

# Node

---

INTRODUCTION

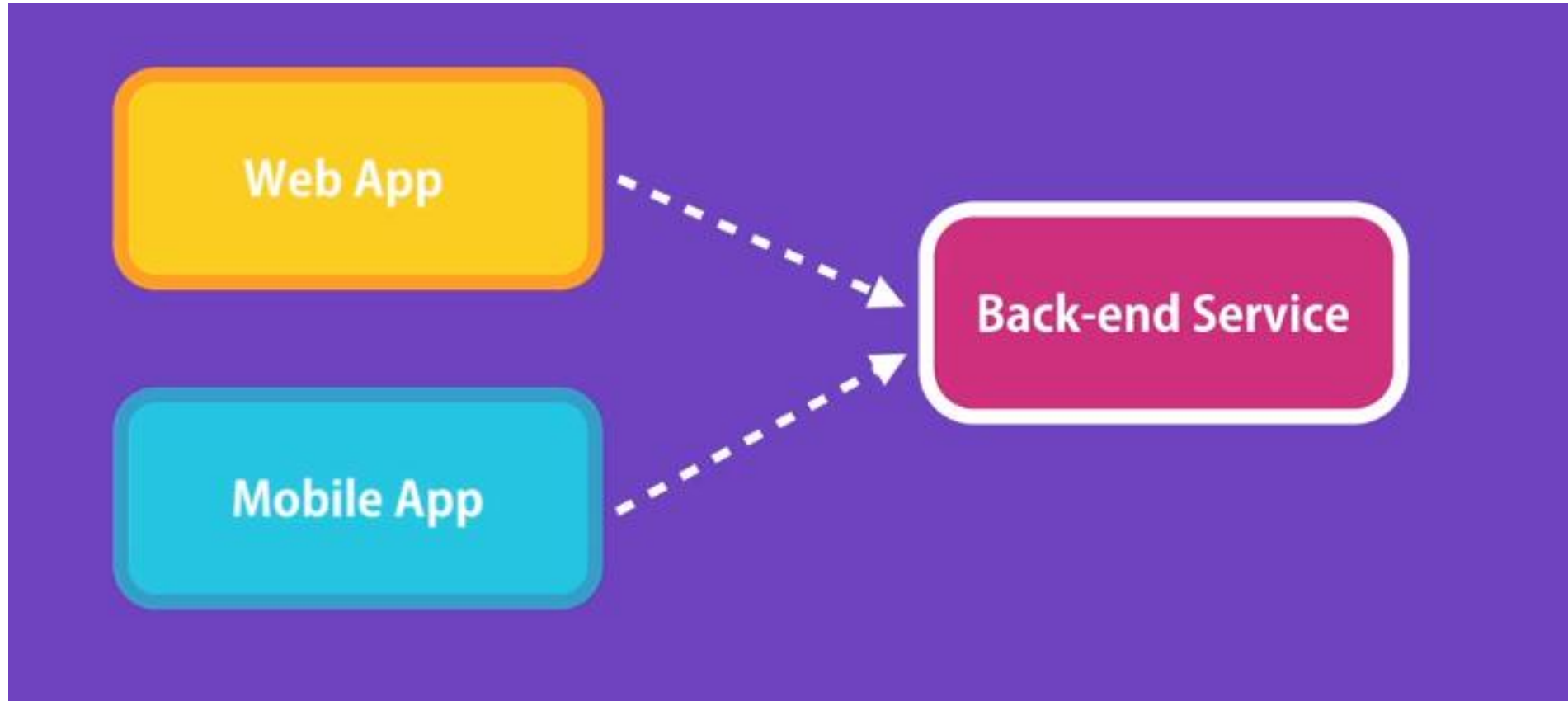
# Node Getting Started

---

A run Time Environment for Executing Java Script Code.  
Mostly we build ....



Application Programming Interface

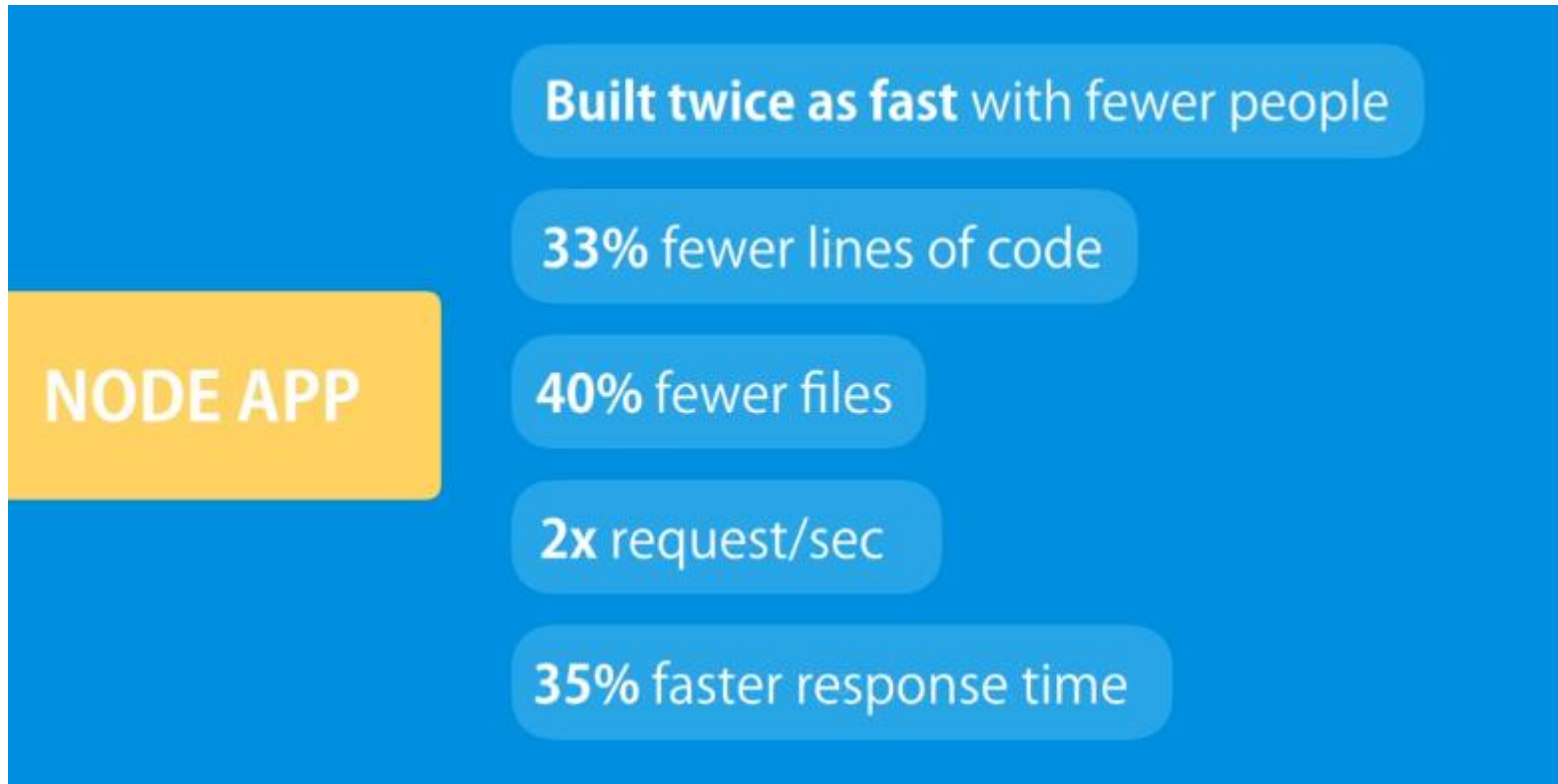


Highly-scalable, data-intensive  
and real-time apps

# Why Node

---

Paypal, Uber, NetFlix, Wall Mart



**Great for prototyping and agile development**

**Superfast and highly scalable**

**JavaScript everywhere**

**Cleaner and more consistent codebase**

**Large ecosystem of open-source libs**

# Node Architecture

---



Chakra



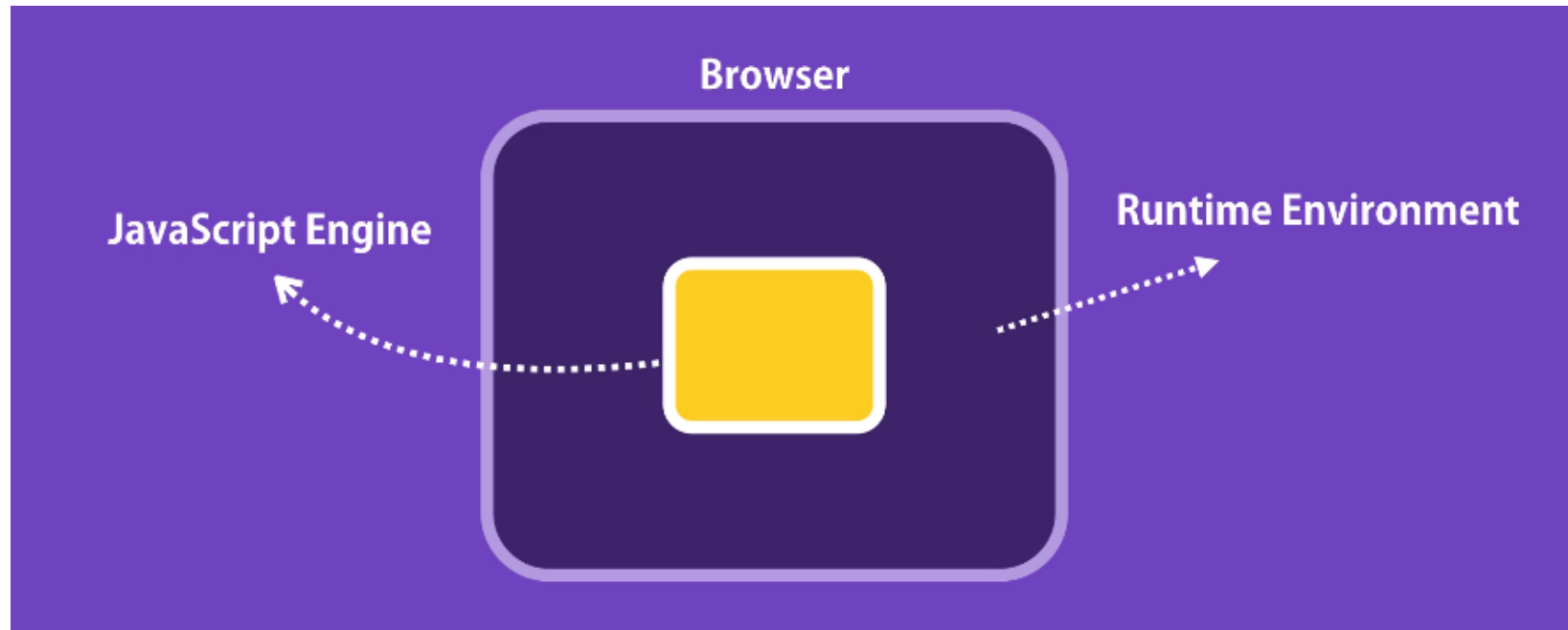
SpiderMonkey



v8

# Node Architecture

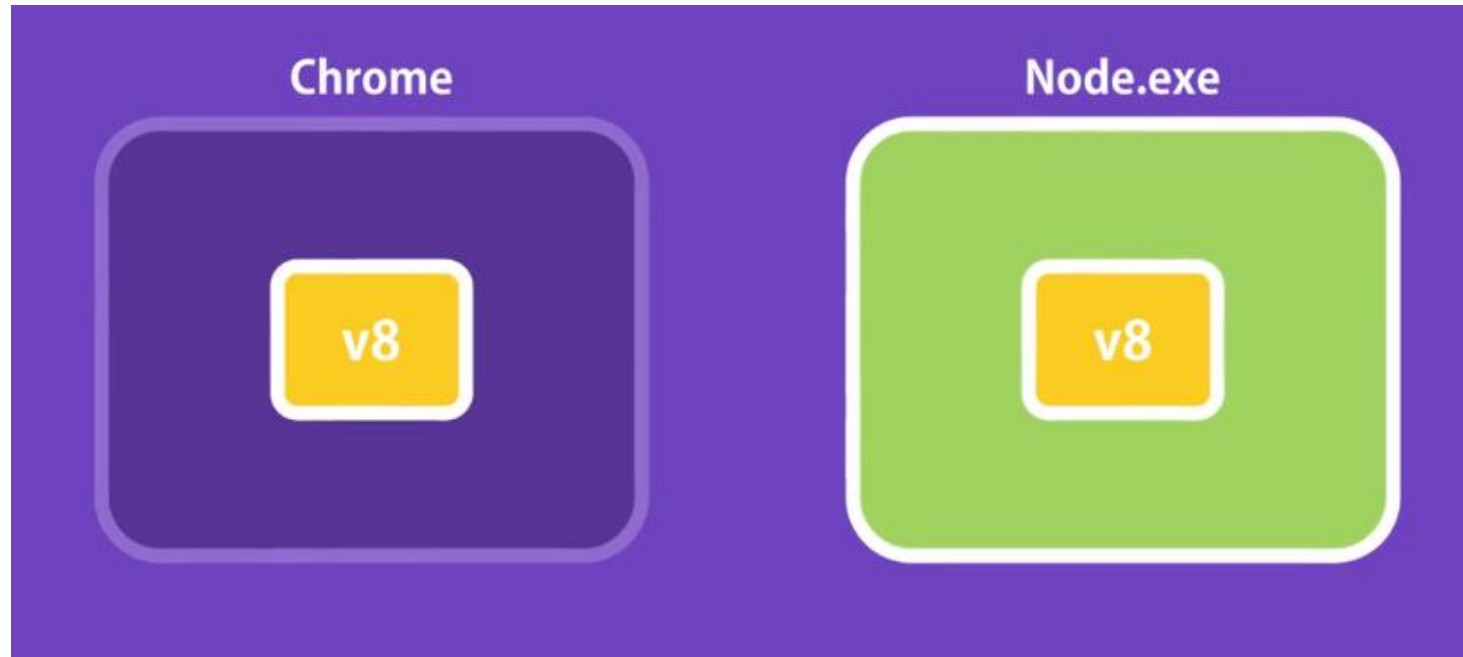
---





# Node Architecture by Ryan Dahl

---



# JSON

---

JAVASCRIPT OBJECT NOTATION

# Object

---

```
{  
    "name":      "Jack B. Nimble",  
    "at large":  true,  
    "grade":     "A",  
}
```

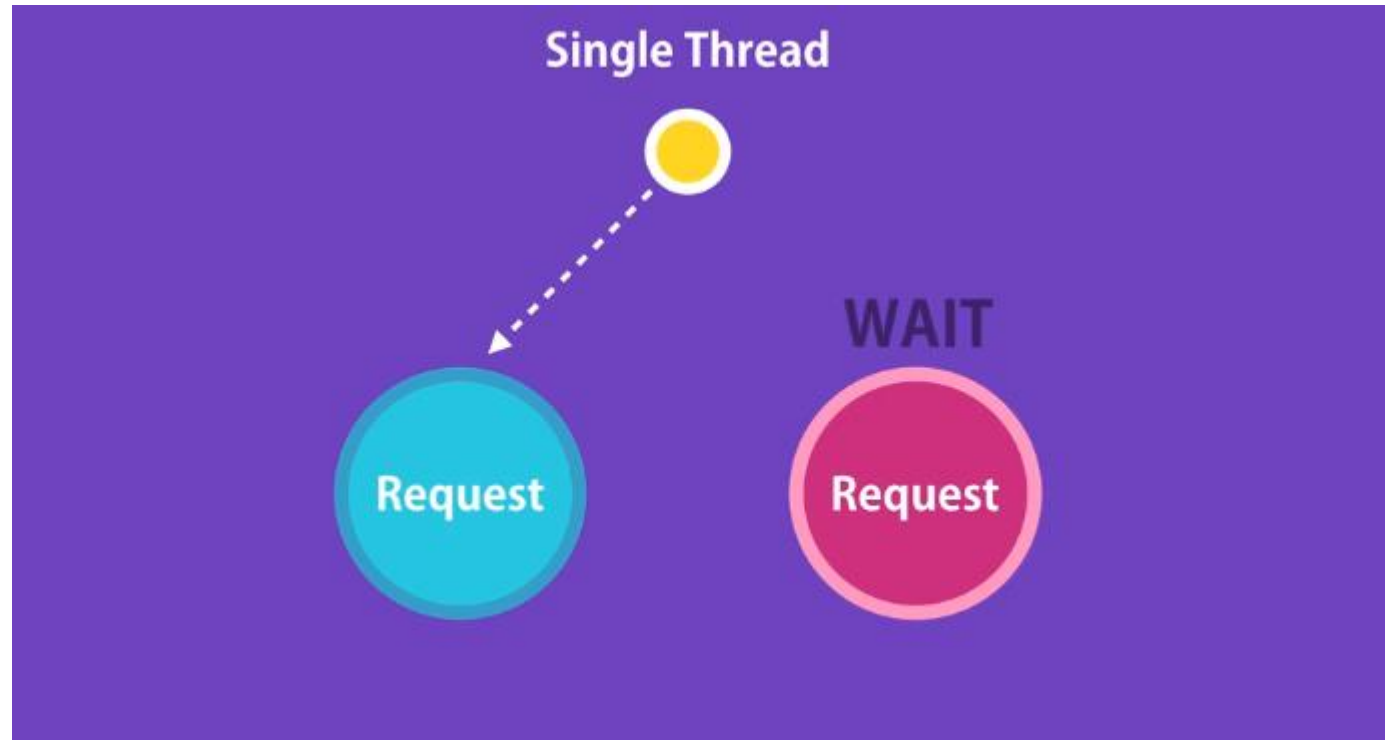
# Node Architecture by Ryan Dahl

---

```
document.getElementById('');  
fs.readFile()  
http.createServer()
```

# Not suitable for computational Hungry Tasks

---



[HOME](#)[ABOUT](#)[DOWNLOADS](#)[DOCS](#)[GET INVOLVED](#)[SECURITY](#)[NEWS](#)[FOUNDATION](#)

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

## Download for Windows (x64)

**10.15.0 LTS**

Recommended For Most Users

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

**11.6.0 Current**

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

# node app.js

---

```
function sayHello(name) {  
  console.log("Welcome " + name);  
}  
sayHello("Usman");  
  
//window or document objects are not available here
```

# Node Recap

---

- Node is a runtime environment for executing JS code. –
- Essentially, Node is a C++ program that embeds Chrome's v8 engine, the fastest JS engine in the world. –
- We use Node to build fast and scalable networking applications. It's a perfect choice for building RESTful services. –
- Node applications are single-threaded. That means a single thread is used to serve all clients. –



# Node Recap

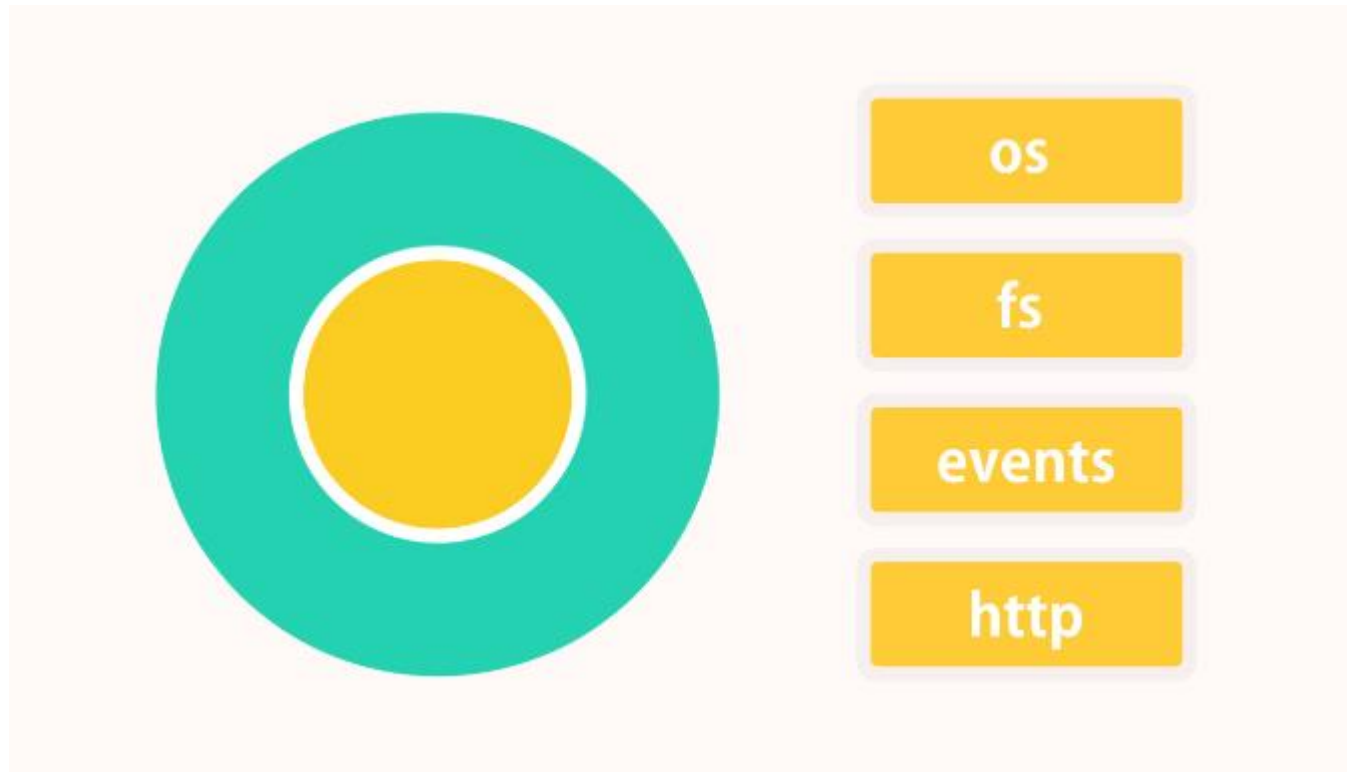
---

- Node applications are asynchronous or non-blocking by default. That means when the application involves I/O operations (eg accessing the file system or the network), the thread doesn't wait (or block) for the result of the operation. It is released to serve other clients. –
- This architecture makes Node ideal for building I/O-intensive applications. –
- You should avoid using Node for CPU-intensive applications, such as a video encoding service. Because while executing these operations, other clients have to wait for the single thread to finish its job and be ready to serve them. –
- In Node, we don't have browser environment objects such as window or the document object. Instead, we have other objects that are not available in browsers, such as objects for working with the file system, network, operating system, etc.

# Node

---

MODULE SYSTEM



# Window object is not available here

---

```
console.log(); // global  
  
setTimeout()  
clearTimeout();  
  
setInterval()  
clearInterval()  
  
window.console.log
```

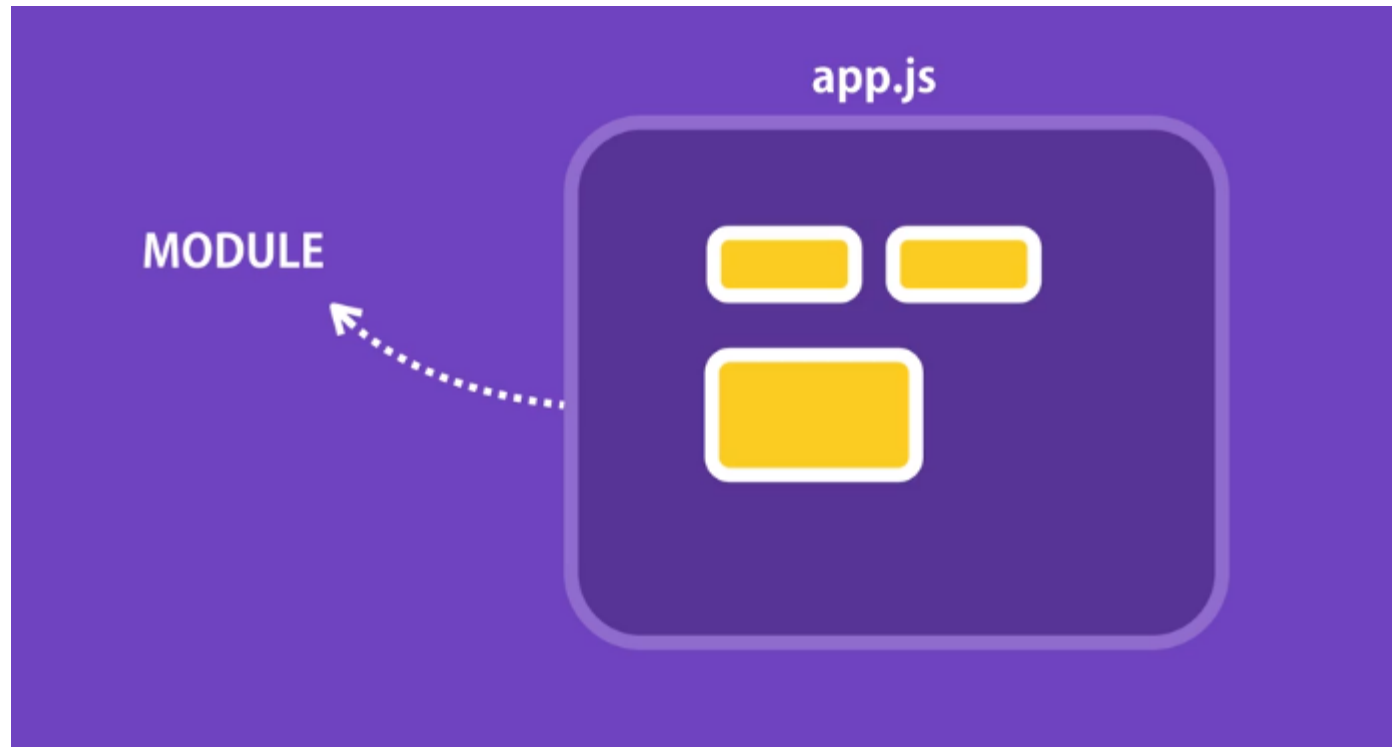
# Modularity

---



# Every file is a module

---



# Logger.js exporting log function and url

---

```
//parameters available here
//exports,require, module, __filename,__dirname
function log(message) {
  console.log(message);
}
var url="usmanlive.com";
console.log(__filename);
console.log(__dirname);
module.exports.log = log;
module.exports.url = url;
```

# Using logger.js

---

```
const logger = require ('./logger');  
logger.log("Hello Usman");
```

# Exercise

---

## Install jshint

- `npm install -g jshint`
- `jshint app.js`

Head to [nodejs.org](http://nodejs.org) and explore docs



# Built in modules (path)

---

```
const path = require('path');  
const pathObj = path.parse(__dirname);  
console.log(pathObj);
```

```
first-app $  
first-app $node app.js  
{ root: '/',  
  dir: '/Users/moshfeghhamedani/Desktop/node-course/first-app',  
  base: 'app.js',  
  ext: '.js',  
  name: 'app' }  
first-app $
```

# Built in modules (fs)

---

```
const fs = require('fs');  
//sync should be avoided at all cost  
//console.log(fs.readdirSync("./"));  
files = fs.readdir('./', function  
name(err, result) {  
  if(result) console.log(result);  
});  
console.log(files);
```

# Built in modules (os)

---

```
const os = require('os');  
console.log(os.freemem());  
console.log(os.uptime());  
//template string available with ES6  
console.log(`Home Directory ${os.homedir}`);
```

# Built in modules (os)

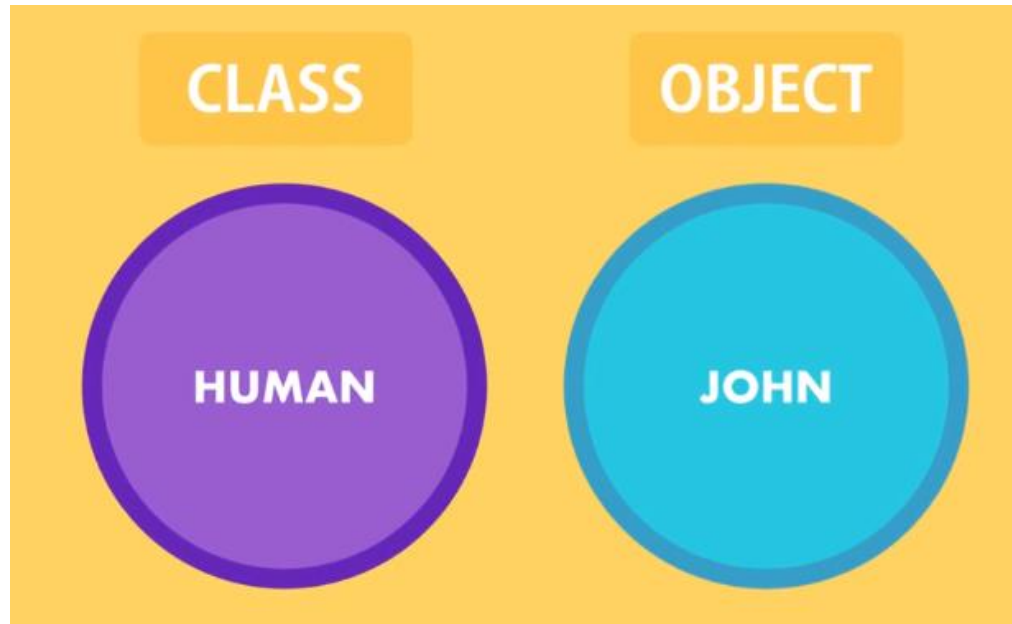
---

```
const fs = require('fs');  
//sync should be avoided at all cost  
//console.log(fs.readdirSync("./"));  
files = fs.readdir('./', function  
name(err, result) {  
  if(result) console.log(result);  
});  
console.log(files);
```

# Node Events

---

```
const EventEmitter = require('events');  
const emitter = new EventEmitter();
```



# Here's a prototype-based class User

---

```
function User(name) {  
  this.name = name;  
}  
User.prototype.sayHi = function() {  
  alert(this.name);  
}  
let user = new User("John");  
user.sayHi();
```

# The “class” syntax

---

```
class User {  
  constructor(name) {  
    this.name = name;  
  }  
  sayHi() {  
    alert(this.name);  
  }  
}  
  
let user = new User("John");  
user.sayHi();
```

# A MessageLogged Event

---

```
const EventEmitter = require('events');
const emitter = new EventEmitter();
emitter.on("MessageLogged", function(args) {
    console.log("Event Received", args);
})
emitter.emit('MessageLogged', {id:1, name: 'hello'});
```



# logger.js

---

```
const EventEmitter = require('events');
class Logger extends EventEmitter {
  log(message) {
    console.log(message);
    this.emit("MessageLogged"); //pass params also
  }
}
module.exports = Logger;
```

# app.js

---

```
const Logger = require ('./logger');  
var logger = new Logger();  
logger.on('MessageLogged',function () {  
    console.log("Event captured");  
});  
logger.log("Log this mssage");
```

# Node http module

---

WHOALA WE ARE BUILDING API NOW

# http

---

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

const server = http.createServer();

server.on('connection', (socket) => {
  console.log('New connection...');
});

server.listen(3000);

console.log('Listening on port 3000...');
```

# First api call listener – localhost:3000

---

```
const http = require('http');
const server = http.createServer((req, res) => {
  if (req.url === '/') {
    res.write(JSON.stringify([1, 4, 5, 6]));
    res.end();
  }
});
server.listen(3000);
console.log('server started at 3000....');
```

# Exercise

---

Create a web server which should return dummy articles from url

- Localhost:3000/api/articles

# Convert JSON to string

---

```
var arr = [ "John", "Peter", "Sally", "Jane" ];  
var myJSON = JSON.stringify(arr);
```

# Convert String to JSON

---

```
var text = '{ "name":"John", "birth":"1986-12-14", "city":"New York"}';  
var obj = JSON.parse(text);  
obj.birth = new Date(obj.birth);
```