**Frontend**

1. Type domain name: browser send a message to Internet Service Provider (ISP), ISP relays the message to DNS server, then it looks up in its database to find exact IP address of the one you type. The IP address will be sent back to browser. Then you can send direct request to that IP address through ISP. This request can be delivered via Internet Backbone.

2. Website: (1) HTML: structure of the website; (2) CSS: styling the website; (3) JS: allow website to do things.

3. Javascript: interpreted language, slow, execute all instructions line by line. It is one language that is supported by all of the major web browsers; Java: compiled language, fast.

4. JS syntax: variable-var; define function: function test(b) {var a = 1; return a + b;}, because JS is script language, its syntax is similar to Python; print: console.log(a);string length: s.length; slicing: s.slice(0, 1), similar to Python slicing; casting: s.toUpperCase(); decimal division is not rounded; random number: var n = Math.random(); logic operator: ===, !==, triple equal sign checks data type while bi-equal sign does not check; control statement: if (a === b) { c = a + b;}

5. syntax 2: var a = [b,c,d,e];array size: a.length; find out whether an element is in an array: a.includes(b), return boolean; add element to the array: a.push(b); loop: while(){}, for(i=0;i<2;i++){}

6. document object model (DOM): document.getElementById()/getElementByClassName(), document.querySelector(), document.querySelectorAll().

7. object, constructor function: function object(a, b,c) {this.a = b; this.b = b; this.c = c;}

8. define function in an object: var object = {move: function() {…}}

9. jQuery is a library to be used in web development, which simplifies the code in JS. jQuery should be placed where before JS in the body tag.

**Backend**

1. Why Node.js? allow us to build the backend using JS. It allows us to take JS out of the browser and liberates it, allowing it to interact directly, interacting directly with the hardware of the computer.

2. NPM-node package manager

3. Express: Node framework. Similar to jQuery to JS. Simplify the codes in Node

4. nodemon: automatically start and close server based on the change in your JS code.

5. API: (1) endpoint: each API interact with an external system, will have an endpoint; (2) paths: in order to narrow down on a specific piece of data you want from the external server; (3) parameters; (4) authentication: monetize the server’s data or limit your use to a threshold. Each time you use the API, they have to identify you as a developer and keep track how often you are using their server to get data and charge you.

6. Postman: testing for API

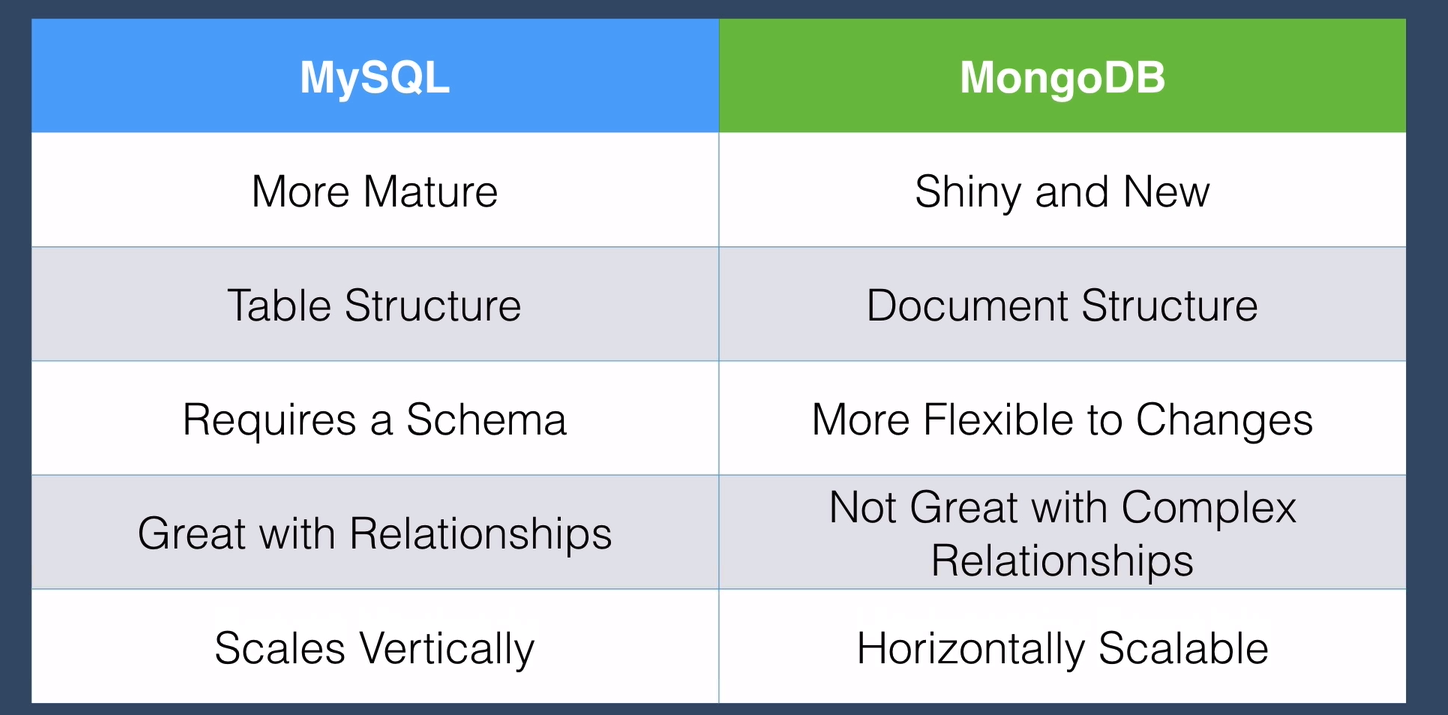
7. JSON format: Javascript Object Notation. The reason we use JSON is because this format is readable by a human, and easily collapsed down and turned back to JS object.

8. inside JS, only res.send() is allowed, which indicates the end of execution.

9. different between SQL and NoSQL: (1) in SQL, if we want to add one more column beyond expectation, it is not convenient to adjust the table. Also, if some info in one row is missing, we may insert NULL to represent this data, this is dangerous. (2) NoSQL: key-val pair. More flexible when the data structure is not pre-defined.

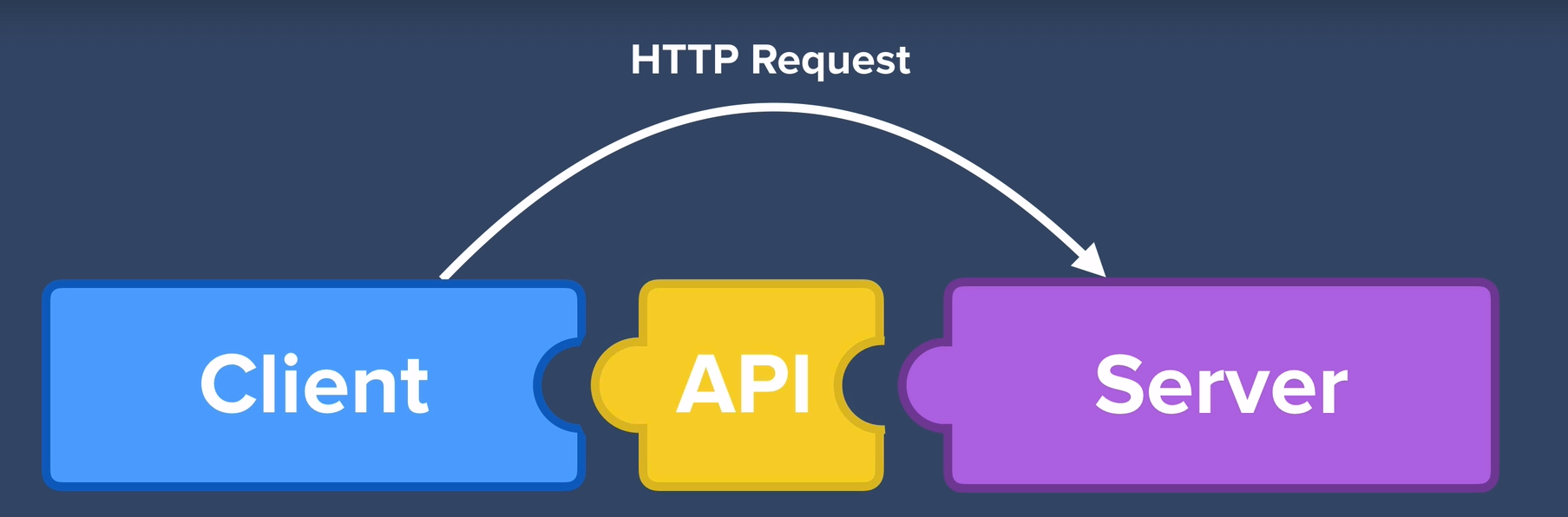
10. How to select database: depends your particular situation. If you need to store data such as orders, customer details ,products inventory, things that have lots of relationships between each other, you’d better choose SQL. If you have a website where you have something that is more of one to many kind of relationship, i.e., Instagram, a single user generate lots of content creating a one user to many posts, MongoDB would be better.

11. NoSQL is more scalable than SQL.



12. ODM: object document mapper. The most popular way of working with MongoDB and Node.js is using package Mongoose.

13. ODM allows your Node.js app which speaks the language of JS objects to be able to talk to your MongoDB database which speaks in the language of documents and collections. That is, connect JS objects with and MongoDB collections.

14. REST: Representational State Transfer. Classical: client->server, through HTTP request, but this is not the only language the server can process. the server will have a whole bunch of APIs which are services that it can expose for clients. 

15. why makes APIs RESTful? REST is essentially just an architectural style for designing APIs.

16. two most important RESTful API rules: (1) use HTTP request Verbs (DB, CRUD): get(read), post(create), put(update), patch(update), delete(delete); (2) use specific pattern of Routes/endpoints URLs

17. the difference between put and patch: put is replace all, patch is replace part of that thing.

18. postman: enable us to send data and test our API without having to worry about building the front-end