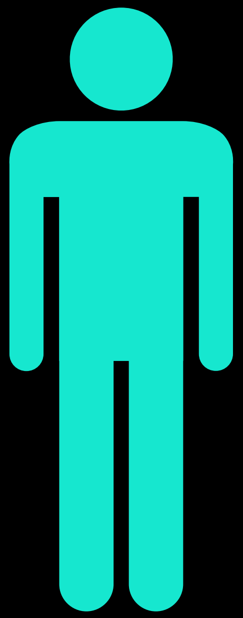
🚩 Web Development 🚩
🎖️ Crash Course 🎖️

🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥

① HTML

Hyper
Text
Markup
Language

HTML =



Tag

```
<name></name>
```

Tag

```
<name>content</name>
```

Tag

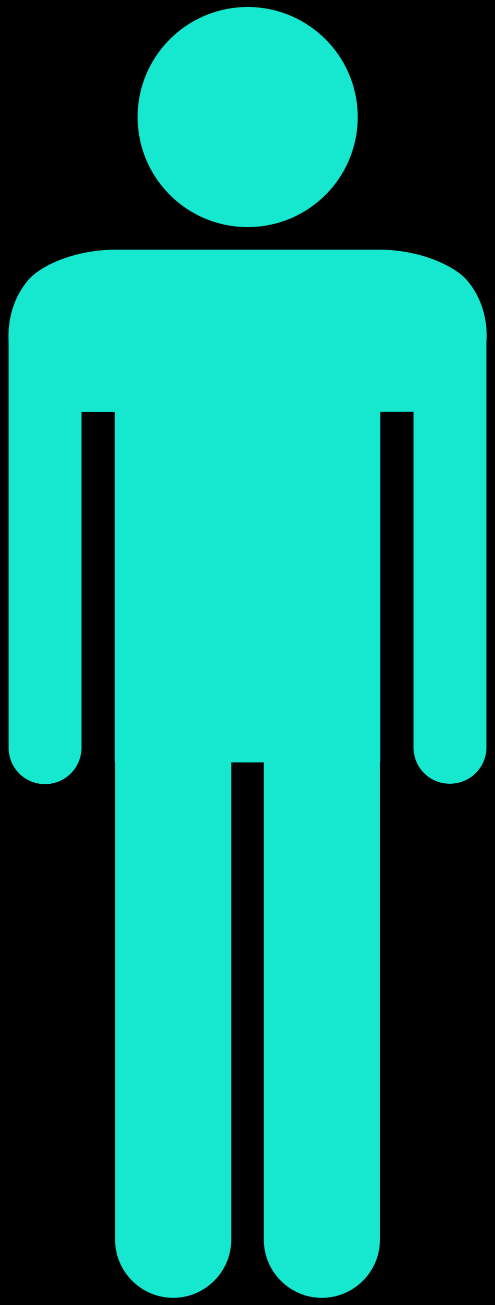
```
<name attribute=“value”>
    content
</name>
```

Self-closing Tag

```
<name attribute=“value” />
```

Structure of Webpage

```
<!DOCTYPE html>
<html>
    <head>
        . . .
    </head>
    <body>
        . . .
    </body>
</html>
```



Structure of Webpage

```
<!DOCTYPE html>
<html>
    <head>
        . . .
    </head>
    <body>
        . . .
    </body>
</html>
```

Structure of Webpage

```
<!DOCTYPE html>
<html>
    <head>
        . . .
    </head>
    <body>
        . . .
    </body>
</html>
```

Structure of Webpage

```
<!DOCTYPE html>
<html>
    <head>
        . . .
    </head>
    <body>
        . . .
    </body>
</html>
```

Structure of Webpage

```
<!DOCTYPE html>
<html>
    <head>
        . . .
    </head>
    <body>
        . . .
    </body>
</html>
```

Metadata Tag

```
<title>
    Name of the Webpage
</title>
```

Metadata Tag

```
<meta attribute="value" />
      charset="utf-8"
      name="???" content="???"
```

Metadata Tag

```
<link rel="" type="" href="" />
```

Text - Heading

```
<h1></h1>

<h2></h2>

<h3></h3>
<h4></h4>

<h5></h5>

<h6></h6>
```

Text - Content

```
<p></p>
```

Text - List

```
<ol></ol>
<ul></ul>
```

Text - List

```
<ul>
    <li>content</li>
    <li>content</li>
</ul>
```

Text - Emphasis

```
<em></em>
<strong></strong>
<strike></strike>
<sub></sub>
<sup></sup>
```

Hyperlink

```
<a href=“” title=“”></a>
```

Absolute Path VS Relative Path

`URL: https://sit.kmutt.ac.th/page1`

/ - Absolute Path

```
/folder1/file.png

= https://sit.kmutt.ac.th/folder1/
file.png
```

./ - Relative Path

```
./folder1/file.png

= https://sit.kmutt.ac.th/page1/
folder1/file.png
```

Image

```
<img src=“” alt=“” />
```

Structure

```
<header></header>
<nav></nav>
<main></main>
<article></article>
<section></section>
<aside></aside>
<footer></footer>
```

Table

```
<table>
    <thead>
        <th>
            <td>Content</td>
        </th>
    </thead>
    <tbody>
        <tr>
            <td>Content</td>
        </tr>
    </tbody>
</table>
```

General Block

```
<div></div>
```

General Inline

```
<span></span>
```

General attribute

```
<name class=“” id=“”></name>
```

Form

```
<form action=“” method=“”>
    <input type=“” name=“” />
    <button>Submit</button>
</form>
```

Comment

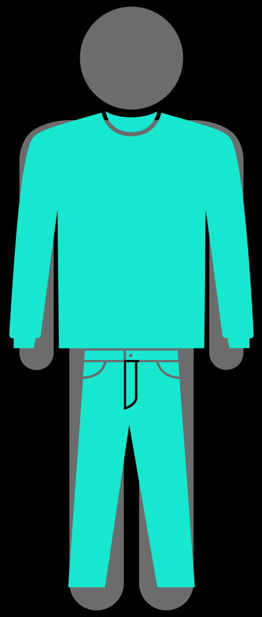
```
<!-- This is comment -->
```

https://htmlreference.io

② CSS

Cascading StyleSheet

CSS =



Selector

```
element {

}
```

Selector

```
element {
    property: value,
    property: value
}
```

CSS in HTML

1. Inline Style
2. Internal Stylesheet
3. External Stylesheet

CSS in HTML - Inline Style

```
<name style=“”></name>
```

CSS in HTML - Internal Stylesheet

```
<style></style>
```

CSS in HTML - External Stylesheet

```
<link rel="stylesheet"
type="text/css" href="" />
```

Selector - HTML Element

```
element {

}
```

Selector - All

```
* {

}
```

Selector - Class

```
.className {

}
```

Selector - Class

```
#id {

}
```

Selector - Many element

```
.class1, #id, element {

}
```

Selector - Descendant

```
.class1 element {

}
```

Selector - Child

```
.class1 > element {

}
```

Selector - Siblings

```
.class1 + element {

}
```

Selector - Attribute

```css
element[attr=value] {

}
```

Selector - Pseudo-class

```
element:hover {

}
```

Selector - Pseudo-element

```
element::before {

}
```

Specificity

Element = 1
Class = 10
ID = 100

Units

px
em
rem
vh
vw
%

Color

```
background
color
```

HEX Code - #123456

Name - Red

RGB - rgb(123,123,123)

RGBA - rgba(123, 123, 123, 123)

Font

```
font-family
font-size
font-style
font-weight
```

sans-serif, serif, monospace

italic

100 - 900, bold

List

```
list-style
list-style-type
```

Box Model

```
padding
margin
```

Positioning

```
text-align
position
```

Absolute VS Relative VS Static

```
position: absolute
position: relative
top
bottom
left
right
```

```
position: static
```

Block VS Inline VS Inline-Block

`display`

Media Query

```
@media (max-width: 1000px) {
    h1 {
        font-size: 30px
    }
}
```

Flexbox

display: flex
flex-direction: ⬆️ column/➡️ row
flex-wrap: wrap/no-wrap
justify-content: flex-start/flex-end/center/space-around/space-between
align-items: flex-start/flex-end/center/stretch

CSS - Comment

```
/*
    This is a comment
*/
```

https://cssreference.io

③ Bootstrap

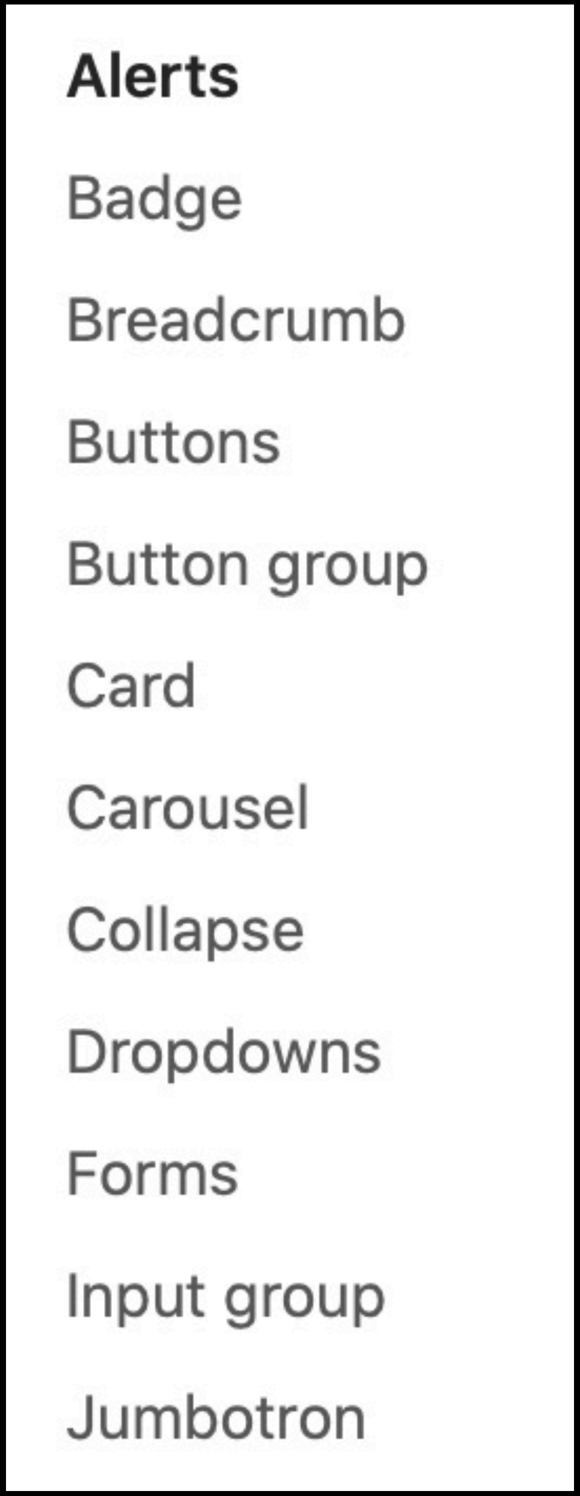
Bootstrap

Open-source component library

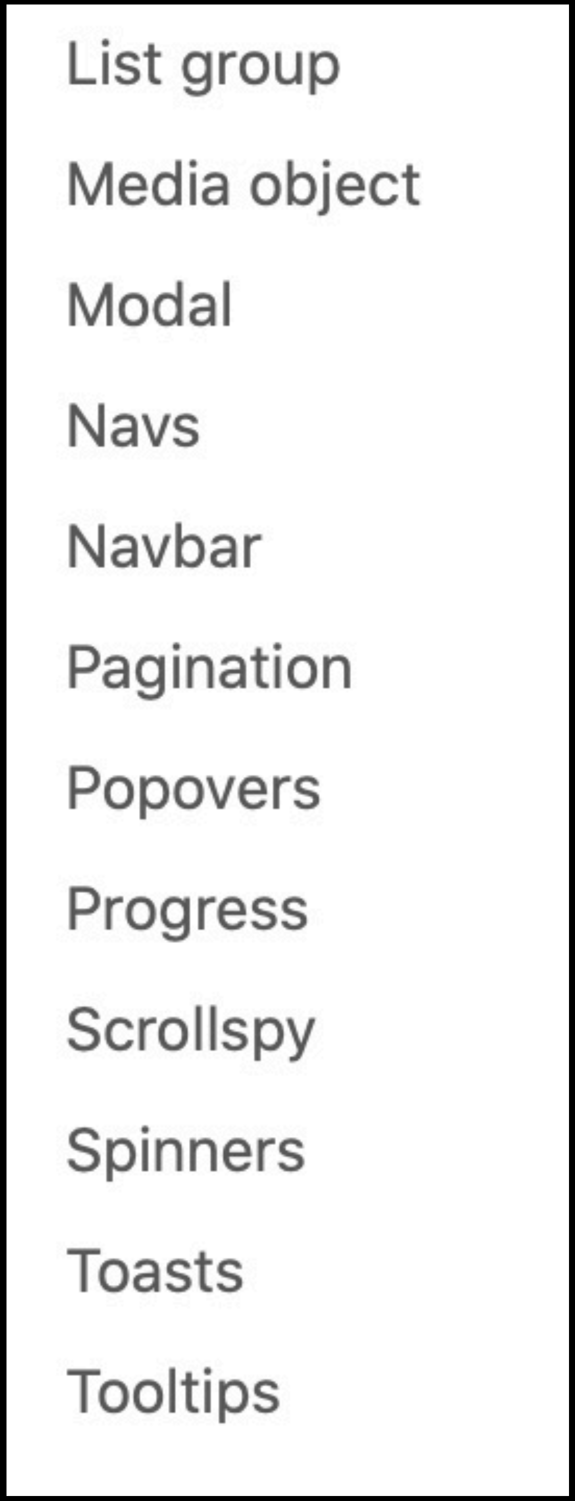
Bootstrap - Layout/Grid

```
.container, .container-fluid
  .row
    .col-xx-xx
```

Bootstrap - Components

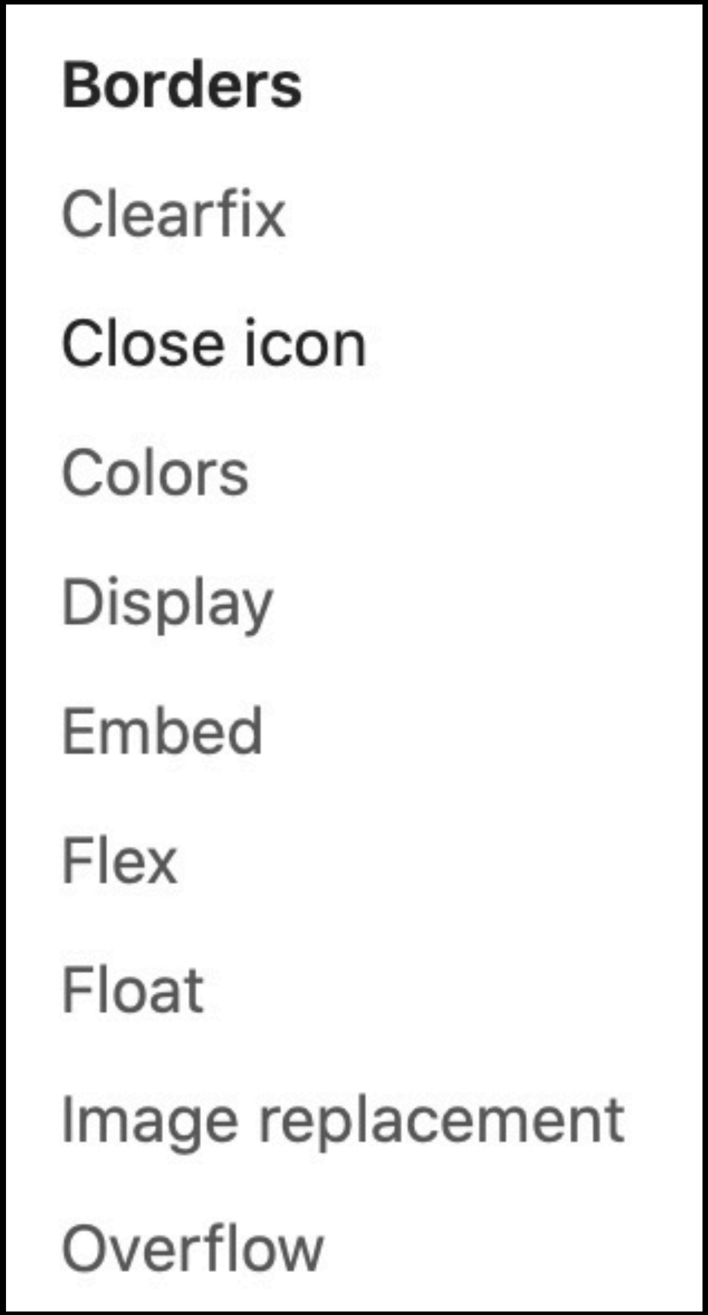


Alerts
Badge
Breadcrumb
Buttons
Button group
Card
Carousel
Collapse
Dropdowns
Forms
Input group
Jumbotron



List group
Media object
Modal
Navs
Navbar
Pagination
Popovers
Progress
Scrollspy
Spinners
Toasts
Tooltips

Bootstrap - Utilities



Borders

Clearfix

Close icon

Colors

Display

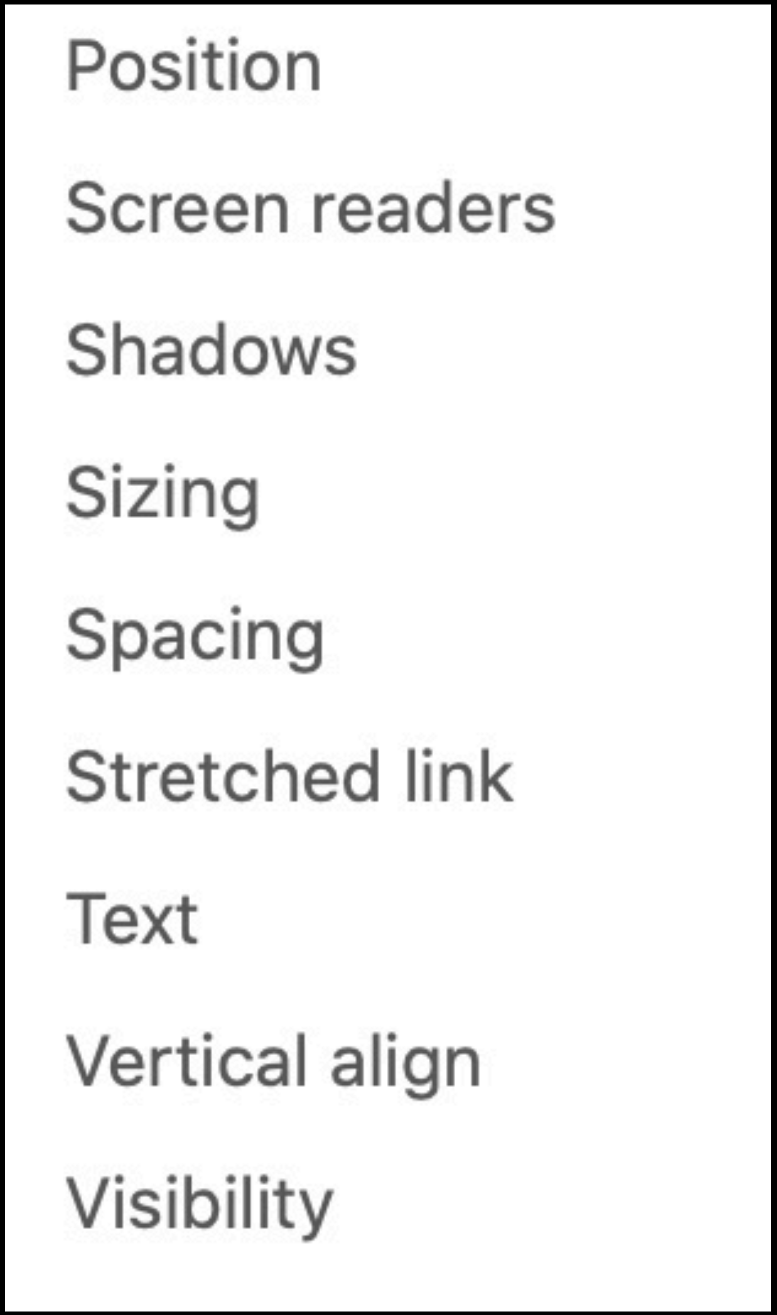
Embed

Flex

Float

Image replacement

Overflow



Position

Screen readers

Shadows

Sizing

Spacing

Stretched link

Text

Vertical align

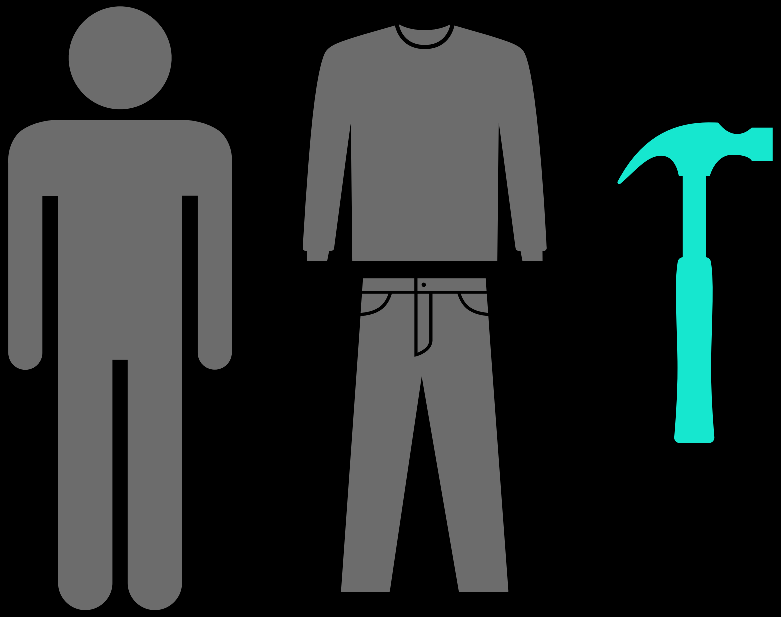
Visibility

Bootstrap - Icon

Font awesome

④ JavaScript

JavaScript =



JavaScript - `console.log`

```
console.log()
```

Naming Convention

https://google.github.io/styleguide/jsguide.html

JavaScript - Variable Declaration

```
var variableName
```

JavaScript - Condition

```
if (condition) {
    // statements
} else if (condition2) {
    // statements
} else {
    // statements
}
```

JavaScript - Comparison (=== vs ==)

=== - Strictly equal
== - Equal

JavaScript - Loop

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

for (var c in arr) {
    // statements
}
```

JavaScript - Array

```
var arr = [1, “Hello World”, 3]
```

JavaScript - String

```
var str = “Hello World”
```

JavaScript - Object

```
var obj = {
    key: value
}
```

JavaScript - Function

```
var func1 = function(parameter) {
    // statements
}

function func2(parameter) {
    // statements
}
```

JavaScript - Comment

```
// This is comment
/*
   This is multiple
   line comment
*/
```

JavaScript with HTML

1. Inline JavaScript
2. Internal JavaScript
3. External JavaScript

JavaScript with HTML - Inline

```
<button onClick=“alert(‘Hello’)”>
    Click Me
</button>
```

JavaScript with HTML - Internal

```
<script>
    alert(‘Hello’)
</script>
```

JavaScript with HTML - External

```
<script src=“”></script>
```

Local Storage

```
localStorage.setItem(
    “name”, “value”
)
localStorage.getItem(“name”)
```

JSON

```
JSON.stringify(obj)
JSON.parse(“JSON String”)
```

setInterval

```
setInterval(function() {
   alert(“Run every 1 second”)
}, 1000)
```

setTimeout

```
setTimeout(function() {
   alert(“1 second passed!”)
}, 1000)
```

DOM API

Representational of Webpage in **Tree**

DOM API - `window`

One of global object that have properties about window (browser)

DOM API - `document`

One of global object that have properties about webpage

DOM API - Element

```
document.getElementById(“id”)
document.querySelector(“css
selector”)
document.querySelectorAll(“css
selector”)
```

DOM API - Element

```
// Get value of the attribute
document.getElementById(“input1”).value;
// Set value of the attribute
document.getElementById(“input1”).value = 1;
```

DOM API - Event Listener

```
document.getElementById(“btn1”)
.addEventListener(“click”, function() {
    alert(“Clicked”);
});
```

⑤ Github

Version Control

No more something like this…

project.V1.pdf

project.V2.pdf

project.VFinal.pdf

project.VFinalFixed.pdf

project.VFinalFixedUpdated.pdf

Git

One of Version Control

Github

Website that run Git

Repository

Place to store code and change

```
$ git init
```

Clone

Download repository from online to machine

```
$ git clone url
```

Staged

File that already added and ready to commit

```
$ git add .
```

Commit

Make a flag that this is one version

```
$ git commit –m “Message”
```

Commit - Convention

[VERB] description without file name

Push/Pull

Push = Upload to Server
Pull = Download from Server

```
$ git push origin master
$ git pull origin master
```

⑥ SQL

CRUD

Create
Read
Update
Delete

SQL INSERT

```
INSERT INTO somewhere
VALUES(values)
```

SQL SELECT Clauses

```
SELECT something
FROM somewhere
WHERE condition
GROUP BY something
HAVING condition
ORDER BY something
```

SQL UPDATE

```
UPDATE somewhere
SET something = value
WHERE condition
```

SQL DELETE

```
DELETE FROM somewhere
WHERE condition
```

⑦ Environment Variables

⑧ Work Flow

⑨ SSH

①⓪ Advanced Github

Branching & Checkout

Like create a duplicate to experiment

```
$ git branch branchName
$ git checkout branchName
```

Merge branch

Merge back to original branch

```
$ git checkout master
$ git merge branchName
```

Merge branch - Conflict

Sometime code cannot go along - FIX IT!

Branch - Convention

master = main branch (stable version)
dev = release candidate for next stable version
feature/name = feature development
hotfix/date = hot fix to fix critical bug

①① ECMAScript 6+

Variable Declaration

```
let variableName
const variableName
```

Arrow Function

```
const f1 = () => {
    // statements
}
```

Default Parameters

```
const f1 = (a = 1) => {
    // statements
}
```

Spread & Rest Operator

```
const f1 = (a, b, c, …others) {
    // statements
}
```

Spread & Rest Operator - Array

```
const arr = [1, 2, 3, 4]
const arr2 = […arr1, 5, 6]
```

Spread & Rest Operator - Object

```
const obj = {
    key: value,
    key2: value2
}

const obj2 = {
    ...obj,
    key3: value3
}
```

Destructuring - Array

```
const arr = [1, 2, 3, 4]
const [a, b] = arr
```

Destructuring - Object

```
const obj = {
    key: value,
    key2: value2
}

const {a: aRenamed, b = 2} = obj
```

Array Function

```
.map()
.forEach()
.sort()
.filter()
.reduce()
```

Template Literals

```
const a = 10
console.log(`Hello, ${a} times`)
```

Class

```
class Person {
    constructor() {
    }
}
```

Asynchronous Programming

```
console.log(“Outer Loop 1”)
setTimeout(() => console.log(
    “Inner Loop”
), 1000)
console.log(“Outer Loop 2”)
```

Promise

```
axios.get(
        “https://api.github.com”
    )
.then(s => JSON.parse(s))
.then(str => console.log(str))
.catch(err => console.error(err))
```

Async/Await

```
async function f1() {
    const s = await axios.get(
        “https://api.github.com”
    )
    console.log(JSON.parse(s))
}
```

Callback

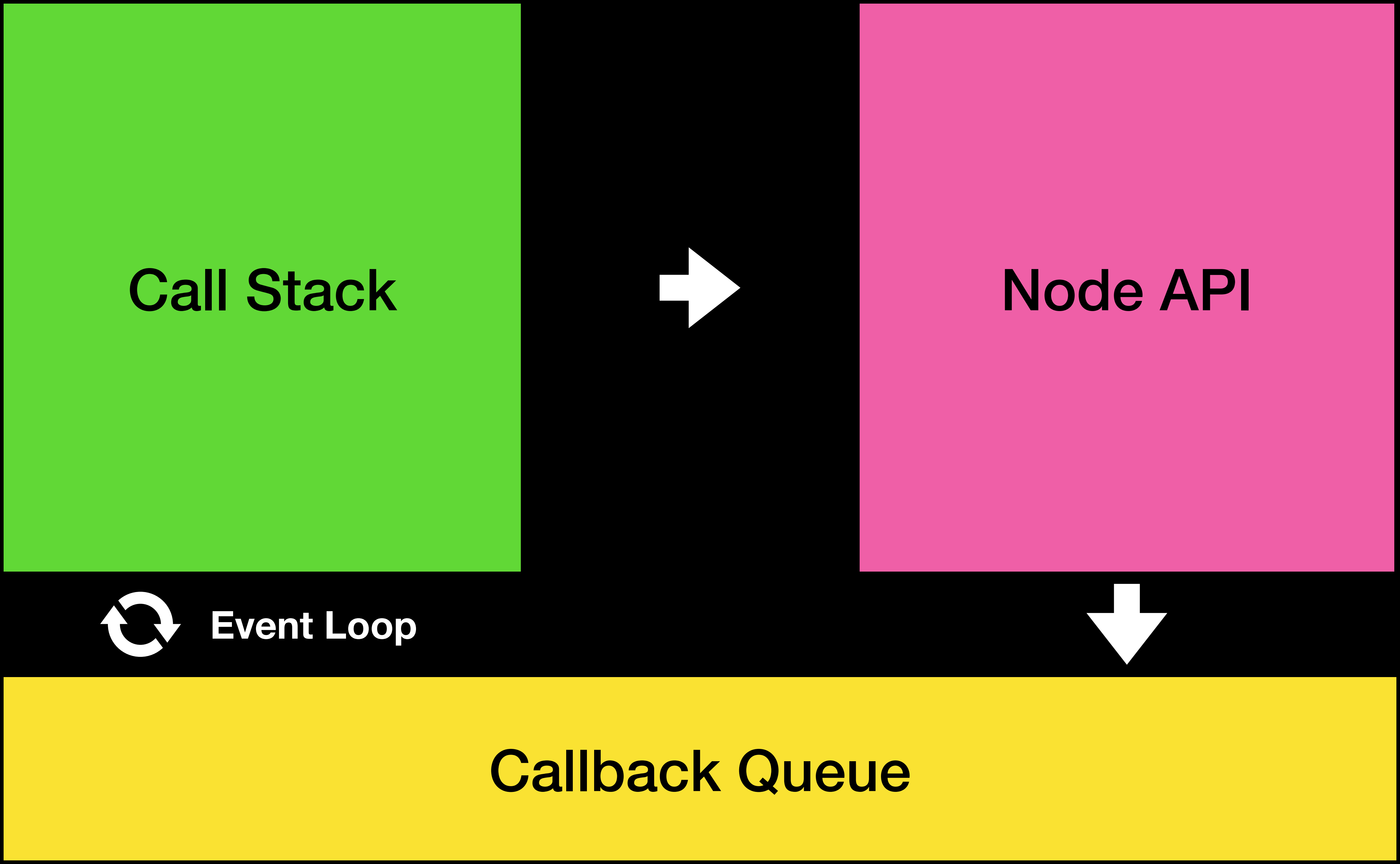
Function that run once the work of the main function finished.

①② NodeJS

NodeJS

Run JavaScript outside web environment

Call Stack & Event Loop



NodeJS - Module System

app.js

```
const fs = require(“fs”)

const utilities = require(“./utils”)
```

utils.js

```
const add = (a, b) => a + b
module.exports = {
   add
}
```

NodeJS - File System

```
const fs = require(“fs”)

fs.writeFile(“test.txt”, data, (err) => {
    if (err) return console.log(err)
    console.log(“File created”)
})

fs.readFile(“test.txt”, (err, data) => {
    if (err) return console.log(err)
    console.log(data)
})
```

Package Manager

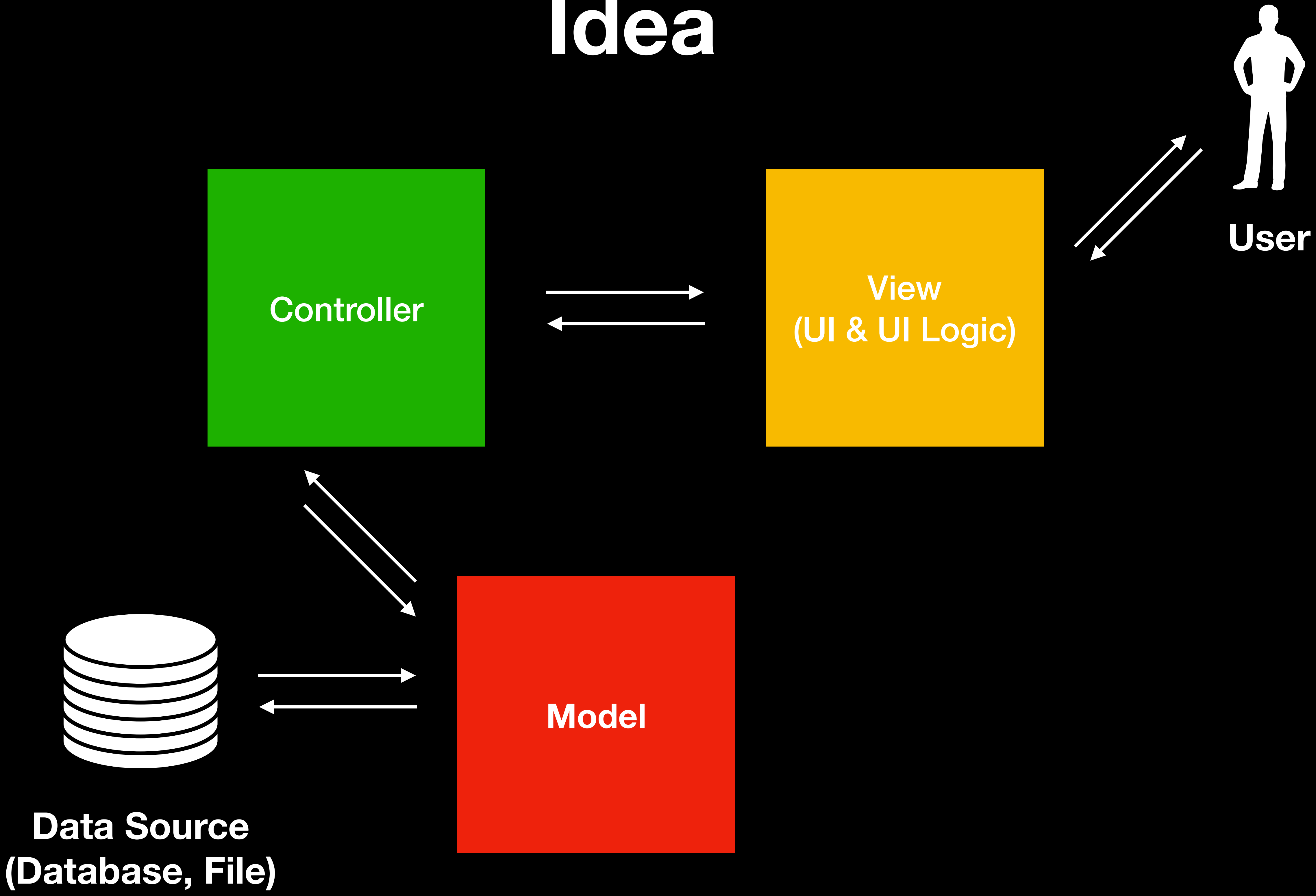
Use other code to help accelerate development

NPM

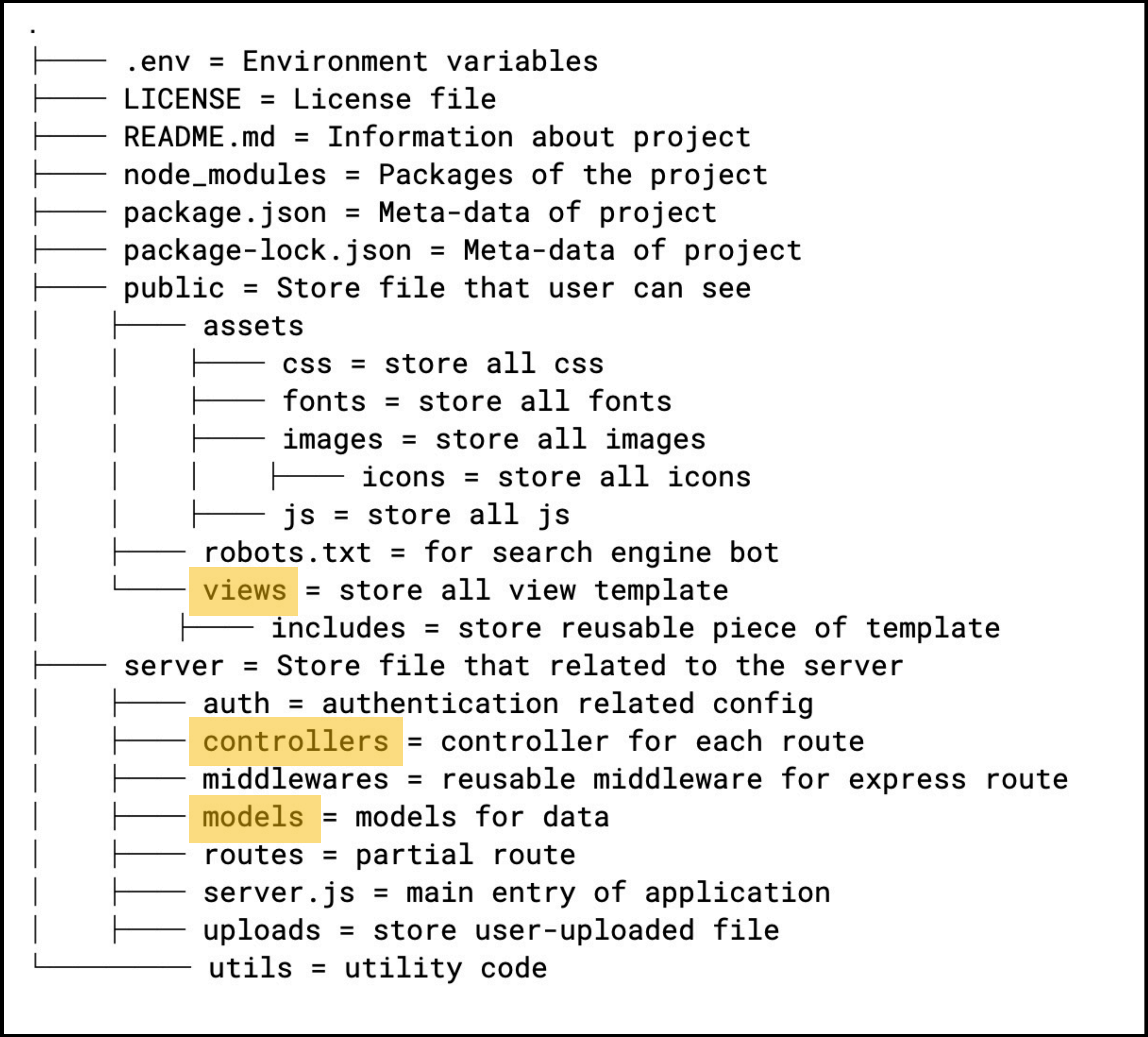
```
$ npm init
$ npm install
$ npm install <package_name>
```

⑬ MVC

Idea


User
Controller
View
(UI & UI Logic)
Model
Data Source
(Database, File)

Project Structure

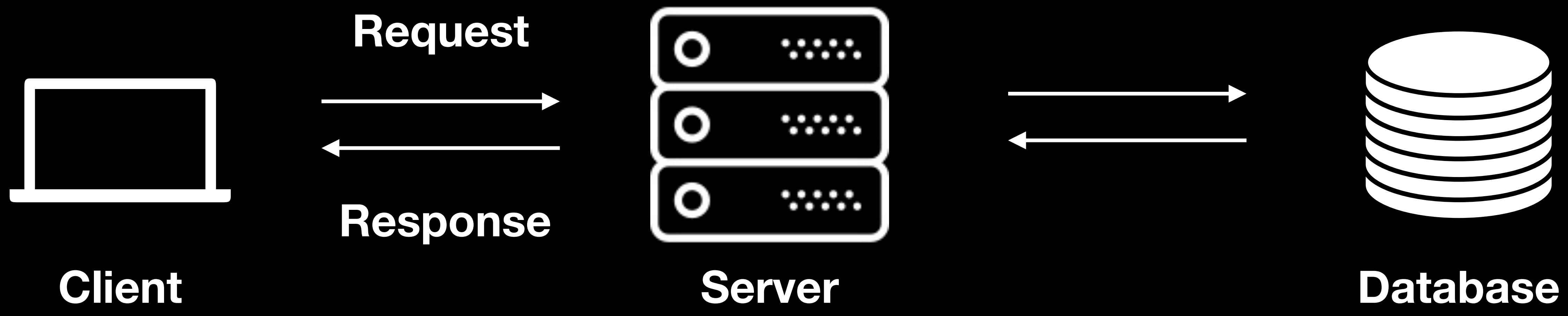


```
.
├──── .env = Environment variables
├──── LICENSE = License file
├──── README.md = Information about project
├──── node_modules = Packages of the project
├──── package.json = Meta-data of project
├──── package-lock.json = Meta-data of project
├──── public = Store file that user can see
|     ├──── assets
|     |     ├──── css = store all css
|     |     ├──── fonts = store all fonts
|     |     ├──── images = store all images
|     |     |     ├──── icons = store all icons
|     |     ├──── js = store all js
|     ├──── robots.txt = for search engine bot
|     └──── views = store all view template
|         ├──── includes = store reusable piece of template
├──── server = Store file that related to the server
|     ├──── auth = authentication related config
|     ├──── controllers = controller for each route
|     ├──── middlewares = reusable middleware for express route
|     ├──── models = models for data
|     ├──── routes = partial route
|     ├──── server.js = main entry of application
|     ├──── uploads = store user-uploaded file
└────────── utils = utility code
```

⑭ ExpressJS

Web Server

3-tier architecture



Express - Hello World

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello
World!'))

app.listen(port, () => console.log(`Example
app listening on port ${port}!`))
```

Routing

```
app.METHOD(PATH, HANDLER)
```

Routing - Route parameters

```
app.get('/users/:userId', (req, res) =>
res.send(`Hello, ${req.params.userId}`))
```

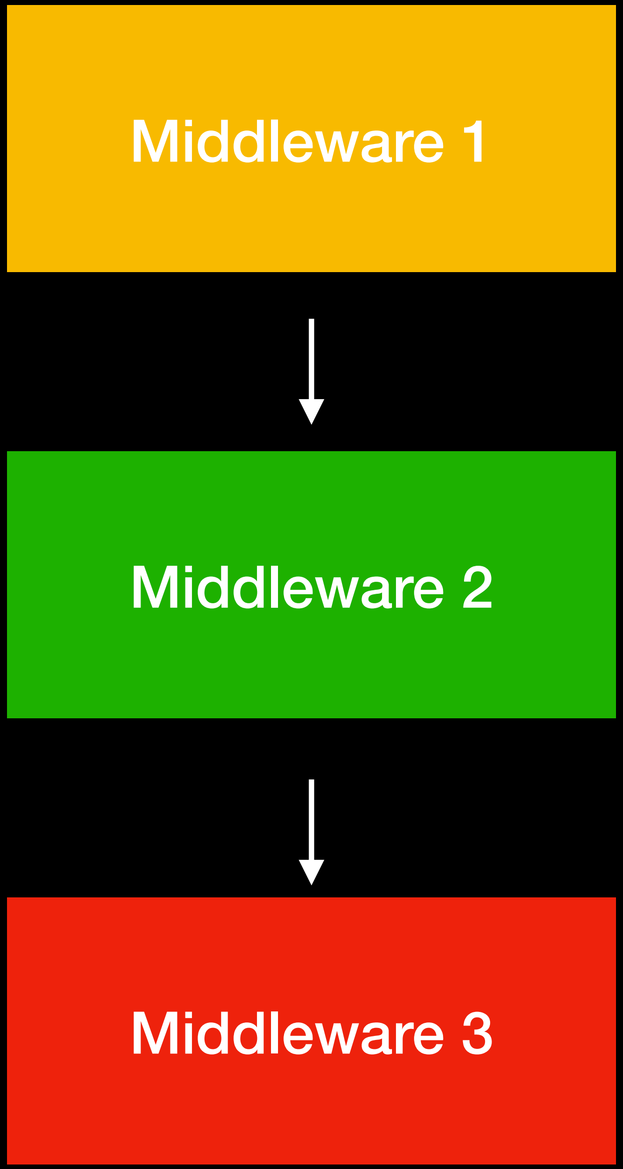
Router

```
const express = require(‘express')
const router = express.Router()
router.get('/', function (req, res) {
    res.send('Birds home page’)
})
router.get('/about', function (req, res) {
    res.send('About birds')
})
module.exports = router
```

Static File

```
app.use(express.static('public'))
```

Middleware



body-parser

`req.body`

JSON

```
res.json()
```

API

```
app.route(‘/books')
  .get(function (req, res) {
      res.send('Get all random book’)
  })
  .post(function (req, res) {
      res.send('Add a book’)
  })
  .put(function (req, res) {
      res.send('Update the book’)
  })
```

①⑤ EJS

Template Engine

Create template
Generate HTML file dynamically

EJS Syntax

`<%` 'Scriptlet' tag, for control-flow, no output

`<%_` ‘Whitespace Slurping’ Scriptlet tag,
strips all whitespace before it

`<%=` Outputs the value into the template (HTML escaped)

`<%-` Outputs the unescaped value into the template

`<%#` Comment tag, no execution, no output

`<%%` Outputs a literal '<%'

EJS Syntax

%> Plain ending tag

-%> Trim-mode ('newline slurp') tag, trims following newline

_%> ‘Whitespace Slurping’ ending tag, removes all whitespace after it

EJS - Include

```
<%- include(‘file/path', {key: value}); %>
```

EJS - Layout

```
<%- include('header'); -%>
    <h1>Title</h1>
    <p>My page</p>
<%- include('footer'); -%>
```

EJS - More Example

```
<% if (user) { %>

<h2><%= user.name %></h2>

<% } %>
```

EJS with ExpressJS

```
res.render(‘filename’, {

   additional: data

})
```

①⑥ RESTful API

Idea

Convention for create API endpoint

HTTP Verb + endpoint

e.g. GET /users

HTTP Verbs

GET

— — —

POST

— — —

PUT
PATCH

— — —

DELETE

GET

Get Data

GET /users = Get all users
GET /users/1 = Get user with ID 1
GET /books/novels/harry-potter-1
= Get harry potter 1 in novels category

POST

Create new data

POST /users = Create new user
POST /books/novels = Create new book in novel category

DELETE

Delete data

DELETE /users/1 = Delete user with ID 1
DELETE /books/novels/harry-potter-8 = Delete harry potter 8 in category novel

PUT

Update data with a new set of data

PUT /users/1 = Update data of user with ID 1 with a new set of data

PATCH

PATCH /users/1 = Update data of user with ID 1 with some new value of existing properties

API Request Client - Postman

POSTMAN

Call API - axios

```
axios.METHOD(url)
  .then(data => console.log(data))
  .catch(err => console.log(err))
```

MySQL in NodeJS

```
// In real development this setting
should be provided and can config
through .env file

const mysql = require('mysql2');
const pool = mysql.createPool({
    host: '35.247.178.19',
    user: 'YOUR USERNAME',
    password: 'YOUR PASSWORD',
    database: 'development',
    waitForConnections: true,
    connectionLimit: 10,
    queueLimit: 0
});
```

MySQL in NodeJS

```
pool.getConnection((err, connection) => {
        if (err) return console.error(err);
        connection.query("SQL COMMAND",(err,
          results, fields) => {
        if (err) return console.log(err);
        console.log(results);
        console.log(fields);
    });
});
```

RESTful API with ExpressJS

```
route.get(“/users”, (req, res) =>
  res.json(Users.findAll()));

route.get(“/users/:userId”, (req,
res) =>
  res.json(Users.findAll({
      userId: req.params.userId
  })));
```

RESTful API with ExpressJS

```
route.post(“/users”, (req, res) => {
    User.create({...req.body});
    res.send(`User ${req.body.userId}
      created!`);
});
```

RESTful API with ExpressJS

```
route.put(“/users/:userId”, (req, res) => {
  Users.update({
      userId: req.params.userId
  },
  {
      ...User.findAll({
        userId: req.params.userId
      }),
      userName: ‘lnwzaa007’
  });
  res.send(`User ${req.params.userId}
    updated!`);
});
```

RESTful API with ExpressJS

```
route.delete(“/users/:userId”, (req, res) => {
   User.delete(req.params.userId);
   res.send(`User ${req.params.userId}
     deleted!`)
});
```

RESTful API with ExpressJS

Note! There is a better way to write code as this one does not have error handling

jQuery

```
$(“css selector”)
```

AJAX - Call API from front-end

```
$.ajax(“/users”, {
    data: {
        name: ‘Pete’,
        github: ‘https://github.com/Pittawat2542'
    },
    method: ‘POST’
})
  .done((data, textStatus, jqXHR) => {
      console.log(data);
  })
  .fail((jqXHR, textStatus, errorThrown) => {
      console.log(textStatus);
  })
```

HTML Form

```
<form action="/users" id="search">
    <h3>Search User</h3>
    <div>
        <label for="id">
          USER ID:
        </label>
        <input type="text" name="id" required>
     </div>
     <button type="submit">Search</button>
</form>
```

Form with AJAX

```
$(“#search”).submit((event) => {
    event.preventDefault();
    const form = $(this);
    $.ajax({
        url: `/users/${form.serialize()}`,
        method: “POST",
       // data: {},
    });
})
.done((data, textStatus) => {
      console.log(data);
  })
.fail((jqXHR, textStatus, error) => {
      $("#result").text(jqXHR.responseText);
  });
```

NodeJS Endpoint

```
route.get(“/users/:userId”, async (req, res) => {
    cosnt { userId } = req.params;
    if (!userId) return res.sendStatus(400);
    try {
        const user = await User.findAll({ userId });
        if (!user) {
            return res.sendStatus(404);
        }
        res.json(user);
    } catch (err) {
        res.sendStatus(500);
    }
}
```

⑰ File

HTML File Upload

```
<form action="/pictures" method="POST"
enctype="multipart/form-data">
    <div>
        <label>Picture</label>
        <input type="file" name="picture" />
    </div>
    <button type="submit">Submit</button>
</form>
```

Multer Config

```
const path = require(“path");
const multer = require(“multer");
const storage = multer.diskStorage({
    destination: (req, file, callback) => {
        callback(null,
          path.join(__dirname, "../uploads"));
    },
    filename: (req, file, callback) => {
        callback(null, file.fieldname + "-" +
          Date.now() +
          path.extname(file.originalname));
    }
});
```

Multer Config

```
const fileFilter = (req, file, callback) => {
    const fileExtension =
      path.extname(file.originalname)
      .toLowerCase();
    if (fileExtension == ".jpg" ||
        fileExtension == ".jpeg" ||
        fileExtension == “.png") {
        callback(null, true);
    } else {
        callback(new Error(`Not supported file
          extension: ${fileExtension}`));
    }
};
```

Multer Config

```
const uploadPicture = multer({
   storage: storage,
   fileFilter: fileFilter,
   limits: {
       fileSize: 5 * 1024 * 1024
    }
}).single(“picture");
```

Handling File in ExpressJS

```
route.post(“/picture”, (req, res) => {
    const userId = req.user.id;
    try {
        multer.uploadPicture(req, res,
          async err => {
        if (err) {
            return res.status(400).send({
                error: err
            });
        }
        await database.createPicture({
            picture_file_name: req.file.filename
        });
        res.redirect(“/complete");
        });
    } catch (err) {
        console.error(err);
    }
};
```

⑱ Utilities

lodash

A modern JavaScript utility library delivering modularity, performance & extras.

Moment.js

Parse, validate, manipulate, and display dates and times in JavaScript.

SweetAlert 2

A beautiful, responsive, customizable, accessible replacement for javascript's popup boxes

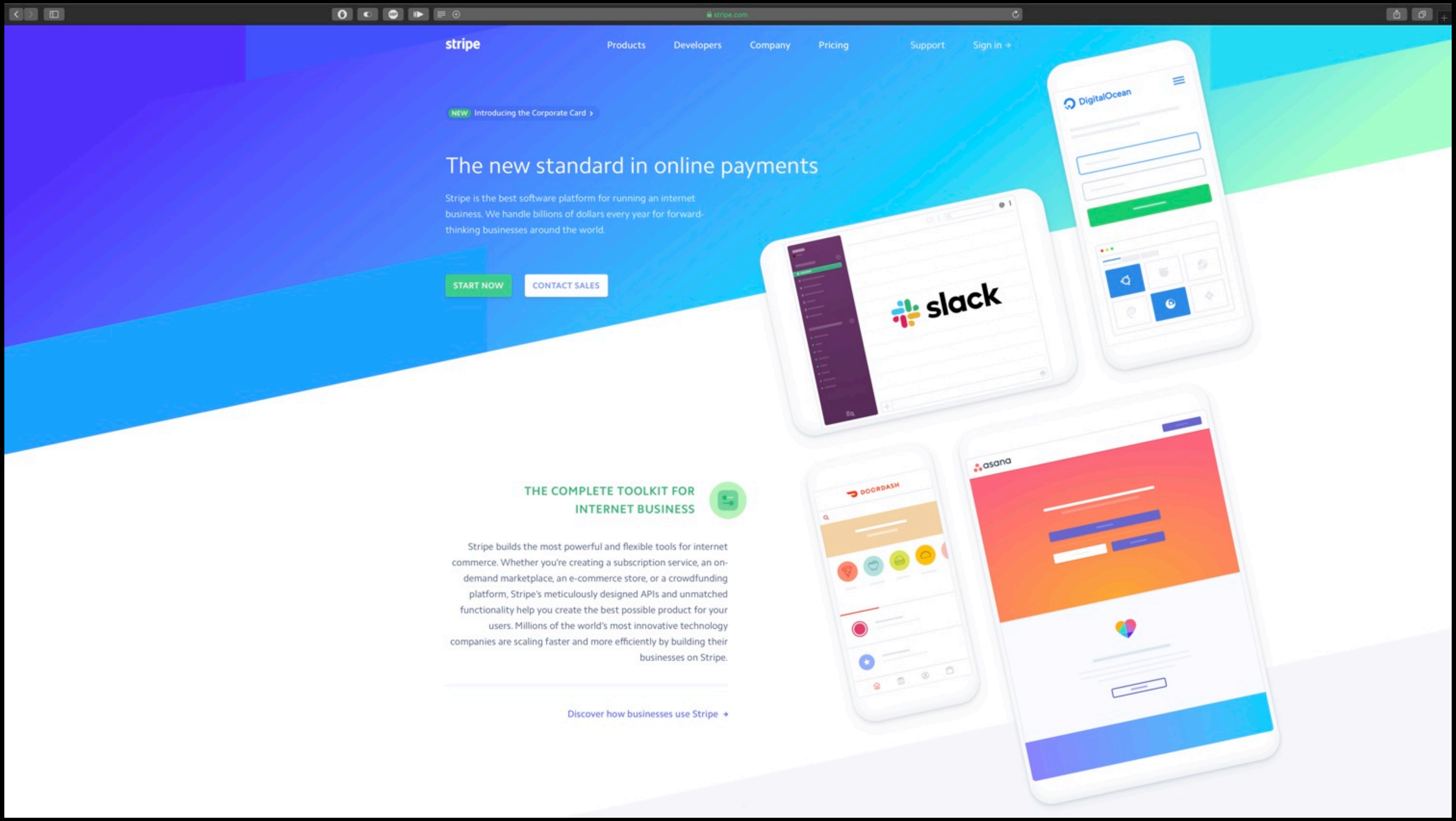
Convention for Development

DB - Model - View - Controller - Route

Let’s see real example on
JPC XV Website

Go on further…

Stripe



The new standard in online payments

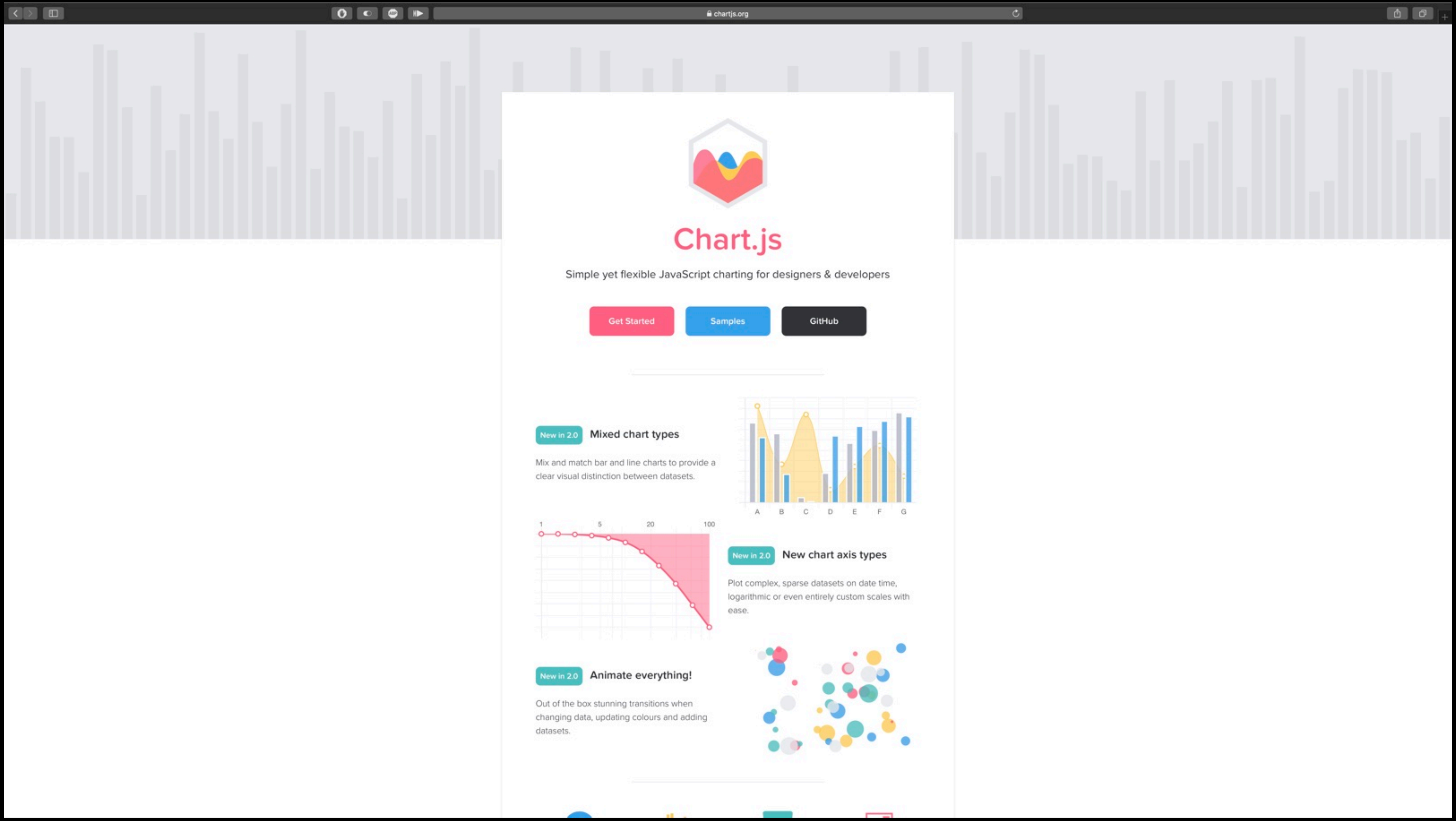
Stripe is the best software platform for running an internet business. We handle billions of dollars every year for forward-thinking businesses around the world.

THE COMPLETE TOOLKIT FOR INTERNET BUSINESS

Stripe builds the most powerful and flexible tools for internet commerce. Whether you're creating a subscription service, an on-demand marketplace, an e-commerce store, or a crowdfunding platform, Stripe's meticulously designed APIs and unmatched functionality help you create the best possible product for your users. Millions of the world's most innovative technology companies are scaling faster and more efficiently by building their businesses on Stripe.

Discover how businesses use Stripe →

ChartJS



chartjs.org

Chart.js

Simple yet flexible JavaScript charting for designers & developers

New in 2.0 **Mixed chart types**

Mix and match bar and line charts to provide a clear visual distinction between datasets.

New in 2.0 **New chart axis types**

Plot complex, sparse datasets on date time, logarithmic or even entirely custom scales with ease.

New in 2.0 **Animate everything!**

Out of the box stunning transitions when changing data, updating colours and adding datasets.

🏅End🏅

bit.ly/node-slide