Introduction to Node.js

11 - 12 January 2020 @ SIT, KMUTT

Slide

http://bit.ly/sit-node-workshop

 **Quick** Recap About **Web**

Front-end VS **Back-end**

Source: https://www.developerkafasi.com/frontend-vs-backend/

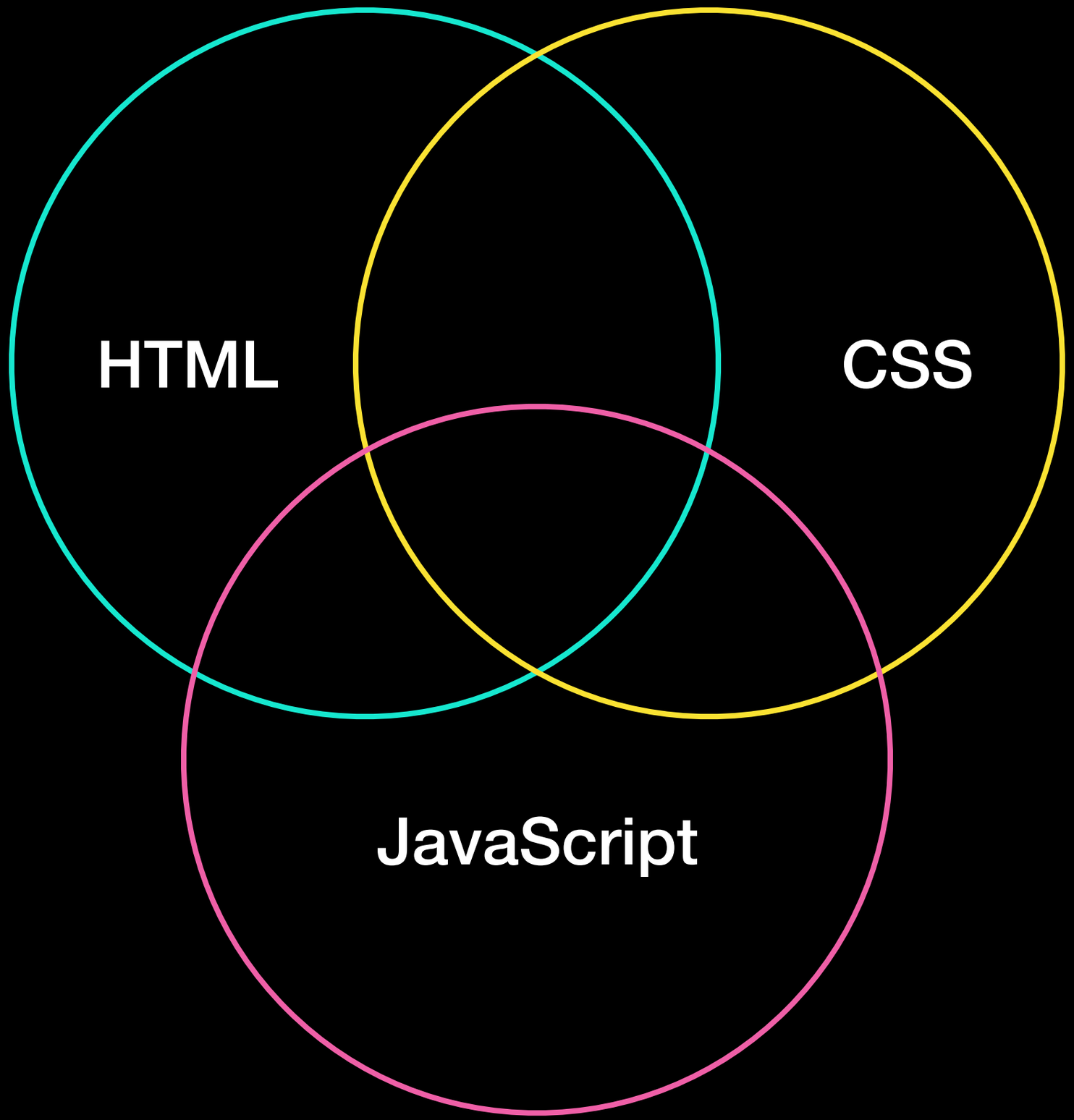
Front-end

User Interface
Look good and emphasise brand

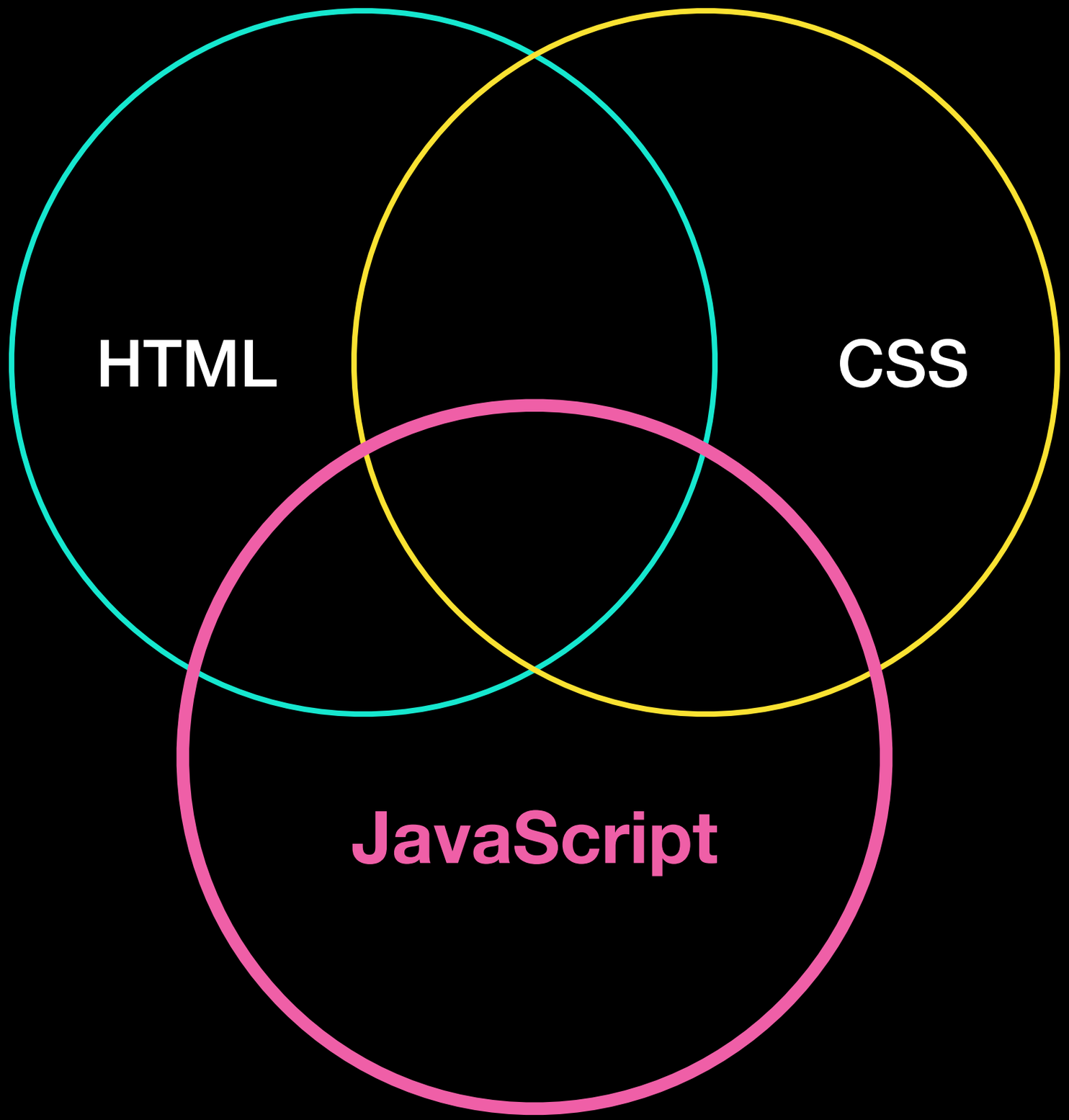
+

User eXperience
Ease of use and feel natural

Front-end



Front-end



Back-end

Business Logic

+

Operating with Database

Back-end

PHP

Go

Java

Node.js

Python

ASP.NET

Back-end

PHP

Go

Java

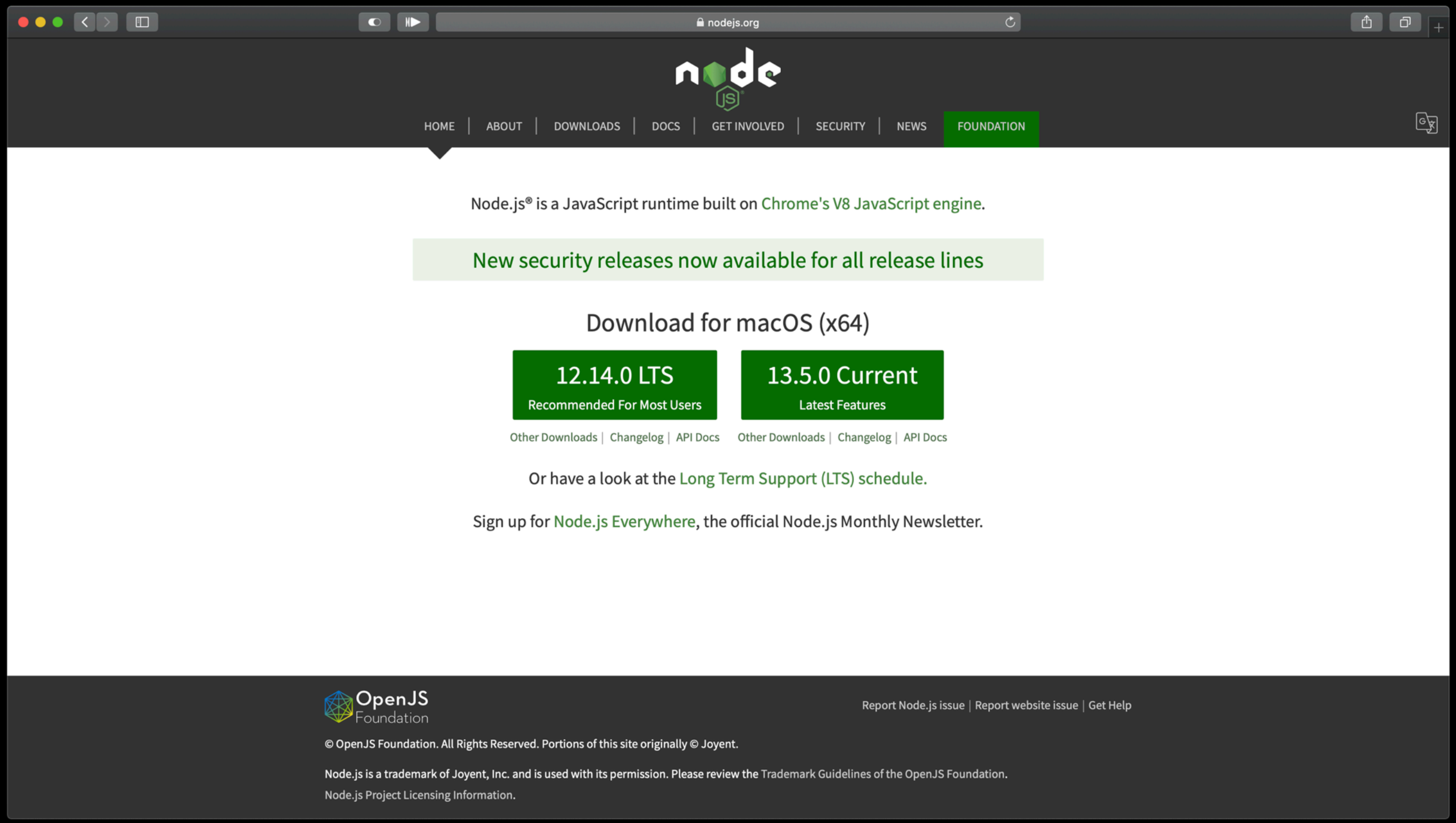
Node.js

Python

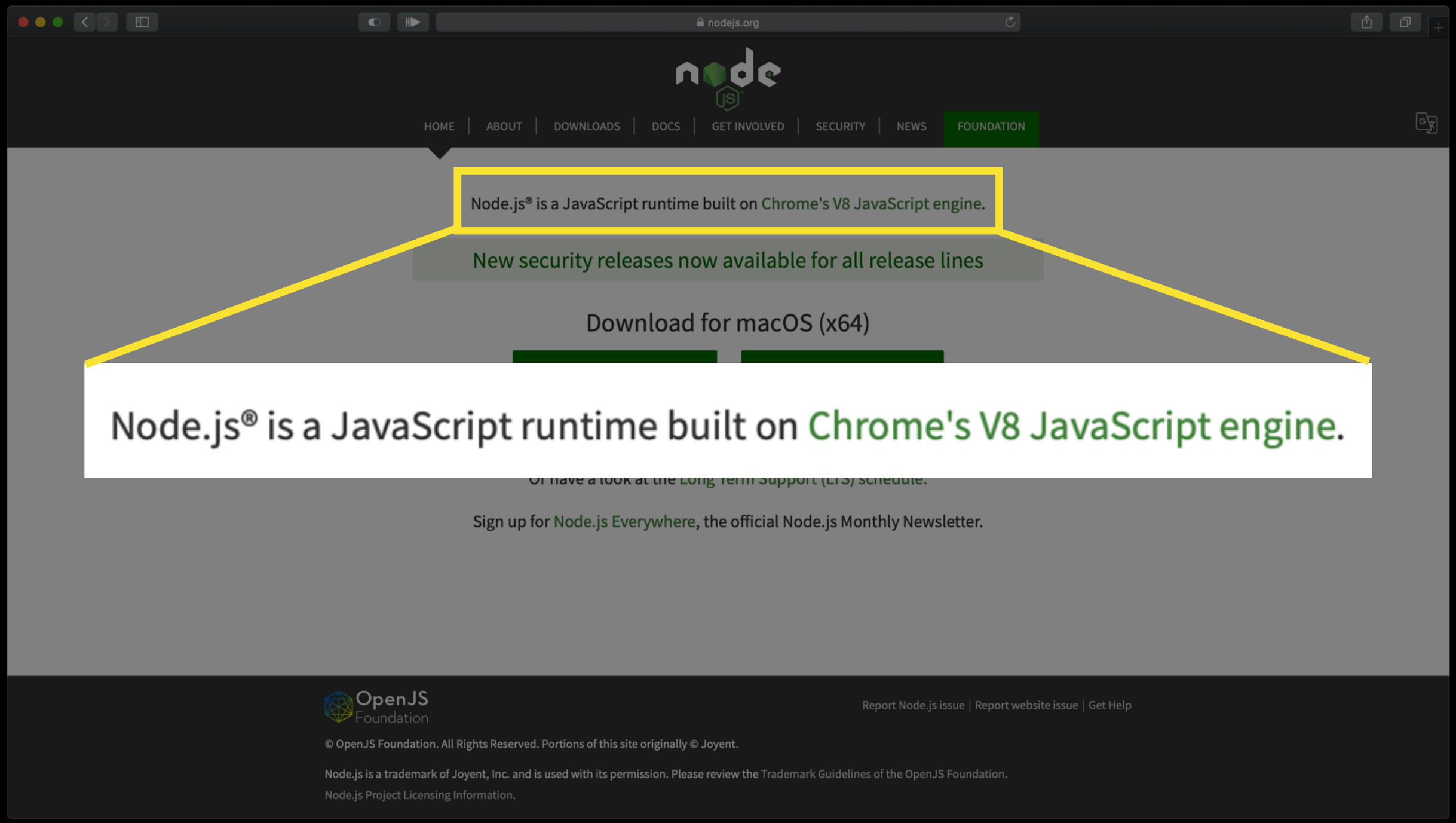
ASP.NET

 Introduction to **Node.js**

What is Node.js?



Source: https://nodejs.org/en/

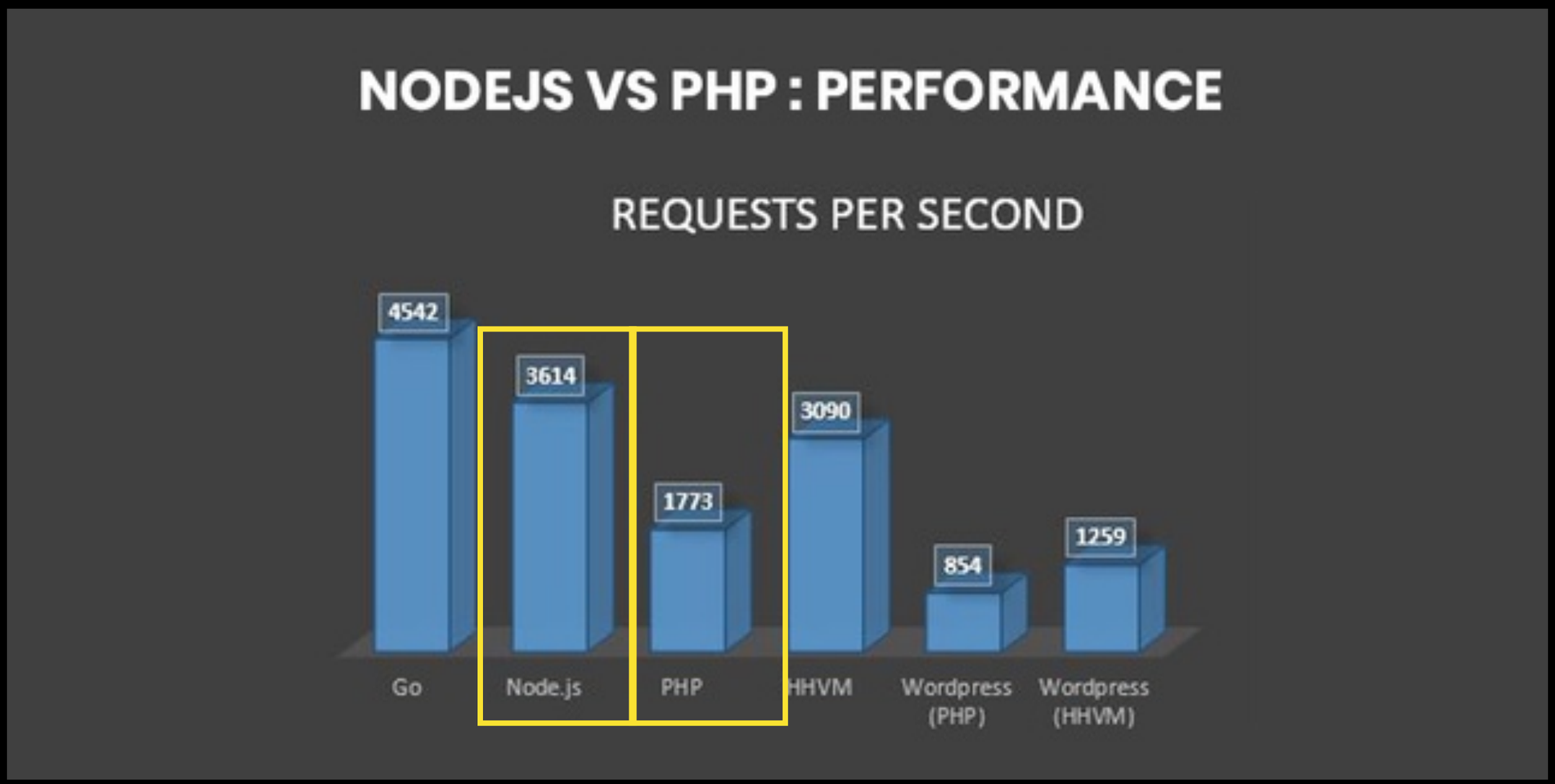


Source: https://nodejs.org/en/

JavaScript **Runtime**

Why Node.js?

NODEJS VS PHP : PERFORMANCE

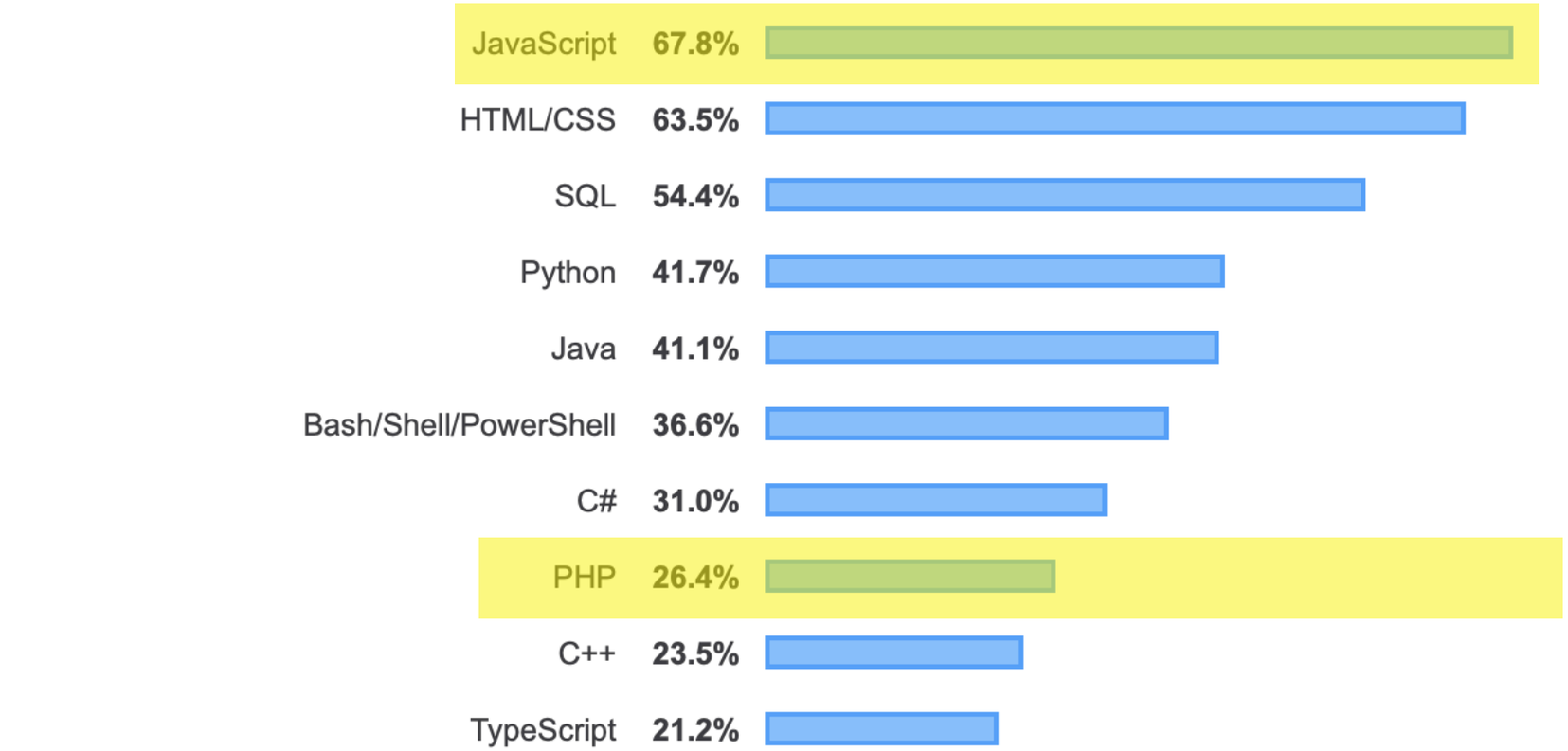


Source: https://hackernoon.com/nodejs-vs-php-which-is-better-for-your-web-development-he7oa24wp

Most Popular Technologies

Programming, Scripting, and Markup Languages

All Respondents | Professional Developers



Source: https://insights.stackoverflow.com/survey/2019

Introduction to **JavaScript**

What is JavaScript?

Programming language to manipulate **DOM** in the browser*

*Originally

What is DOM?

Document **O**bject **M**odel - kind of a **tree**, data structure, to represent **HTML Element**

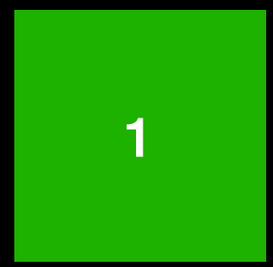
JavaScript: **Syntax**

Data Types

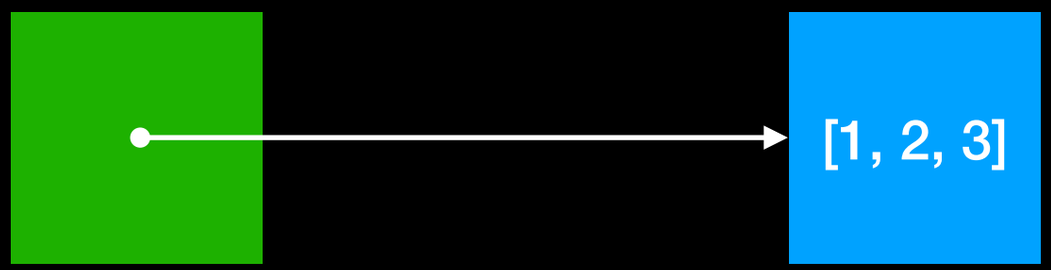
Primitive	Boolean	true, false
	Number	12, 34, 213.32
	String	‘123”, ‘Hello’
	undefined	
Reference	Array	[1, 3, ‘Hello’]
	Object	{ name: ‘Bob’, age: 12 }

Data Types

Primitive



Reference



Variables

```
var abc = 123;
```

Arithmetic Operators

```
Addition: 1 + 2 // 3
Subtraction: 1 - 2 // -1
Multiplication: 1 * 2 // 2
Division: 1 / 2 // 0
Modulus: 1 % 2 // 1

let a = 3;
Increment: a++ // 4
Decrement: a-- // 3
```

Assignment Operators

=

+=

-=

*=

/=

%=

Debugging

console.log(‘A tools to help you debug’**)**

Comment

```
// single line comment
/* multi-line comment */
```

Condition

```
if(condition/boolean) {
    // statement
}
```

Truth Table: AND

AND	TRUE	FALSE
TRUE	**TRUE**	FALSE
FALSE	FALSE	FALSE

Truth Table: OR

OR	TRUE	FALSE
TRUE	TRUE	TRUE
FALSE	TRUE	**FALSE**

Truth Table: NOT

NOT TRUE = FALSE

NOT FALSE = TRUE

Logical Operators

AND: &&

OR: ||

NOT: !

Comparison Operators

Equality: ==
Strict Equality: ===
Greater Than: >
Greater Than or Equal: >=
Less Than: <
Less Than or Equal: <=

Truthy/Falsy

Truthy	true	{ }
	‘0’	function ( ) { }
	“false”	
	[ ]	
Falsy	false	null
	0	undefined
	‘’	NaN
	“”	

Loop

```
for (var i=0; i<10; i++) {
    // statement
}
```

Loop

break;

continue;

Loop

```
while (true) {
    // statement
}
```

Loop

```
let obj = {name: ‘Bob’, age: 12};
for (var key in obj) {
    // statement
}
```

Function

```
function (param1, param2) {
    // statement
    return 1;
}
```

JSON

JSON.parse(jsonString**)**

JSON.stringify(obj**)**

ES2015+

Variables - Block Scope

```
let abc = 123;
```

Variables - Constant

```
const abc = 123;
```

Exponetiation

**

Arrow Function

```
const f1 = (param1) => {
    // statement
}
```

Spread/Rest Operator

```
[...arr1]
```

```
{...obj1}
```

Array Destructuring

```
const arr1 = [1, 2, 3];
let [a, b, c] = arr1;
console.log(b); // 2
```

Object Destructuring

```
const obj1 = {name: ‘abc’ , age: 2};
let {age, name} = obj1;
console.log(name); // ‘abc’
```

Loop

```
let obj = {name: ‘Bob’, age: 12};
for (let value of obj) {
    // statement
}
```

Array Function - forEach

```
[1, 2, 3].forEach(i => console.log(i))
```

Array Function - map

```
[1, 2, 3].map(i => console.log(i))
```

Array Function - filter

```
[1, 2, 3].filter(i => i > 5)
```

Array Function - reduce

```
[1, 2, 3].reduce((prev, i) => prev += i, 0)
```

Class

```
class Person {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
}
```

🎁 Introduction to **Node.js**

Hello World

📁 app.js

```
console.log('Hello World')
```

```
$ node app.js
```

🌳 Node.js: **Module System**

Require

```
require(‘node module’)
require(‘third party package’)
require(‘./folder/filename’)
```

Synchronous Programming

VS

Asynchronous Programming

Synchronous

```
function syncWork() {
    console.log(‘2’)
}
console.log(‘1’)

syncWork();

console.log(‘3’)
```

Synchronous

```
function syncWork() {
    console.log(‘2’)
}
console.log(‘1’)                    // ‘1’

syncWork();

console.log(‘3’)
```

Synchronous

```
function syncWork() {
    console.log(‘2’)
}
console.log(‘1’)                    // ‘1’

syncWork();                         // ‘2’

console.log(‘3’)
```

Synchronous

```
function syncWork() {
    console.log(‘2’)
}
console.log(‘1’)                    // ‘1’

syncWork();                         // ‘2’

console.log(‘3’)                    // ‘3’
```

Asynchronous

```
function asyncWork() {
    setTimeout(() => {
        console.log(‘2’)
    }, 1000)
}
console.log(‘1’)

syncWork();

console.log(‘3’)
```

Asynchronous

```
function asyncWork() {
    setTimeout(() => {
        console.log(‘2’)
    }, 1000)
}
console.log(‘1’)                    // ‘1’

syncWork();

console.log(‘3’)
```

Asynchronous

```
function asyncWork() {
    setTimeout(() => {
        console.log(‘2’)
    }, 1000)
}
console.log(‘1’)                    // ‘1’

syncWork();                         // ‘3’

console.log(‘3’)
```

Asynchronous

```
function asyncWork() {
    setTimeout(() => {
        console.log(‘2’)
    }, 1000)
}
console.log(‘1’)                    // ‘1’

syncWork();                         // ‘3’

console.log(‘3’)                    // ‘2’
```

JavaScript: **Fetch API**

```
fetch(‘https://jsonplaceholder.typicode.com/posts')
```

JavaScript: **Promise**

```
fetch('https://jsonplaceholder.typicode.com/posts')
.then(res => res.json())
.then(data => console.log(data))
.catch(err => console.log(err))
```

JavaScript: **Async/Await**

```
const asyncF1 = async () => {
    try {
        const response = await fetch('https://
jsonplaceholder.typicode.com/posts');
        const data = await response.json();
        return data;
    } catch (error) {
        console.log(error);
    }
}
```

🌳 Node.js: **File System**

Write File

```
const fs = require(‘fs’);
fs.writeFileSync(‘test.txt’, ‘Hello World’);
```

Read File

```
const fs = require('fs');
const str = fs.readFileSync('test.txt');
```

Node.js: **Path**

__dirname

```
console.log(__dirname);
```

Join

```
const path = require(‘path’);
path.join(__dirname, ‘..’, ‘public’,);
```

🌳 Node.js: **Node Package Manager** (NPM)

Basic Command

$ npm init
$ npm init -y
$ npm install
$ npm install <*package_name*>
$ npm install --save-dev <*package_name*>
$ npm install --global <*package_name*>

🌳 Node.js: **nodemon**

Automatic restart server

when file change

$ npm install --global ***nodemon***

Automatic restart server

when file change

$ ***nodemon*** app.js

🌳 Node.js: **Command Line Arguments**

Access arguments

process.argv

Node.js: Easier Command Line Arguments with **Yargs**

Install Yargs

```
$ npm i yargs
```

Config Yargs

```
const yargs = require(‘yargs’);
yargs.command({
  command: ‘c’,
  describe: ‘cccc’,
  ////////////////////////////////////
  handler(argv) {
    // statement
  }
});
yargs.argv;
```

```
builder: {
  a: {
      describe: 'aaaa’,
      demandOption: true,
      type: ‘string'
    }
},
```

🌳 Node.js: Colorful Command Line with **Chalk**

Install Chalk

```
$ npm i chalk
```

Config Chalk

```
const chalk = require('chalk');
console.log(chalk.blue.bgRed.inverse('Hello'))
```

🎁 Introduction to **Version Control**

Track you **change**

No more…

- work.zip
- work_final.zip
- work_final_latest.zip
- work_final_latest_finished.zip
- .
- .
- .

Keywords

Git

Staged

Commit

Repository

Local Repository

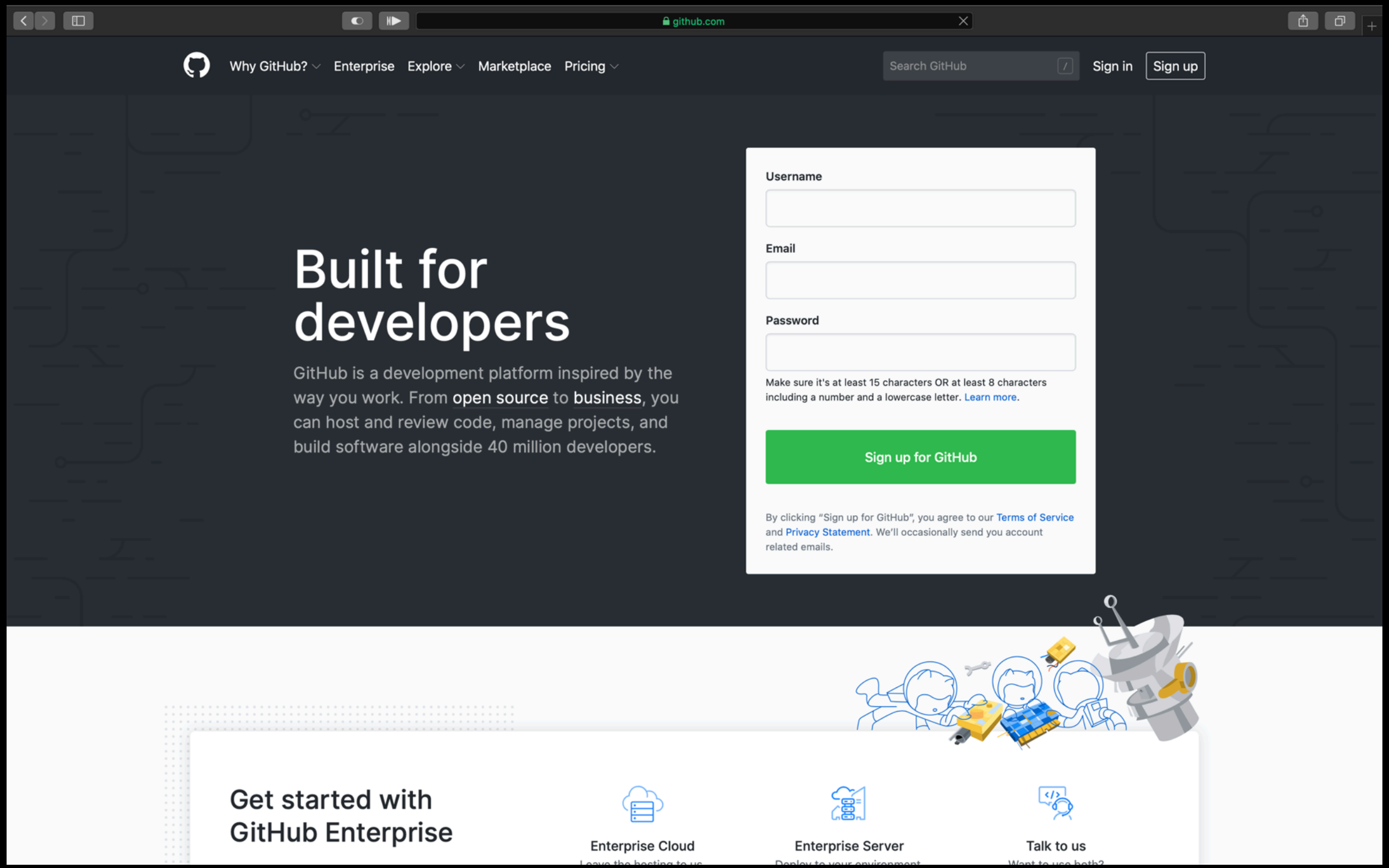
Remote Repository

Very Basic Workflow

1. Work on feature
2. $ git add . # Add every change to staged
3. $ git commit -m “Message”
4. $ git push # Push local change to server

🎁 Introduction to **Github**

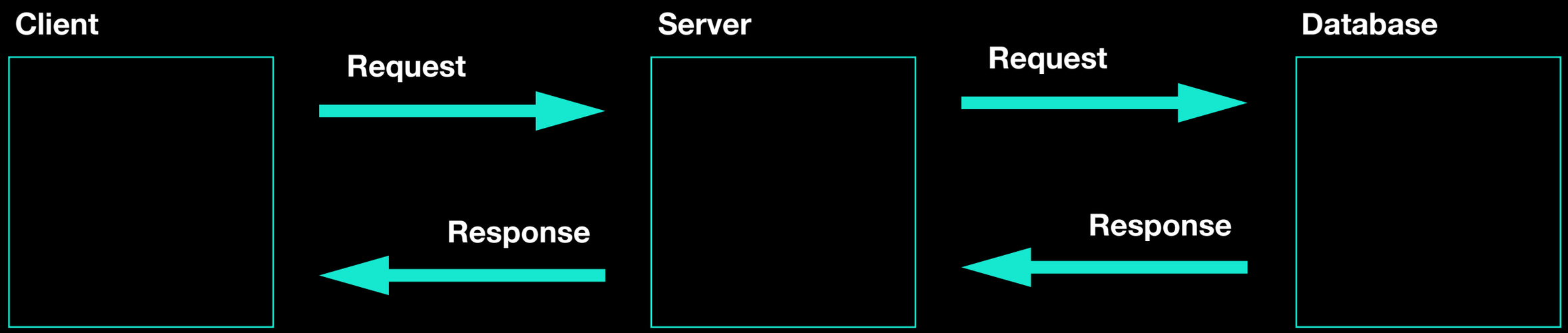
Remote Repository

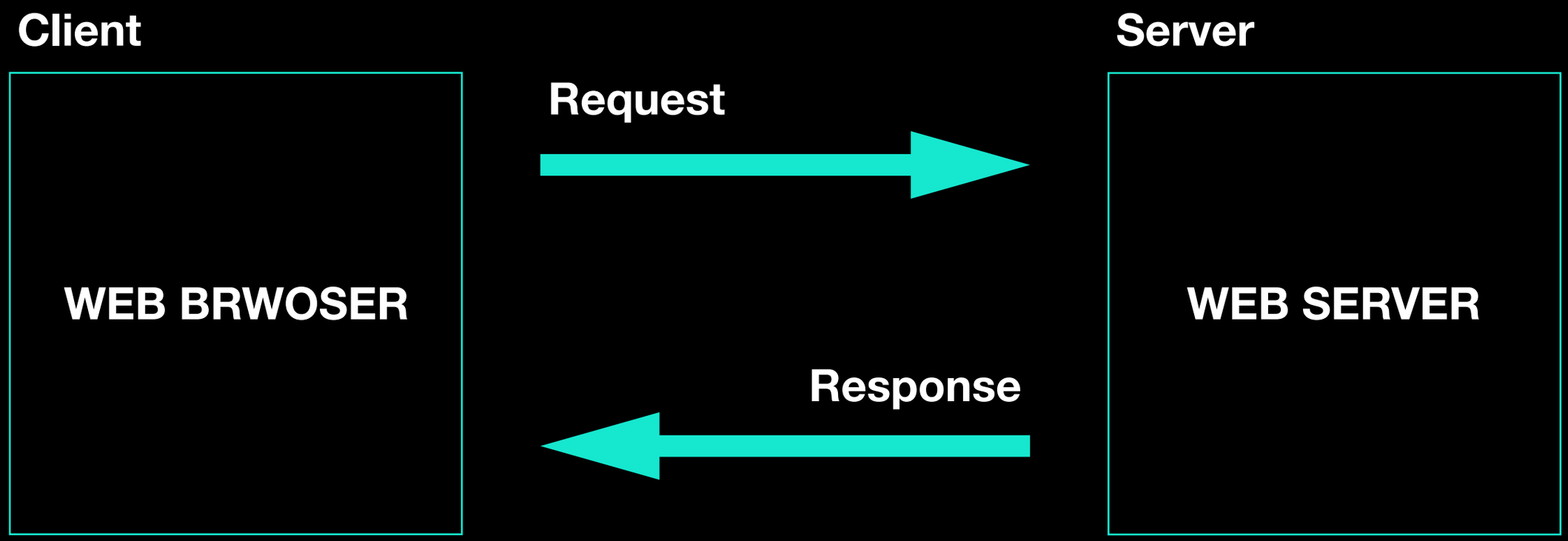

github.com
Why GitHub?
Enterprise
Explore
Marketplace
Pricing
Search GitHub
/
Sign in
Sign up
Built for developers
GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 40 million developers.
Username
Email
Password
Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. Learn more.
Sign up for GitHub
By clicking "Sign up for GitHub", you agree to our Terms of Service and Privacy Statement. We'll occasionally send you account related emails.
Get started with GitHub Enterprise
Enterprise Cloud
Enterprise Server
Talk to us

github.com

🧰 Workshop: **Todo List Application (Command Line)**

 **Quick** Recap About **Web Architecture**


Client
Request
Server
Request
Database
Response
Response


Client
Server
Request
WEB BRWOSER
WEB SERVER
Response

 Introduction to **ExpressJS**

```
var http = require('http');
var fs = require('fs');
var path = require('path');

http.createServer(function (request, response) {
    console.log('request ', request.url);

    var filePath = '.' + request.url;
    if (filePath == './') {
        filePath = './index.html';
    }

    var extname = String(path.extname(filePath)).toLowerCase();
    var mimeTypes = {
        '.html': 'text/html',
        '.js': 'text/javascript',
        '.css': 'text/css',
        '.json': 'application/json',
        '.png': 'image/png',
        '.jpg': 'image/jpg',
        '.gif': 'image/gif',
        '.svg': 'image/svg+xml',
        '.wav': 'audio/wav',
        '.mp4': 'video/mp4',
        '.woff': 'application/font-woff',
        '.ttf': 'application/font-ttf',
        '.eot': 'application/vnd.ms-fontobject',
        '.otf': 'application/font-otf',
        '.wasm': 'application/wasm'
    };

    var contentType = mimeTypes[extname] || 'application/octet-stream';

    fs.readFile(filePath, function(error, content) {
        if (error) {
            if(error.code == 'ENOENT') {
                fs.readFile('./404.html', function(error, content) {
                    response.writeHead(404, { 'Content-Type': 'text/html' });
                    response.end(content, 'utf-8');
                });
            }
            else {
                response.writeHead(500);
                response.end('Sorry, check with the site admin for error: '+error.code+' ..\n');
            }
        }
        else {
            response.writeHead(200, { 'Content-Type': contentType });
            response.end(content, 'utf-8');
        }
    });

}).listen(3000);
console.log('Server running at http://127.0.0.1:3000/');
```

```
const express = require(‘express’);
const app = express();

app.get(‘/‘, (req, res) => res.send(‘Hello World’))

app.listen(3000, () => console.log(‘Server running at http://127.0.0.1:3000/'))
```

$ npm I express

Initialization

```
const express = require(‘express’);
const app = express();
```

Middleware

```
app.get(‘/‘, (req, res) => res.send(‘Hello World’))
```

Start listening

```
app.listen(3000, () => console.log(‘Server running at http://127.0.0.1:3000/'))
```

🌎 ExpressJS: **Routing**

HTTP Verbs

GET	Get data
POST	Create data
PUT	Update data with a new one
PATCH	Update data with some new fields
DELETE	Delete data

GET	app.get()
POST	app.post()
PUT	app.put()
PATCH	app.patch()
DELETE	app.delete()
	app.use()

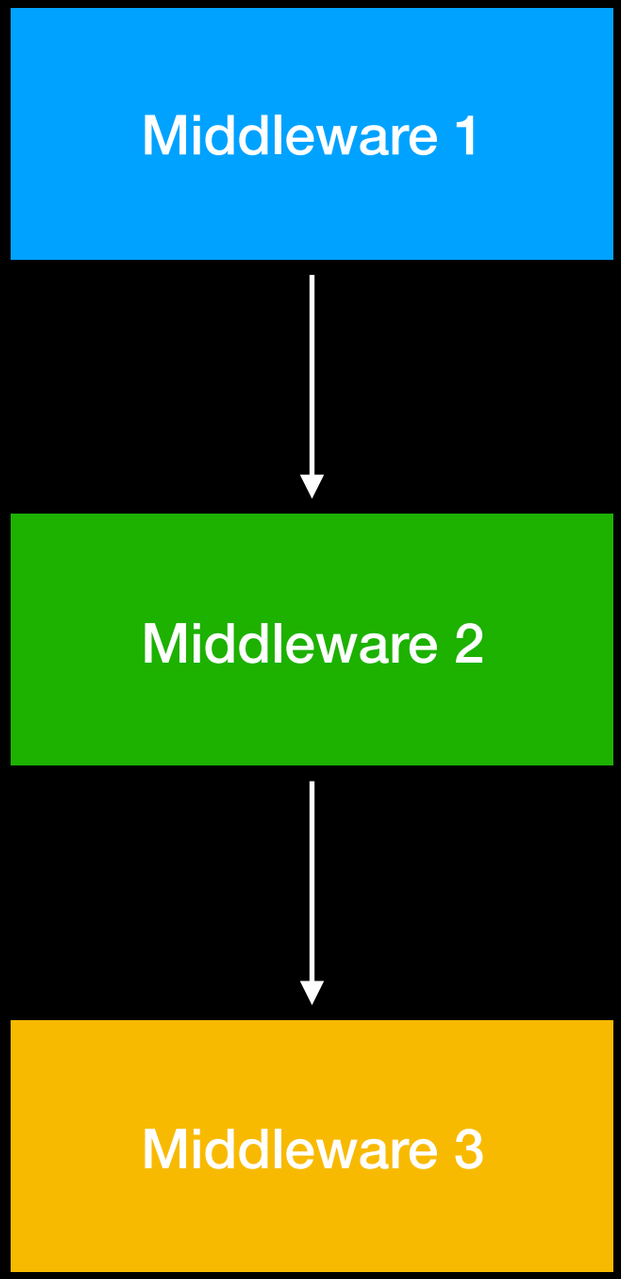
```
app.get(‘/‘, (req, res, next) => {
    // Handle it
})
```

res.send()

res.json()

res.sendFile()

 ExpressJS: **Middleware**


Middleware 1
Middleware 2
Middleware 3

next()

🌎 ExpressJS: **Static File**

```
app.use(express.static(
    path.join(__dirname, ‘public’)
))
```

JSON: **J**ava**S**cript **O**bject **N**otation

```
[
    {
        “name”: “Bob”,
        “age”: 12
    }
]
```

JSON

```
JSON.parse(jsonString)
```

```
JSON.stringify(obj)
```

🌎 ExpressJS: **body-parser**

```
$ npm i body-parser

const bodyParser = require('body-parser');
// application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({
    extended: false
}));
// application/json
app.use(bodyParser.json())
```

What is **Template Engine**?

Static **template** file with HTML and special syntax

🌎 ExpressJS: **EJS**

<% 'Scriptlet' tag, for control-flow, no output
<%= Outputs the value into the template (HTML escaped)
<%- Outputs the unescaped value into the template
%> Plain ending tag

 ExpressJS: **Working with Template Engine**

```
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
```

```
res.render(‘’, {})
```

🧰 Workshop: **Todo List Application (Web)**

🎁 Introduction to **Database**

Permanent collection of data

SQL VS **NoSQL**

Imagine it like **table**

Introduction to **SQL**

Language to manipulate data in database

Retrieve Data

SELECT <*column_name*>
FROM <*table*>
WHERE <*condition*>

Condition

= Equal

<> Not Equal

LIKE _% Check String Pattern

Add New Data

INSERT INTO <*table*>
VALUES (<*value_list*>*)*

Update Existing Data

UPDATE <*table*>
SET <*column_name*> = <*value*>
WHERE <*condition*>

Delete Data

DELETE FROM <*table*>

WHERE <*condition*>

🌳 Node.js: **Working with Database (MySQL)**

Install Required Package

```
$ npm i mysql2
```

Config

```
const mysql = require("mysql2/promise");
const pool = mysql.createPool({
  host: ‘url’,
  port: 3306,
  user: ‘username’,
  password: ‘password’,
  database: ‘database_name’,
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0
});

module.exports = pool;
```

Usage

```
const db = require('./utils/db');
// Async task!!!
const result = await db.query('SQL HERE')
```

🎁 Introduction to **REST API**

Request

1. HTTP Verb
2. Endpoint
3. Payload
4. Headers

Request

GET /
Content-Type: ‘application/json’
{“name”: “Bob”}

HTTP Verbs

GET	Get data
POST	Create data
PUT	Update data with a new one
PATCH	Update data with some new fields
DELETE	Delete data

Endpoint

GET /posts	Get All Posts
GET /posts/:postId	Get Specified Post
POST /posts	Create New Post
PUT /posts/:postId	Update Specified Post
DELETE /posts/:postId	Delete Specified Post

🌳 Node.js: **Make REST API Application**

Node.js server serve as

API server

Front-end communicate through **AJAX** to the correct endpoint

🎁 Introduction to **Environment Variable**

Remove Critical Credentials Out of Your App to Make It **Safes** When It Available On Remote Repository

Also, serve as a **config**

🌳 Node.js: **Working with dotenv**

Install

```
$ npm i dotenv
```

Config

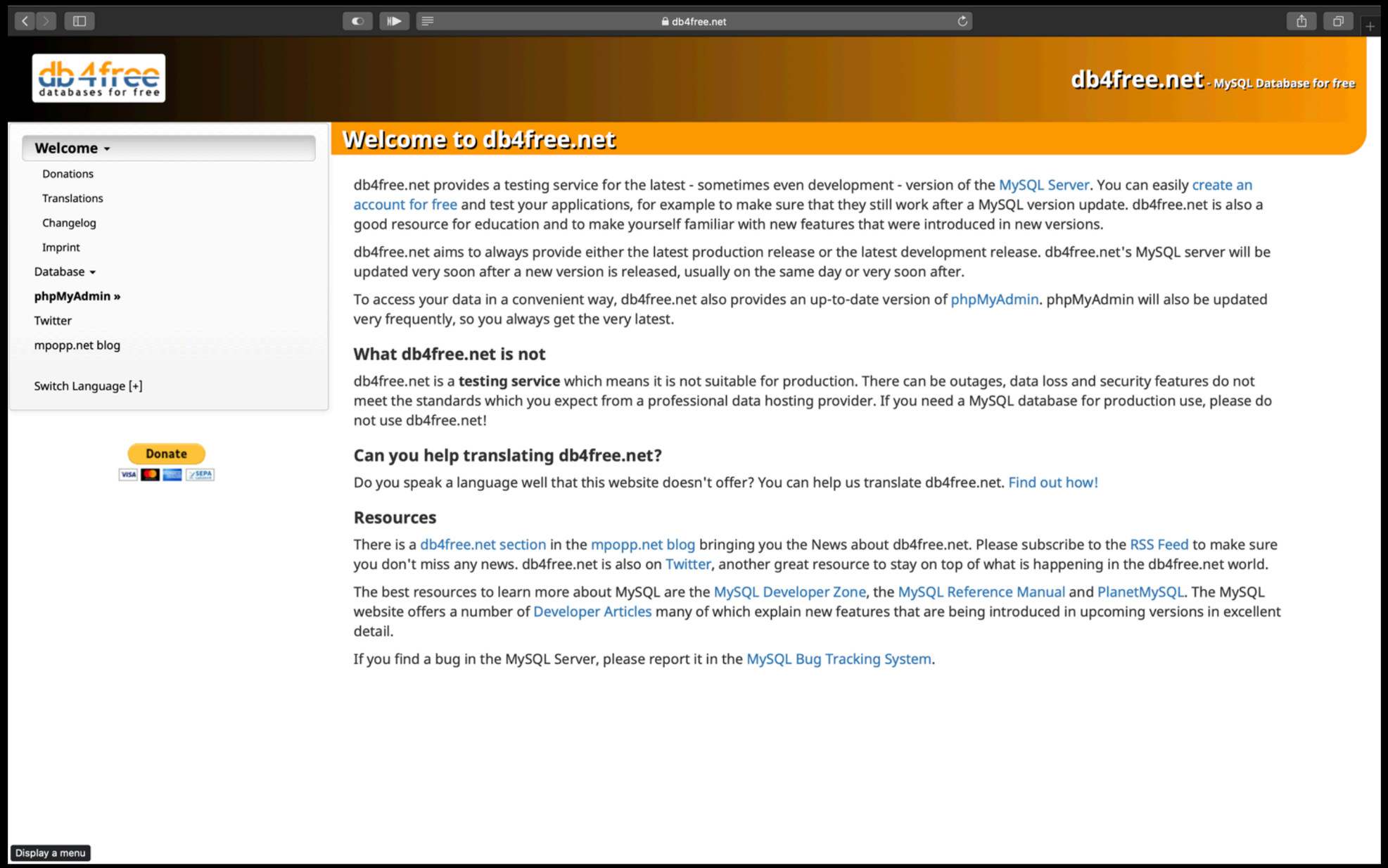
Create file .env

Inside, ___=‘___’

Put require(‘dotenv’).config() above everything in every files needed

Access variable via process.env.___

Register DB4Free


db4free.net
db4free.net - MySQL Database for free
Welcome
Donations
Translations
Changelog
Imprint
Database
phpMyAdmin »
Twitter
mpopp.net blog
Switch Language [+]
Donate
Display a menu
Welcome to db4free.net
db4free.net provides a testing service for the latest - sometimes even development - version of the MySQL Server. You can easily create an account for free and test your applications, for example to make sure that they still work after a MySQL version update. db4free.net is also a good resource for education and to make yourself familiar with new features that were introduced in new versions.
db4free.net aims to always provide either the latest production release or the latest development release. db4free.net's MySQL server will be updated very soon after a new version is released, usually on the same day or very soon after.
To access your data in a convenient way, db4free.net also provides an up-to-date version of phpMyAdmin. phpMyAdmin will also be updated very frequently, so you always get the very latest.
What db4free.net is not
db4free.net is a testing service which means it is not suitable for production. There can be outages, data loss and security features do not meet the standards which you expect from a professional data hosting provider. If you need a MySQL database for production use, please do not use db4free.net!
Can you help translating db4free.net?
Do you speak a language well that this website doesn't offer? You can help us translate db4free.net. Find out how!
Resources
There is a db4free.net section in the mpopp.net blog bringing you the News about db4free.net. Please subscribe to the RSS Feed to make sure you don't miss any news. db4free.net is also on Twitter, another great resource to stay on top of what is happening in the db4free.net world.
The best resources to learn more about MySQL are the MySQL Developer Zone, the MySQL Reference Manual and PlanetMySQL. The MySQL website offers a number of Developer Articles many of which explain new features that are being introduced in upcoming versions in excellent detail.
If you find a bug in the MySQL Server, please report it in the MySQL Bug Tracking System.

db4free.net

🧰 Workshop: **Blog (Web)**

What’s next?

Alternative to ExpressJS

NestJS with TypeScript

More secure type with **TypeScript**

TypeScript as a **Compiler**

Handle **File Upload**

Multer

Handle **Validation**

Express-validation

Handle **Authentication**

Passport

Single Page Application

React

Vue

Angular

MERN/MEAN/MEVN Stack

Angular

MongoDB **Express** **React** **Node.js**

Vue

Real-time Web Application

Socket.io

Q & A