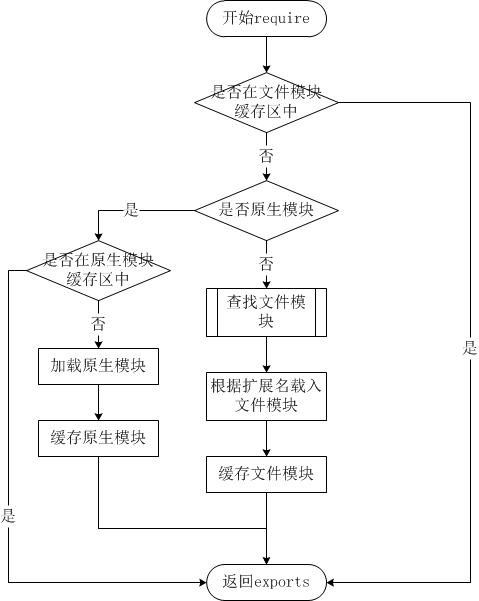
1. Node.js 应用是由哪几部分组成的：
2. **引入 required 模块：**我们可以使用 **require** 指令来载入 Node.js 模块。
3. **创建服务器：**服务器可以监听客户端的请求，类似于 Apache 、Nginx 等 HTTP 服务器。
4. **接收请求与响应请求** 服务器很容易创建，客户端可以使用浏览器或终端发送 HTTP 请求，服务器接收请求后返回响应数据。
5. Node Package Manager(NPM)
6. 允许用户从NPM服务器下载别人编写的第三方包到本地使用。
7. 允许用户从NPM服务器下载并安装别人编写的命令行程序到本地使用。
8. 允许用户将自己编写的包或命令行程序上传到NPM服务器供别人使用。
9. Node.js REPL(Read Eval Print Loop:交互式解释器) 表示一个电脑的环境，类似 Window 系统的终端或 Unix/Linux shell，我们可以在终端中输入命令，并接收系统的响应。
10. Node 自带了交互式解释器，可以执行以下任务：
11. 读取 - 读取用户输入，解析输入了Javascript 数据结构并存储在内存中。
12. 执行 - 执行输入的数据结构
13. 打印 - 输出结果
14. 循环 - 循环操作以上步骤直到用户两次按下 ctrl-c 按钮退出。
15. Node.js 是单进程单线程应用程序，但是通过事件和回调支持并发，所以性能非常高。
16. Node.js 的每一个 API 都是异步的，并作为一个独立线程运行，使用异步函数调用，并处理并发。
17. Node.js 基本上所有的事件机制都是用设计模式中观察者模式实现。
18. Node.js 单线程类似进入一个while(true)的事件循环，直到没有事件观察者退出，每个异步事件都生成一个事件观察者，如果有事件发生就调用该回调函数.
19. Node.js 使用事件驱动模型，当web server接收到请求，就把它关闭然后进行处理，然后去服务下一个web请求。 当这个请求完成，它被放回处理队列，当到达队列开头，这个结果被返回给用户。

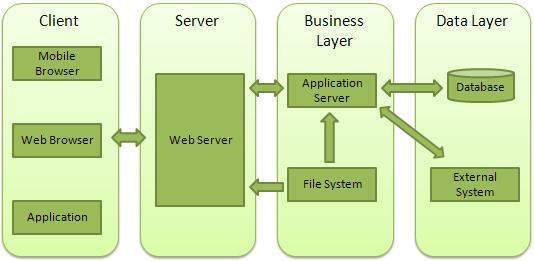




1. events 模块只提供了一个对象： events.EventEmitter。EventEmitter 的核心就是事件触发与事件监听器功能的封装。
2. **on** 函数用于绑定事件函数，**emit** 属性用于触发一个事件。
3. JavaScript 语言自身只有字符串数据类型，没有二进制数据类型。因此在 Node.js中，定义了一个 Buffer 类，该类用来创建一个专门存放二进制数据的缓存区。
4. Node.js 的 require方法中的文件查找策略：



1. **setTimeout(cb, ms)** 全局函数在指定的毫秒(ms)数后执行指定函数(cb)。：setTimeout() 只执行一次指定函数。
2. **setInterval(cb, ms)** 全局函数在指定的毫秒(ms)数后执行指定函数(cb)，setInterval() 方法会不停地调用函数，直到 clearInterval() 被调用或窗口被关闭。
3. 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。

1. Express 框架核心特性：

* 可以设置中间件来响应 HTTP 请求。
* 定义了路由表用于执行不同的 HTTP 请求动作。
* 可以通过向模板传递参数来动态渲染 HTML 页面。