**Web Development memo**

如果是还没决定，要自学，很简答，学javascript。理由如下：

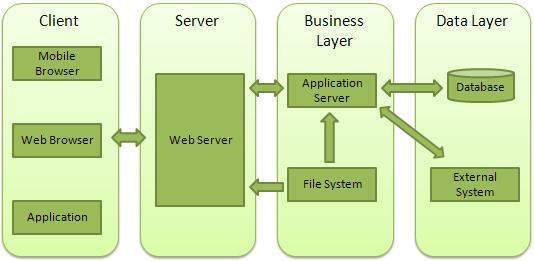
1、对于一个老C++程序员来说，html和CSS就是花一天读w3school的事，无非是用的熟不熟练，根本没有谈的必要。

2、javascript是前端唯一可用的语言，除非你们公司前端纯做展示，不然一定要用js

3、如果公司还没确定技术栈，你就不知道他们会搞什么幺蛾子，到底是java( Spring又或J2EE)，还是php，还是node，又或者是ruby on rails什么的，那么最起码node是很流行的选项之一，而且web后端说白了还是那些东西，登陆啊，验证啊，路由啊，DB啊，我之前从php转node也就一个星期而已。

说起来我以前也是做C开发的，后来想要自己做网站，弄了半年php，现在是node + react，很多东西都是通的

**1. Web 应用架构**



* **Client** - 客户端，一般指浏览器，浏览器可以通过 HTTP 协议向服务器请求数据。
* **Server** - 服务端，一般指 Web 服务器，可以接收客户端请求，并向客户端发送响应数据。
* **Business** - 业务层， 通过 Web 服务器处理应用程序，如与数据库交互，逻辑运算，调用外部程序等。
* **Data** - 数据层，一般由数据库组成。

**2. Http Message**

HTTP requests and HTTP responses use a generic message format of RFC 822 for transferring the required data. This generic message format consists of the following four items.

A Start-line

Zero or more header fields followed by CRLF

An empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields

Optionally a message-body

Http Message example:

**Example 1**

HTTP request to fetch **hello.htm** page from the web server

**Client request head**

GET /hello.htm HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

**Server response head**

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

Content-Length: 88

Content-Type: text/html

Connection: close

**Body**

<html>

<body>

<h1>Hello, World!</h1>

</body>

</html>

**Example 2**

HTTP request to post form data to **process.cgi** CGI page on a web server. The server returns the passed name after setting them as cookies:

**Client request**

POST /cgi-bin/process.cgi HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Content-Type: text/xml; charset=utf-8

Content-Length: 60

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

first=Zara&last=Ali

**Server response**

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Content-Length: 88

Set-Cookie: first=Zara,last=Ali;domain=tutorialspoint.com;Expires=Mon, 19-

Nov-2010 04:38:14 GMT;Path=/

Content-Type: text/html

Connection: close

**Body**

<html>

<body>

<h1>Hello Zara Ali</h1>

</body>

</html>

**Start line instruction:**

A start-line will have the following generic syntax:

start-line = Request-Line | Status-Line

We will discuss Request-Line and Status-Line while discussing HTTP Request and HTTP Response messages respectively. For now, let's see the examples of start line in case of request and response:

**GET /hello.htm HTTP/1.1** (This is Request-Line sent by the client using get)

**POST /cgi-bin/process.cgi HTTP/1.1** (This is Request-Line sent by the client using post)

**HTTP/1.1 200 OK** (This is Status-Line sent by the server)

**Request-Line**

Request-Line = **Request Method** SP ***Request-URI*** SP **HTTP-Version** CRLF

Example: GET /hello.htm HTTP/1.1

1. **or** a verb (like GET, PUT or POST) or a noun (likeHEAD or OPTIONS), that describes the action to be performed. For example, GET indicates that a resource should be fetched or POST means that data is pushed to the server (creating or modifying a resource, or generating a temporary document to send back).

**Request Method: Get--**indicates a resource should be fetched, should have no other effect on the data;

**Request Method: Post--**indicates the data is pushed to the server (eg. Create or modify a resource, or generate doc to send back);

The following table lists all the supported methods in HTTP/1.1.

|  |  |
| --- | --- |
| **S.N.** | **Method and Description** |
| 1 | **GET** The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data. |
| 2 | **HEAD** Same as GET, but it transfers the status line and the header section only. |
| 3 | **POST** A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms. |
| 4 | **PUT** Replaces all the current representations of the target resource with the uploaded content. |
| 5 | **DELETE** Removes all the current representations of the target resource given by URI. |
| 6 | **CONNECT** Establishes a tunnel to the server identified by a given URI. |
| 7 | **OPTIONS** Describes the communication options for the target resource. |
| 8 | **TRACE** Performs a message loopback test along with the path to the target resource. |

1. **The request target,** usually a URL, or the absolute path of the

protocol, port, and domain are usually characterized by the request context. The format of this request target varies between different HTTP methods. It can be:

* An absolute path, ultimately followed by a '?' and query string.This is the most common form, known as the *origin form*, and is used with GET, POST, HEAD, and OPTIONS methods.

POST / HTTP/1.1

GET /background.png HTTP/1.0

HEAD /test.html?query=alibaba HTTP/1.1

OPTIONS /anypage.html HTTP/1.0

* A complete URL, known as the *absolute form*, is mostly used

with GET when connected to a proxy.

GET http://developer.mozilla.org/en-

US/docs/Web/HTTP/Messages HTTP/1.1

* The authority component of a URL, consisting of the domain

name and optionally the port (prefixed by a ':'), is called the *authority form*. It is only used with CONNECT when setting up an HTTP tunnel.

* CONNECT developer.mozilla.org:80 HTTP/1.1

The *asterisk form*, a simple asterisk ('\*') is used with OPTIONS, representing the server as a whole. OPTIONS \* HTTP/1.1

3) **The *HTTP version***, which defines the structure of the remaining

message, acting as an indicator of the expected version to use for

the response.

**3. Node.js**

[node-gitignore](https://www.npmjs.com/package/node-gitignore)

Top of Form

Bottom of Form

Create .gitignore from [**https://raw.githubusercontent.com/github/gitignore/master/Node.gitignore**](https://raw.githubusercontent.com/github/gitignore/master/Node.gitignore)

and add the follow:

# .idea

.idea

# Dependency directory

bower\_components

**Usage**

$ npm install node-gitignore -g

$ node-gitignore