based applications.

1. What is webservices

A **web service** is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a **web service**. For example, a client invokes a **web service** by sending an XML message, then waits for a corresponding XML response

1. Types of webservices

Two types

SOAP Based

Restful

1. Difference between soap and rest

REST

RESTs sweet spot is when you are exposing a public API over the internet to handle CRUD operations on data. REST is focused on accessing named resources through a single consistent interface.

SOAP

SOAP brings it’s own protocol and focuses on exposing pieces of application logic (not data) as services. SOAP exposes operations. SOAP is focused on accessing named operations, each implement some business logic through different interfaces.

Though SOAP is commonly referred to as “web services” this is a misnomer. SOAP has very little if anything to do with the Web. REST provides true “Web services” based on URIs and HTTP.

1. Difference between put and post

PUT puts a file or resource at a specific URI, and exactly at that URI. If there's already a file or resource at that URI, PUT replaces that file or resource. If there is no file or resource there, PUT creates one

The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line.

The actual function performed by the POST method is determined by the server and is usually dependent on the Request-URI. The posted entity is subordinate to that URI in the same way that a file is subordinate to a directory containing it, a news article is subordinate to a newsgroup to which it is posted, or a record is subordinate to a database.

1. Difference between put and patch

PUT = replace the ENTIRE RESOURCE with the new representation provided (no mention of related resources in the spec from what i can see)

PATCH = replace parts of the source resource with the values provided AND|OR other parts of the resource are updated that you havent provided (timestamps) AND|OR updating the resource effects other resources (relationships)

1. Difference between get and post

|  |  |
| --- | --- |
| GET | POST |
| Requests data from a specified resource | Submits data to be processed to a specified resource |
| Get request can be bookmarked | Post request can't be bookmarked |
| Parameters remains in browser history | Parameters don't remains in browser history |
| GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information! | POST is a little safer than GET because the parameters are not stored in browser history or in web server logs |
| Data is visible to everyone in the URL | Data is not displayed in the UR |
| when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters) | No restrictions |

1. What is opensource

Its **source** code is available with a license (It’s generally free) in which the copyright holder provides the rights to study, change, and distribute the **software** to anyone and for any purpose. Meaning that other developers can see how it works and add to it. Examples of open source products include Open Office, the internet browser [Mozilla Firefox](http://en.wikipedia.org/wiki/Mozilla_Firefox), Wikipedia, the [GNU/Linux](http://en.wikipedia.org/wiki/Linux) operating system and its derivative [Android](http://en.wikipedia.org/wiki/Android_(operating_system)), an operating system for mobile devices .the idea behind open source software is that users are effectively co-developers, suggesting ways to improve it and helping to hunt out bugs and problems. This means that if you wish, you can modify it to your own needs, port it to new operating systems and share it with others.

1. What is shrinkwrap

**NPM** itself has a feature called "**shrinkwrap**" that. locks down the versions of a package's dependencies so that you can control exactly which versions of each dependency will be used when your package is installed.

When you run npm shrinkwrap in a project after run­ning npm install, it cre­ates a file called npm-shrinkwrap.json which lists the exact pack­age ver­sions of all the installed pack­ages in the entire hier­ar­chy.

1. Types of http method

Get

Put

Post

Patch

Delete

option

1. What is option method in http

Use by client to find out the methods supported by the web server

1. Types of async

Parallel

Series

Auto

1. What is async.auto

In Async.auto functions execute in manner they are written and value will be stored in the corresponding variable assigned to statements.

For eg:

async.auto({

initialTask: function(callback) {

//Do some operations

callback(null, name, initialModels);

},

task1: ['initialTask', function(callback, results) {

var models = results.initialTask[1];

//Add some more data to models

callback(null, models);

}],

task2: ['initialTask', function(callback, results) {

var models = results.initialTask[1];

//Add some more data to models

callback(null, models);

}],

task3: ['initialTask', function(callback, results) {

var models = results.initialTask[1];

//Add some more data to models

callback(null, models);

}],

finalTask: ['task1', 'task2', 'task3', function(callback, results) {

//Here the followings are the same: results.initialTask[1], results.task1[0], results.task2[0], results.task3[0]

}]

});

Basically async.auto will execute all the functions in the order it deems necessary. So in this case initialTask will be called first. Then task1, task2, and task3 will be called in parallel. Finally finalTask will be called with the results. The reason all the values are the same is related to JavaScript's [call-by-sharing](http://en.wikipedia.org/wiki/Evaluation_strategy#Call_by_sharing), meaning if you change a function parameter itself, then it won't affect the item that was fed into the parameter. If you change the internals of the parameter, it will carry up to the item.

1. why we use body-parser

body-parser extracts the entire body portion of an incoming request stream and exposes it on req.body

$ npm install body-parser

API: var **express =** require('express')

var bodyParser **=** require('body-parser')

var app **=** express()

*// parse application/json*

app.use(bodyParser.**json**())

app.use(function (req, res) {

  res.setHeader('Content-Type', 'text/plain')

  res.write('you posted:\n')

var username=req.body.name;

  res.end(JSON.stringify(**req.body**, null, 2))

)}

1. why to use multer

Multer is a node.js middleware for handling multipart/form-data.

It is written on top of [busboy](https://github.com/mscdex/busboy) for maximum efficiency.

**NOTE**: Multer will not process any form which is not multipart (multipart/form-data).

npm install multer

var express **=** require('express')

var multer  **=** require('multer')

var upload **=** multer({ dest**:** 'uploads/' })

var app **=** express()

app.post('/profile', upload.single('avatar'), function (req, res, next) {

*// req.file is the `avatar` file*

*// req.body will hold the text fields, if there were any*

})

1. what is procedures in mysql

Set of precompiles statements

1. what is triggers

A **trigger** is a special kind of stored procedure that automatically executes when an event occurs in the database server. DML **triggers** execute when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

1. how to get the second least salary from database

Select min(sal) from emp where sal>(select min(sal) from emp);

1. how to delete the value from the array which is modulus by 3

var a = [ 10, 9, 3, 2, 5 ];

for (var i = 0; i < a.length; i++) {

if(i===3){

delete a[i];

}

}

var b = [ 10, 9, 3, 2, 5 ];

b.splice(3,1); //Array.splice(index,length)

console.log('a is ',a,'and a[3] is ',typeof a[3]);

console.log('b is ',b);

1. how to add a property in json object at second position

Using Splice method. If you delete an 0 item with splice method and add a json it will work

Eg.

var a= [1,2,3,4,5,6,7];

//delete a[2];

console.log('-------'+a);

a.splice(3,1);

console.log('@@@@@@@@'+a);

a.splice(2,0,'we');

console.log('########'+a);

1. ECMA script 6

**ECMAScript** (or ES) is a trademarked scripting-language specification standardized by Ecma International in ECMA-262 and ISO/IEC 16262. It was based on JavaScript, which now tracks **ECMAScript**. It is commonly used for client-side scripting on the World Wide Web.

**ECMAScript 6**, also known as **ECMAScript** 2015, is the latest version of the**ECMAScript** standard.

1. How to delete the array item and release the space too

Using the splice method specific array item can be deleted as well space will be released

Var items=[‘cat’,’dog’,rat]

Items.splice(1,1)

First parameter will take index number and second will take the value how many elements you want to delete after that index(including index)

1. What is exception

The term exception is shorthand for the phrase "exceptional event."

Definition: An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.

When an error occurs within a method, the method creates an object and hands it off to the runtime system. The object, called an exception object, contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called throwing an exception.

After a method throws an exception, the runtime system attempts to find something to handle it. The set of possible "somethings" to handle the exception is the ordered list of methods that had been called to get to the method where the error occurred.

1. How exceptions are handled in node js

Exceptions are handled in node js with try and catch block

1. Threshold in node and java
2. Connection pooling

In software engineering, a connection pool is a cache of database connections maintained so that the connections can be reused when future requests to the database are required. [citation needed] Connection pools are used to enhance the performance of executing commands on a database.

1. What are joins

A SQL join is a Structured Query Language (SQL) instruction to combine data from two sets of data (e.g. two tables). An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them. The most common type of join is: SQL INNER JOIN(simple join). An SQL INNER JOIN returns all rows from multiple tables where the join condition is met

INNER JOIN: Returns all rows when there is at least one match in BOTH tables

LEFT JOIN: Return all rows from the left table, and the matched rows from the right table

RIGHT JOIN: Return all rows from the right table, and the matched rows from the left table

FULL JOIN: Return all rows when there is a match in ONE of the tables

1. What is mongodb

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document. Database Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

1. How to add a document in mongodb

Db.collectionName.insert(values)

1. How to add a unique entry in mongodb

Using update command you can insert a unique entry

First time if entry doesn’t exist it will add otherwise it will update it

1. How to create a database in mongodb

Use database\_name

1. How to create a collection in mongodb

Db.createCollection(“collectionName”)

1. How to drop a collection and database in mongodb

Db.dropDatabase()

Db.collectionName.drop()

1. How to display all the entries of a collection

Db.collectionName.find()

1. How to show all the databases and collectons in mongodb

Show dbs

It will display al the databases

Show collections

It will display all the collections in the current db.

1. Why pretty() method used in mongodb

To display the data in proper format.

1. How to add multiple entries in one go in mongodb

You can add multiple entries by sending the rows in array.

For eg. >db.post.insert([

{

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by: 'tutorials point',

url: 'http://www.tutorialspoint.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

},

{

title: 'NoSQL Database',

description: 'NoSQL database doesn't have tables',

by: 'tutorials point',

url: 'http://www.tutorialspoint.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 20,

comments: [

{

user:'user1',

message: 'My first comment',

dateCreated: new Date(2013,11,10,2,35),

like: 0

}

]

}

])

1. Difference between mongodb and rdbms

MySQL is a popular open-source relational database management system (RDBMS)MySQL stores data in tables and uses structured query language (SQL) for database access.

MongoDB is an open-source database.MongoDB stores data in JSON-like documents that can vary in structure. Related information is stored together for fast query access through the MongoDB query language. MongoDB uses dynamic schemas, meaning that you can create records without first defining the structure, such as the fields or the types of their values. You can change the structure of records (which we call documents) simply by adding new fields or deleting existing ones

1. What is type of null in js

Type of null in javascript is object and type of undefined is undefined

1. What is hibernate

Hibernate is an Object-Relational Mapping(ORM) solution for JAVA and it raised as an open source persistent framework created by Gavin King in 2001.

Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieve the developer from 95% of common data persistence related programming tasks.Hibernate sits between traditional Java objects and database server to handle all the work in persisting those objects based on the appropriate O/R mechanisms and patterns. it Provides simple APIs for storing and retrieving Java objects directly to and from the database

If there is change in Database or in any table then the only need to change XML file properties.Abstract away the unfamiliar SQL types and provide us to work around familiar Java Objects.



1. What is sessionfactory

Configuration object is used to create a SessionFactory object which inturn configures Hibernate for the application using the supplied configuration file and allows for a Session object to be instantiated. The SessionFactory is a thread safe object and used by all the threads of an application

1. **Types of datatypes supported by mongodb**

String: This is the most commonly used datatype to store the data. String in

MongoDB must be UTF-8 valid.

 Integer: This type is used to store a numerical value. Integer can be 32 bit or 64

bit depending upon your server.

 Boolean: This type is used to store a boolean (true/ false) value.

 Double: This type is used to store floating point values.

 Min/Max Keys: This type is used to compare a value against the lowest and

highest BSON elements.

 Arrays: This type is used to store arrays or list or multiple values into one key.

 Timestamp: ctimestamp. This can be handy for recording when a document has

been modified or added.

 Object: This datatype is used for embedded documents.

 Null: This type is used to store a Null value.

 Symbol: This datatype is used identically to a string; however, it's generally

reserved for languages that use a specific symbol type.

 Date: This datatype is used to store the current date or time in UNIX time format.

You can specify your own date time by creating object of Date and passing day,

month, year into it.

 Object ID: This datatype is used to store the document’s ID.

 Binary data: This datatype is used to store binary data.

 Code: This datatype is used to store JavaScript code into the document.

 Regular expression: This datatype is used to store regular expression.

1. **Difference between JAX-RS and JAX-ws**

REST based architectures typically will use a lightweight data format, like JSON, to send data back and forth. This is in contrast to JAX-WS which uses XML.

JAX-WS represents SOAP,JAX-RS represents REST

JAX-WS - is Java API for the XML-Based Web Services - a standard way to develop a Web- Services in SOAP notation (Simple Object Access Protocol).

Calling of the Web Services is performed via remote procedure calls. For the exchange of information between the client and the Web Service is used SOAP protocol. Message exchange between the client and the server performed through XML- based SOAP messages.

Clients of the JAX-WS Web- Service need a WSDL file to generate executable code that the clients can use to call Web- Service.

JAX-RS - Java API for RESTful Web Services. RESTful Web Services are represented as resources and can be identified by Uniform Resource Identifiers (URI). Remote procedure call in this case is represented a HTTP- request and the necessary data is passed as parameters of the query. Web Services RESTful - more flexible, can use several different MIME- types. Typically used for XML data exchange or JSON (JavaScript Object Notation) data exchange..

1. Why use MongoDB instead of MySQL?

Organizations of all sizes are adopting MongoDB because it enables them to build applications faster, handle highly diverse data types, and manage applications more efficiently at scale.

Development is simplified as MongoDB documents map naturally to modern, object-oriented programming languages. Using MongoDB removes the complex object-relational mapping (ORM) layer that translates objects in code to relational tables.

MongoDB’s flexible data model also means that your database schema can evolve with business requirements. For example, the ALTER TABLE command required to add a single, new field to [Craiglist’s MySQL database](https://www.mongodb.com/presentations/mongodb-craigslist-one-year-later) would take months to execute. The Craigslist team migrated to MongoDB because it can accommodate changes to the data model without such costly schema migrations.

MongoDB can also be scaled within and across multiple distributed data centers, providing new levels of availability and scalability previously unachievable with relational databases like MySQL. As your deployments grow in terms of data volume and throughput, MongoDB scales easily with no downtime, and without changing your application. In contrast, to achieve scale with MySQL often requires significant, custom engineering work.

|  |  |
| --- | --- |
| MongoDB | MySQL |
| MongoDB is an open-source database MongoDB stores data in JSON-like documents that can vary in structure. | MySQL is a popular open-source relational database management system (RDBMS)MySQL stores data in tables and uses structured query language (SQL) for database access |
| Collection | Table |
| Document | Row |
| Field | Column |
| Embedded documents, linking | Joins |

1. How mongodb works

mongoDB works on JSON objects, i.e javascript object notation.

Ebery thing store in s collection(table) in JSON format.

1. How joins are achieved in mongo

MongoDB don’t supports joins till now.It can be achieved through storing the id’s in different tables

1. what are views?

An output of query can be stored as a **view**. **View** act like small a table but it does not physically take any space. **View** is good way to present data in particular users from accessing the table directly. A **view in oracle** is nothing but a stored sql scripts

1. types of views

The views are Mainly 2 types .  
1-> SIMPLE VIEW   
2-> COMPLEX VIEW  
  
Simple view:   
- It is created by selecting only one table.  
- It does not contains functions.  
- it can perform DML (SELECT,INSERT,UPDATE,DELETE,MERGE, CALL,LOCK TABLE) operations through simple view.  
Complex view :  
  
-It is created by selecting more than one table.  
-It can performs functions.  
-You can not perform always DML operations through Complex views.