NodeJS

INTRO

- What Is NodeJS?
- Why & When NodeJS?
- Blocking & Non Blocking [callback]
- Modules
- NPM
- Create Your Server
- Events
- Streams

What is NodeJS ?

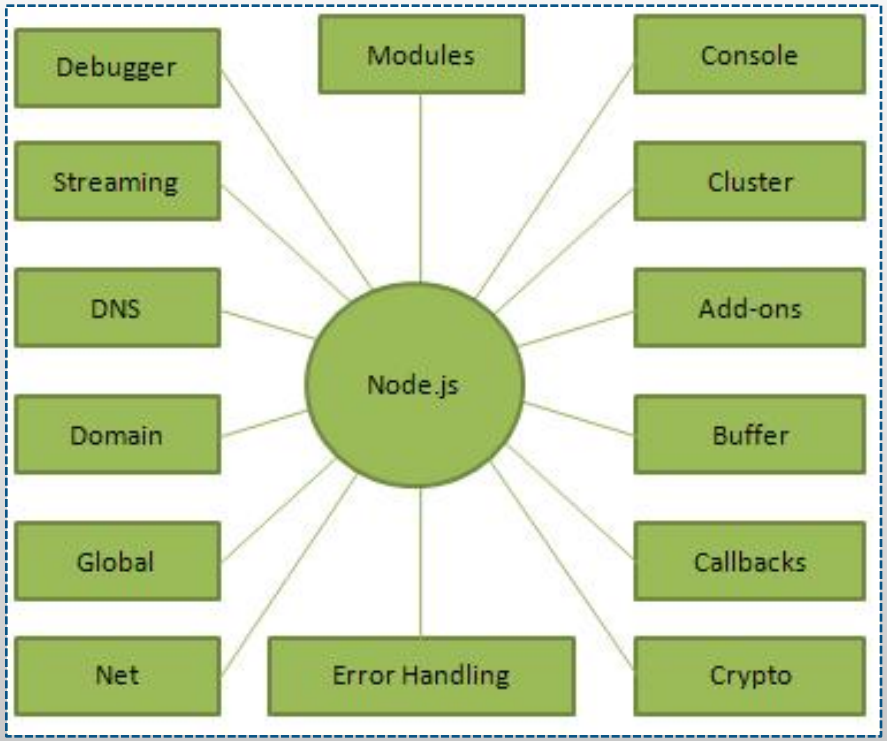
Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009

Allows you to build scalable network applications using JavaScript on the server-side.

It is fast because it is mostly c code

What is NodeJS ?

Node.js = Runtime Environment + JavaScript Library



Why NodeJS ?

- Very Fast
- Single Threaded but Highly Scalable
- No Buffering
- Asynchronous and Event Driven

Features of Node . js

Very Fast – Being built on Google Chrome's V8 JavaScript Engine, Node . js library is very fast in code execution .

Single Threaded but Highly Scalable – Node . js uses a single threaded model with **event looping** . Event mechanism helps the server to respond in **a non–blocking** way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests . Node . js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like **Apache HTTP Server** .

Features of Node . js

No Buffering – Node . js applications never buffer any data . These applications simply output the data in chunks .

Asynchronous and Event Driven – All APIs of Node . js library are asynchronous, that is, **non–blocking** . It essentially means a Node . js based server never waits for an API to return data . The server moves to the next API after calling it and a notification mechanism of Events of Node . js helps the server to get a response from the previous API call .

Blocking & Non Blocking Code

IF I need to write code for reading file content

Blocking Code

Read file from File system, set equal to "contents"
Print contents
Do something else

Non Blocking Code

Read file from File system, set equal to "contents"
whenever you're complete, print the contents
Do something else

This is a callback

Blocking & Non Blocking Code

IF I need to write code for reading file content

Blocking Code

```
var contents = fs.readFileSync(" /etc /hosts");
console.log(contents);
console.log("Doing something else");
```

Stop process until complete

Non Blocking Code

```
fs.readFile(" /etc /hosts", function(err, contents) {
        console.log(contents);
});
console.log("Doing something else");
```

Callback

Modules

Import Library to your code

1. How does “require” return the libraries?
2. How does it find these files?

```
var library= require(“ . /library__path”);
var library= require(“ . . /library__path”);
var library= require(“ /home /username /library__path”);
var library= require(“library__path”);
```

look in same directory

look in parent directory

Search in **node__modules** directories

Will load library once to your project and If you load it again will return the same object

Modules

Create your module

module__file1.js

```
var hello = function() {
        console.log("hello");
}
module.exports = hello;
```

module__file2.js

```
var bye= function() {
        console.log("bye");
}
exports.bye = bye;
```

app.js

```
var hello = require(" ./module__file1");
hello();
var gb = require(" ./module__file2");
gb. bye();
```

Install Node

Install Node on Centos 7

```
yum install -y gcc-c++ make
curl -sL https://rpm.nodesource.com/setup_6.x | sudo -E bash -
yum install nodejs
```

Will Install Node & Node Package Manager (**NPM**)

NPM – The User land Sea

Package manager for node

Comes with node

Module Repository

Dependency Management

Easily publish modules

Installing A NPM Module

In your project Directory

```
$ npm install module__name
OR
$ npm install module__name -g
```

Local

Global

Local : Installs into local **node__modules** directory

project__directory /**node__modules** /module__name

to import local module in your project

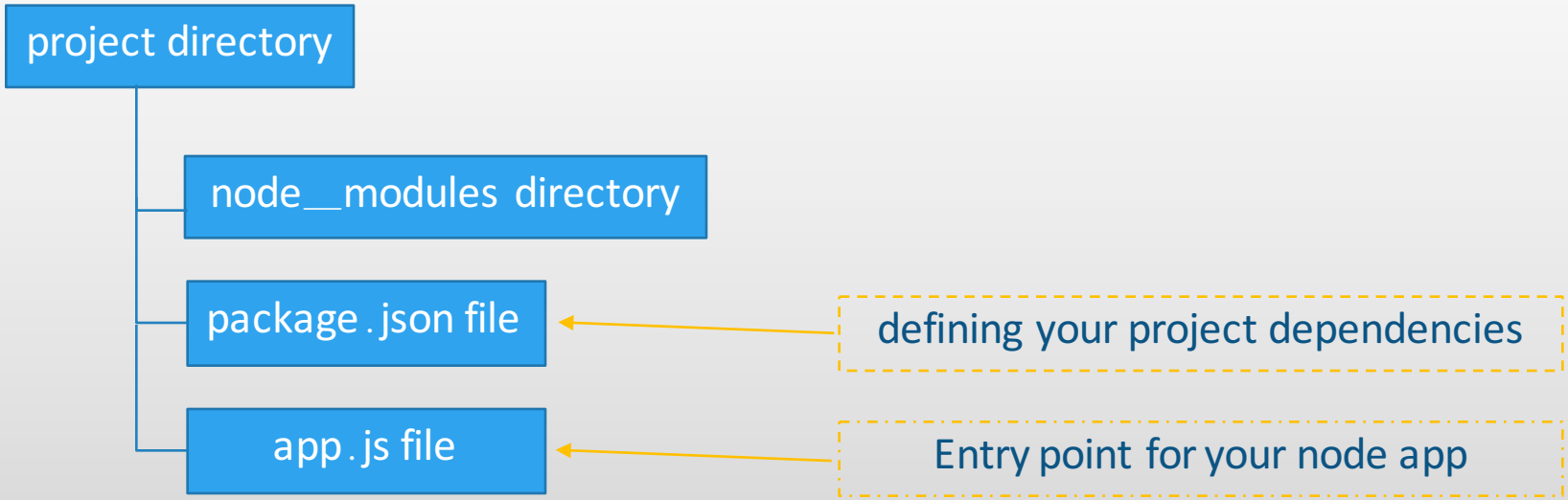
```
var obj= require("module__name");
```

Loads from local node__modules directory

Global Modules can not be required

First App with Node

Project Structure



Package . json

Package . json file

```
{
        "name" : "My App",
        "version" : "1",
        "dependencies" : {
                "module1_name":"1.8.7",
                "module2_name" :"~2",
                "module3_name" :"~1.8",
                "module4_name" :"~2.7.5"
        }
}
```

Version Number

major minor patch

>=2.0.0 <3.0.0

>=1.8.0 <1.9.0

>=2.7.5 <2.7.9

To install all dependencies just go to project directory run this command **npm install**

Create a Server

app.js or **Entry point**

```
var http = require("http");
http.createServer(function(request, response) {
        response.writeHead(200);
        response.write("Hello, this is dog.");
        response.end();
})
http.listen(8080, function(){
        console.log("start listening");
})
```

import **http** module

set event listener to listen for incoming requests

Start listen for incoming requests on port **8080**

To run this program just write in your terminal

node file_name

node app.js

Access a Server

app.js or **Entry point**

To run this program just write in your terminal

node file_name

node app.js

To Test Your Server , open your browser and access this URL

http://localhost:8080

The Event Loop

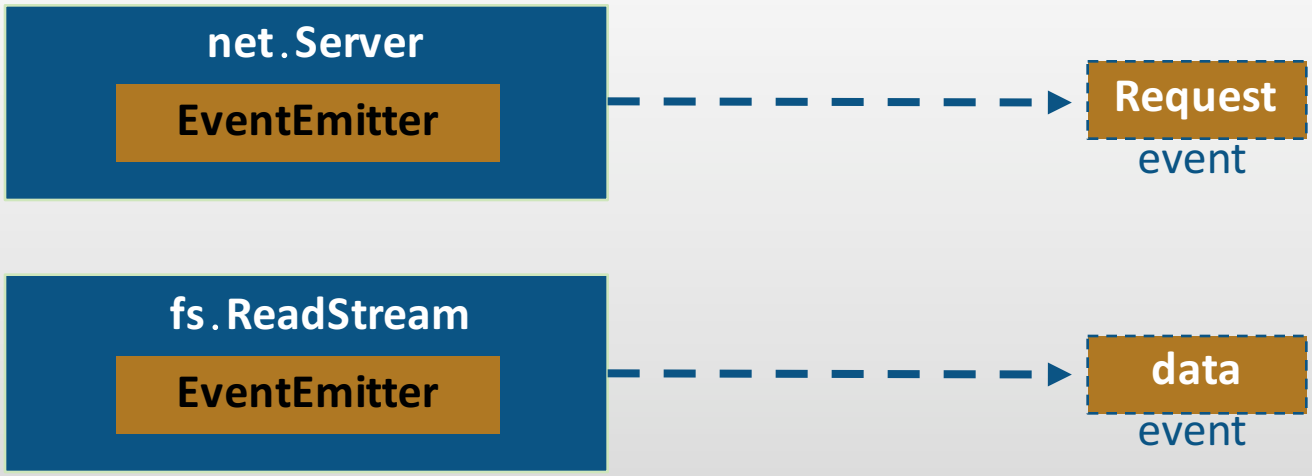
```
var http = require("http");
http.createServer(function(request, response) {

})
http.listen(8080, function(){
        console.log("start listening");
})
```

Events processed one at a time

Event In Node

Many objects in Node **emit** events



Custom Event Emitters

```
var EventEmitter = require("events").EventEmitter;

var logger = new EventEmitter();

//register callback for error event
logger.on("error", function(message){
        console.log('ERR: ' + message); });
});

logger.emit("error","test message");
```

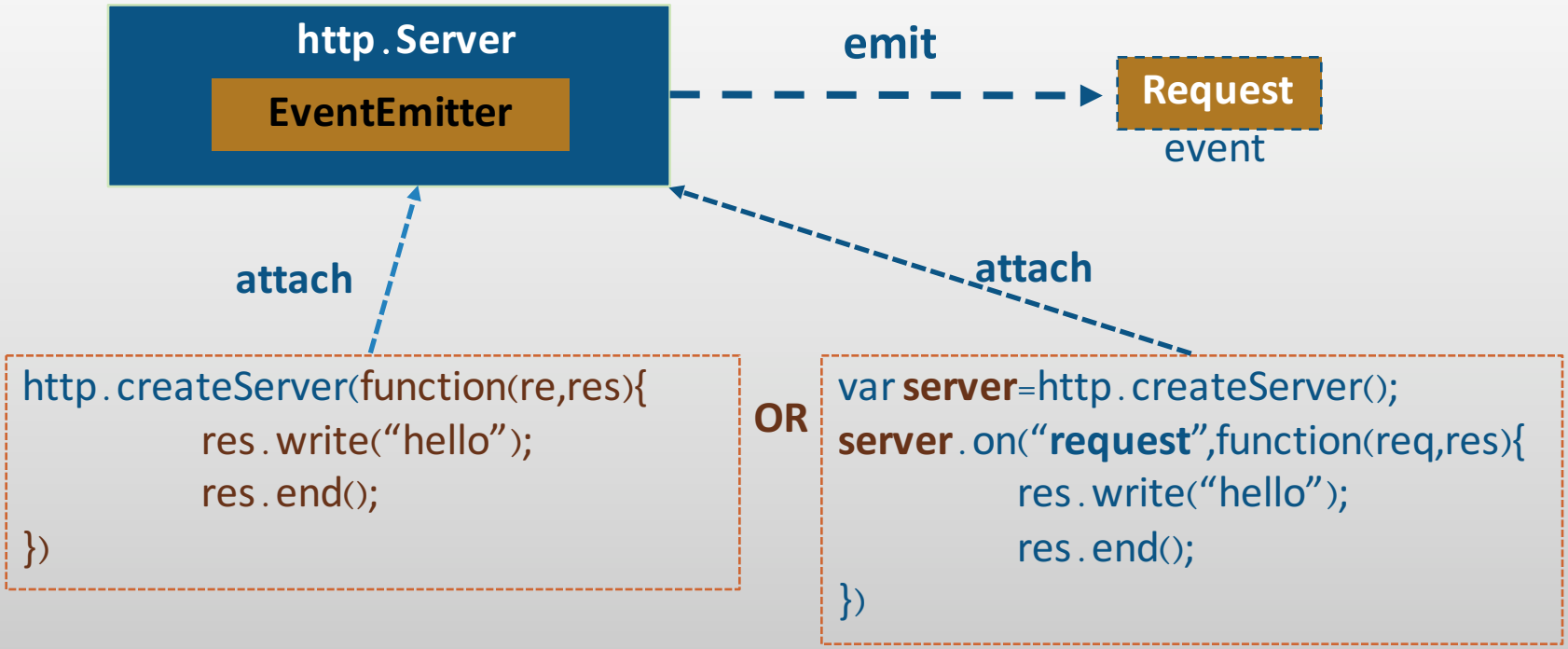
Load events module

Register callback for event and named event with name "error"

Fire "error" event

Event In Node

When "**request**" event is emitted



Stream

If you need to read from or write into file , you need to create stream

represents a sequence of bytes, which can be accessed in sequential order

File System Module

File I /O is provided by simple wrappers around standard POSIX functions. To use this module do **require("fs")**. All the methods have **asynchronous** and **synchronous** forms.

Use fs module to

1. Read from file
2. Write into file
3. Delete file
4. Rename file
5. Check file status

File I / O

Read from File

Asynchronous Method

```
var fs=require("fs");
fs.readFile("test.txt",function(error , data){
        console.log(data. toString());
});
console.log("complete");
```

Synchronous Method

```
var fs=require("fs");
var content=fs.readFileSync("test.txt");
console.log("content");
```

File I / O

Write into File

Asynchronous Method

```
var fs=require("fs");
fs.writeFile("test.txt","data",function(error{
            console.log(error);
});
console.log("complete");
```

Synchronous Method

```
var fs=require("fs");
var content=fs.writeFileSync("test.txt","data");
 console.log("content");
```

File Status

Check file status

Asynchronous Method

```
var fs=require("fs");
fs.lstat("test.txt",function(error, stats){
        console.log(stats. isFile());
});
```

Synchronous Method

```
var fs=require("fs");
var status=fs. lstatSync("test.txt");
 console.log(status. isFile());
```

Event Emitter – File System

```
var fs=require("fs");
var reader=fs.createReadStream("test.txt");
reader.on("data",function(data){
        console.log(data.toString());
})
reader.on('end',function(){
        console.log("end");
})
```

CreateReadStream return EventEmitter Object which emit data event when data received from the file

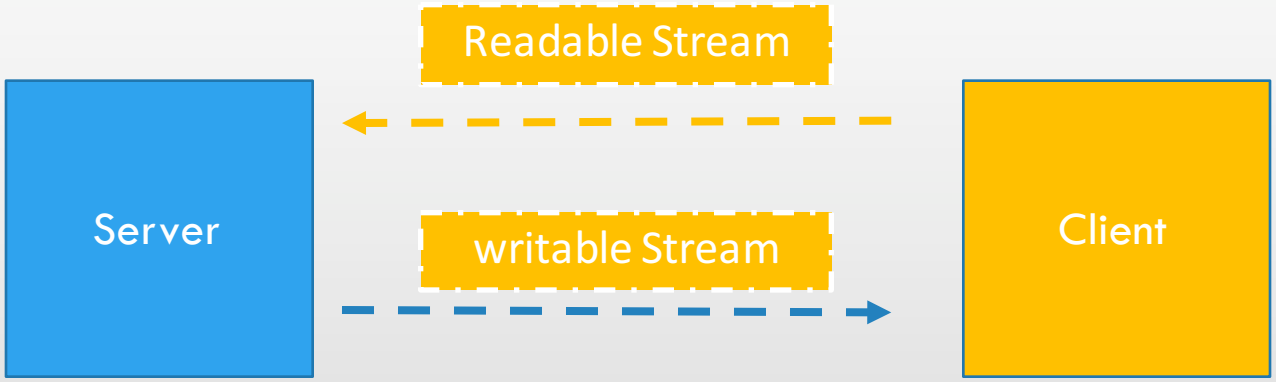
Writer Stream

```
var fs=require("fs");
var reader=fs.createReadStream("test.txt");
var writer=fs.createWriteStream("new__test.txt");
reader.on("data",function(data){
        writer.write(data);
})
reader.on('end',function(){
        console.log("end");
})
```

PIPE

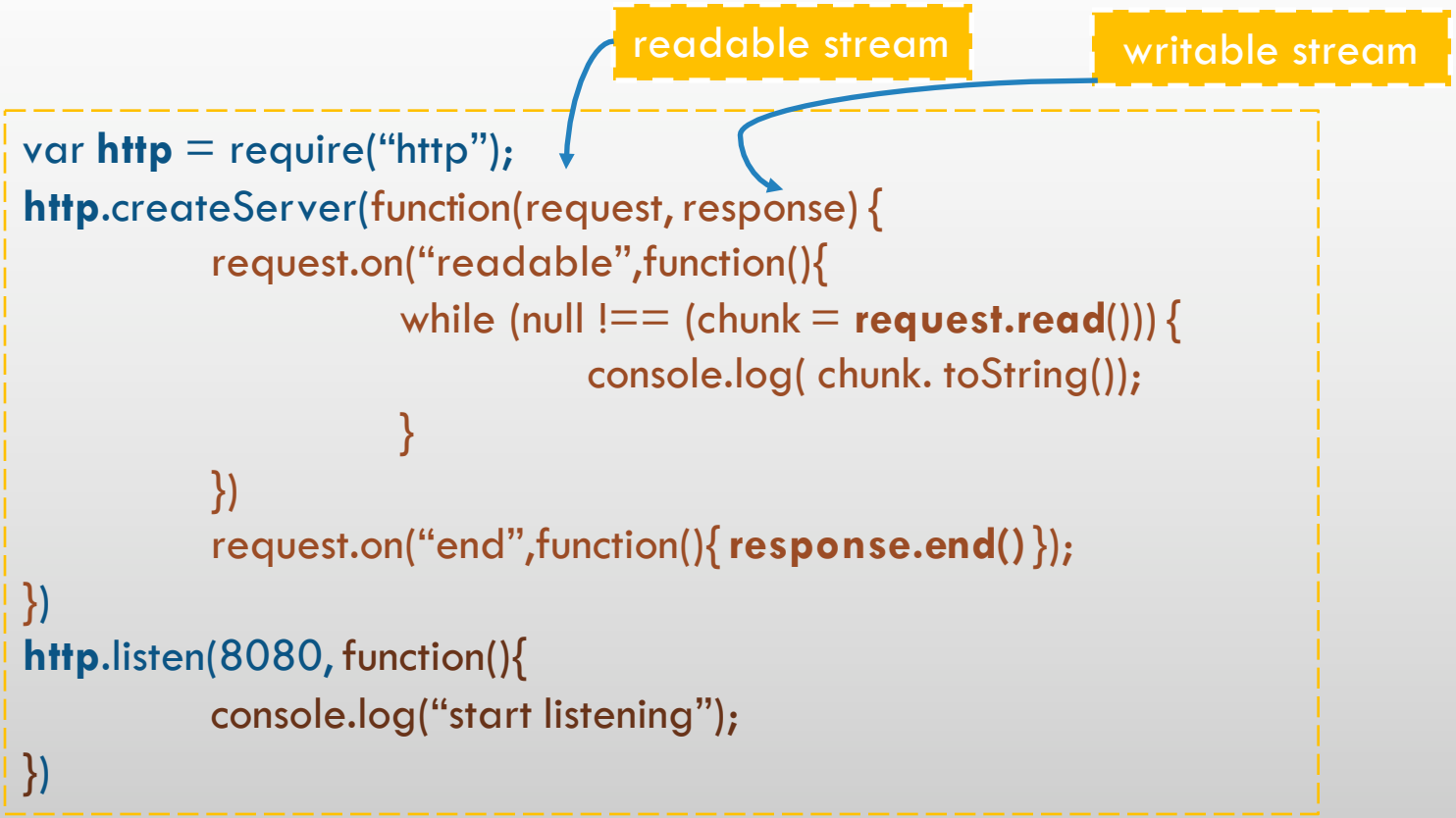
```
var fs=require("fs");
var reader=fs.createReadStream("test.txt");
var writer=fs.createWriteStream("new__test.txt");
reader.pipe(writer)
```

Server Stream



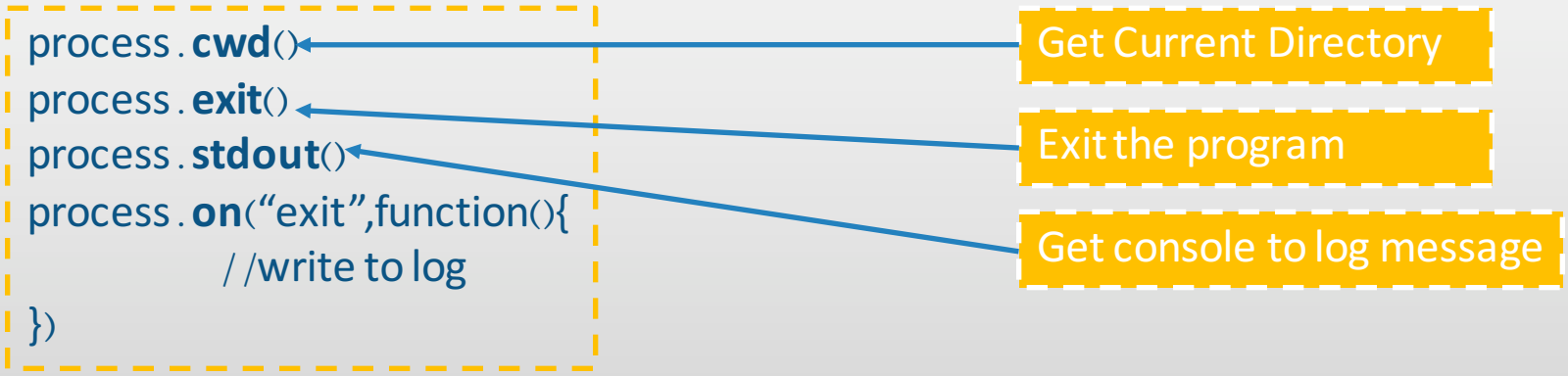

```
http.createServer(function( request,response ){})
```

Server Stream



Process Module

The **process** object is a global that provides information about, and control over, the current Node.js process. As a global, it is always available to Node.js applications **without using require()**.



Most Used Modules

- path
- URL
- crypto
- DNS
- Request
- querystring

https://nodejs.org/api/

Demo

Code Time