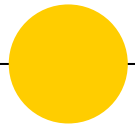
CS4220 Node.js & Angular.js

Cydney Auman
Albert Cervantes
CSULA



APIs & Node.js Server

Node Core Modules

HTTP/HTTPS

- Interfaces designed to support features of the http or https protocol. It provides functionality for running HTTP servers and making HTTP requests.

```
const http = require('http')
const https = require('https')
```

Node HTTP Server

A function is passed as an argument to createServer and is called every time a client tries to connect to the server.

The `request` and `response` variables are objects representing the incoming and outgoing data.

```
const http = require("http");

const server = http.createServer((request, response) => {
  response.writeHead(200, { "Content-Type": "text/plain" })
  response.end("Hello World")
})


server.listen(8000, "localhost")
```

Request

`request` is a request that comes from the client - this sometimes shortened to req.

The request argument contains information about the request, such as its url, http method, headers and etc.

```
http.createServer((request, response) => {
  console.log("Request URL: " + request.url)
  console.log("Request Method: " + request.method)
  console.log("Request Headers: " + JSON.stringify(request.headers))
})
```

Response

The `response` is the next argument in the function. Just like the prior argument is often shortened to res.

```
http.createServer((request, response) => {

  response.writeHead(200, {"Content-Type": "text/plain"})
  response.end("Server is running.")

})
```

With each response, you get the data ready to send, and then you call `response.end()`. Eventually, you ***must*** call this method. This method does the actual sending of data. If this method is not called, the server just hangs forever.

Node with Express

Express.js describes itself as a "a minimal and flexible Node.js web application framework, providing a robust set of features for building single and multi-page, and hybrid web applications."

In short, it's a framework for building web applications with Node.js.

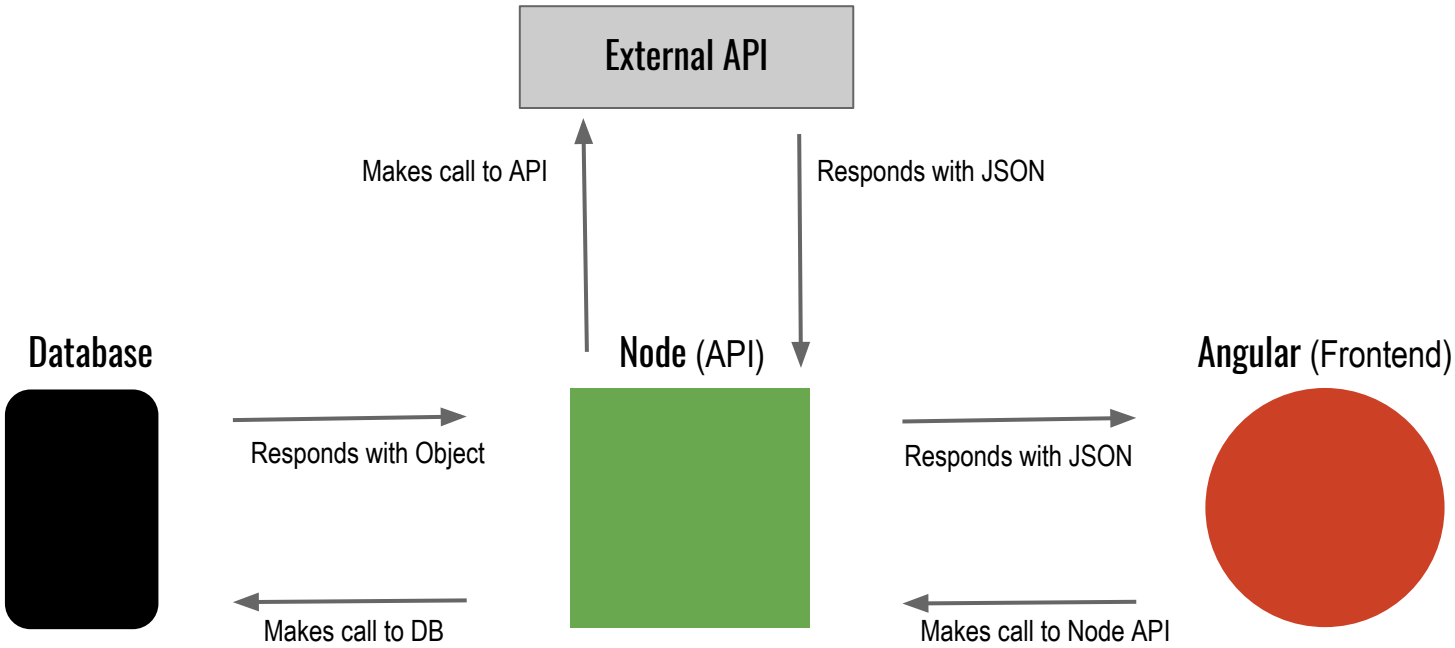
Node with Express

```
const express        = require('express')
const app            = express()

app.get("*",(request, response) => {
   response.writeHead(200, { "Content-Type": "text/plain" })
   response.end("Hello World!")
})

app.listen(8080)
```

Overall Architecture



Web APIs

APIs (application programming interfaces) are a big part of web apps.

The four methods most commonly seen in APIs are:

GET - Asks the server to retrieve a resource

POST - Asks the server to create a new resource

PUT - Asks the server to edit/update an existing resource

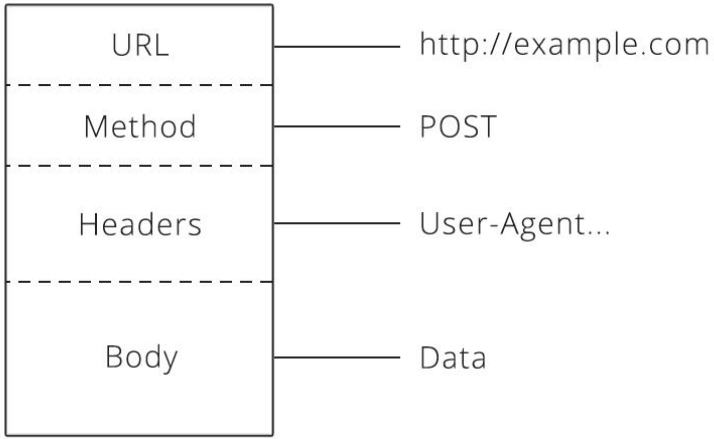
DELETE - Asks the server to delete a resource

APIs - Req/Res

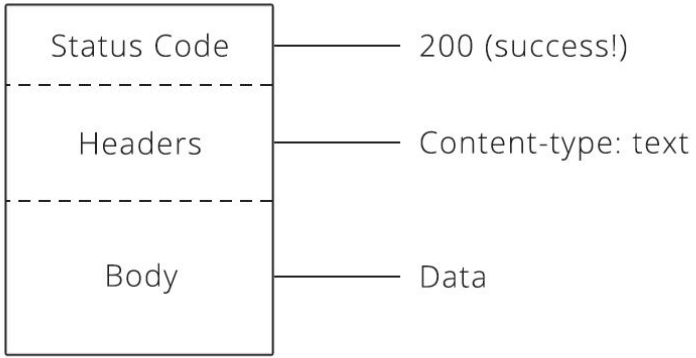
Communication in HTTP centers around a concept called the Request-Response Cycle.

The client sends the server a request to do something. The server, in turn, sends the client a response saying whether or not the server could do what the client asked.

APIs - Req/Res



Request



Response

JSON

APIs have adopted JSON as a format because it's built on the JavaScript, which is found everywhere on the web and usable on both the client side and server side of web apps.

JSON is a very simple format that has two pieces: keys and values. Keys represent an attribute about the object being described.

Angular Factories

Another feature of AngularJS is the ability to encapsulate data functionality into factory, service or provider.

With the factory you create an object inside of the factory and return it.

```
angular
     .module('SomeSrvc', [])
     .factory(SomeService, function($resource) {
          return {
               api: $resource('/api/:type')
          }
     })
```

References and Reading

Eloquent Javascript

-- http://eloquentjavascript.net/20_node.html

Node HTTP

-- https://nodejs.org/dist/latest-v7.x/docs/api/http.html

Express,js

-- https://expressjs.com/

APIs

-- https://zapier.com/learn/apis/chapter-1-introduction-to-apis/