PART 1

Question A

Using simple code, explain what kind of situations would you use the methods:

i. GET ii. POST iii. UPDATE iv. PUT

answers:

1. GET

* GET method can be cached
* GET method remain in the browser history
* GET method can be bookmarked
* GET method should never be used when dealing with sensitive data
* GET method have length restrictions
* GET method are only used to request data

this.getAllFacility = function (facilityCode) {  
 var deferred = $q.defer();  
 $http.get('http://10.4.104.59/miportal-app/o/miportal-rest/api/organizations')  
 .then(function (data, status, headers, config) {  
 deferred.resolve(data.data);  
 },function (data, status, headers, config) {  
 deferred.reject(status);  
 });

1. POST

* POST method are never cached
* POST method do not remain in the browser history
* POST method cannot be bookmarked
* POST method have no restrictions on data length

function createFormTreatmentPricing(data) {  
 var deferred = $q.defer();  
 // NOTE: make sure to stringify POST data  
 $http.post(config.backendApiMhtis + apiTreatmentPricingEntreeURL , JSON.stringify(data)).then(function(data) {  
 deferred.resolve(data);  
 },  
 function (data) {  
 deferred.reject(data);  
 });  
 return deferred.promise;  
}

3)Put

* PUT is used to send data to a server to create/update a resource.
* The difference between POST and PUT is that PUT requests are idempotent. That is, calling the same PUT request multiple times will always produce the same result. In contrast, calling a POST request repeatedly have side effects of creating the same resource multiple times.

function updateFormTreatmentPricing(data) {  
 var deferred = $q.defer();  
 // NOTE: make sure to stringify POST data  
 $http.put(config.backendApiMhtis + apiTreatmentPricingEntreeURL , JSON.stringify(data)).then(function(data) {  
 deferred.resolve(data);  
 },  
 function (data) {  
 deferred.reject(data);  
 });  
 return deferred.promise;  
}

Question B

**Tokens**

*Tokens is provided by JWT* (JSON Web Token). Nowadays JWT (JSON Web Token) is everywhere.

JWT consists of three parts:

* Header, containing the type of the token and the hashing algorithm
* Payload, containing the claims
* Signature, which can be calculated as follows if you chose HMAC SHA256: HMACSHA256( base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)
* If you have to support a web application only, either cookies or tokens are fine - for cookies think about XSRF, for JWT take care of XSS.
* If you have to support both a web application and a mobile client, go with an API that supports token-based authentication.

Adding JWT to Koa applications is only a couple of lines of code:

|  |  |
| --- | --- |
|  | var koa = require('koa'); |
|  | var jwt = require('koa-jwt'); |
|  |  |
|  | var app = koa(); |
|  |  |
|  | app.use(jwt({ |
|  | secret: 'very-secret' |
|  | })); |
|  |  |
|  | // Protected middleware |
|  | app.use(function \*(){ |
|  | // content of the token will be available on this.state.user |
|  | this.body = { |
|  | secret: '42' |
|  | }; |
|  | });  Question C  **What kind of format is best to be returned by an API as a response? Explain your answer and provide comparisons, if needed.**  **Answers:**  Json format  [JSON](http://en.wikipedia.org/wiki/JSON) is short for JavaScript Object Notation, and is a way to store information in an organized, easy-to-access manner |

As a simple example, information about me might be written in JSON as follows:

|  |  |
| --- | --- |
|  | var jason = { |
|  | "age" : "24", |
|  | "hometown" : "Missoula, MT", |
|  | "gender" : "male" |
|  | };  Benefit of using Json   * Since the JSON format is text only, it can easily be sent to and from a server, and used as a data format by any programming language. * it’s becoming more and more important for sites to be able to load data quickly and asynchronously without delaying page rendering * we can convert any JavaScript object into JSON, and send JSON to the server * We can also convert any JSON received from the server into JavaScript objects * work with the data as JavaScript objects, with no complicated parsing and translations |

Drawback of using Json

* Its limit in terms of the number of data types defined by default

**Comparison JSON and SOAP**

|  |  |
| --- | --- |
| **JSON** | **Soap** |
| Its an object | It’s a protocol |
| Cannot communicate to server or browsers by itself | It has ability to communicate to both browser and servers |
| It maintains a format a key value pair | Maintain XML format |
| Tis contain message only | Its contain envelop,header,body,faults etc |
| It lacks the ability as it is just an object | Has the ability to connect to client application in remote location |
| JSON cannot use SOAP | Soap can used JSON |
| Lighter than SOAP | Heaver as compared to JSON |