Introduction to Web Development

About Me

Weder Mariano de Sousa

Specialist Senior Java - GFT

Technician **System Development**

Graduated **Computer Science**

Post Graduate in **Midias UFG**

Post Graduate in **Information Security**

GOJava

AWS User Group Goiânia

https://www.linkedin.com/in/wedermarianodesousa/

https://github.com/weder96

https://twitter.com/weder96

https://dev.to/weder96

aws certified Cloud Practitioner FOUNDATIONAL

aws certified Developer ASSOCIATE

AWS community builder Serverless

aws Technical Accredited PARTNER

aws Cloud Economics Accredited PARTNER

aws Business Accredited PARTNER

aws TECHNOLOGY File Storage SPECIALIZED

aws TECHNOLOGY Storage Data Migration SPECIALIZED

aws TECHNOLOGY Storage Data Protection & Disaster Recovery SPECIALIZED

aws Well-Architected PROFICIENT

aws TECHNOLOGY Amazon Braket

aws TECHNOLOGY Architecting

aws TECHNOLOGY Migration Foundations

aws TECHNOLOGY Amazon EKS

aws TECHNOLOGY Networking Core

aws TECHNOLOGY Serverless

aws TECHNOLOGY Events and Workflows

aws TECHNOLOGY Compute

aws TECHNOLOGY Cloud Essentials

This class aims to

I. Introduce different areas of Web development

II. Help you pick your path

What do you want to do?

I. Work as a developer at a company
II. Stafi freelancing or create your own agency
III. Become a consultant
IV. Create an app to make money
V. Code as a hobby

What do you want to do?

I. Frontend
II. Backend
III. Full Stack
IV. DevSecOps
V. Data Science
VI. Tester
VII. Systems Architect
VIII. Cloud Computing

Tools

I. OS : macOS, Linux, Windows

II. Text Editor : VSCode, Sublime Text, PyCharm

III. Browser : Chrome, Firefox, Safari

IV. Terminal : iTerm, Hyper, cmder, Git Bash

V. Design : Figma, Canvas, Sketch, Adobe XD

VI. Tests : Selenium,TestComplete ,Cypress, JMeter, OWASP , Cucumber, JUnit 5,Postman

The Building Blocks

HTML & CSS are almost always the first things that you're going to learn in Web development

I. HTML5
II. CSS Fundamentals
III. CSS Grid & Flexbox
IV. CSS Custom Propefiies
V. CSS Transitions
VI. CSS 3

Responsive Design

Most people use the Internet on their mobile devices, so creating responsive layouts is impofiant

I. Viewpofi
II. Media Queries
III. Fluid Widths
IV. Mobile-First Design

CSS Preprocessors

Makes CSS more efficient and adds more functionality to standard CSS

I. SASS/SCSS
II. Stylus
III. LESS

CSS Frameworks

Great for prototyping or when we are not great with design

I. Bootstrap
II. Materialize
III. Tailwind CSS

CSS Methodologies

Make it easy to maintain CSS

I. Block, Element, Modifier (BEM)
II. Object-Oriented CSS (OOCSS)
III. Scalable and Modular Architecture for CSS (SMACSS)

Vanilla JavaScript

It is the programming language of the browser,
and impofiant to make pages dynamic

I. Fundamentals
II. DOM
III. JSON
IV. Fetch API
V. Modern JavaScript (ES2015+)

JavaScript Libraries

I. jQuery, Zepto, Lodash
II. Module loaders : RequireJS, SystemJS
III. Templating : Handlebars, EJS, Mustache.js, Pug

Optimizations

I. Code Minification
II. Image Compression
III. Lazy loading

Tools Developers

I. Git & GitHub, GitLab, Bitbucket
II. Browser Dev Tools
III. Package managers: npm, yarn
IV. Task runners : gulp,grunt
V. Module bundlers: webpack, parcel, rollup

Frontend Deployment

Get our website into a Web server and to our users

I. Static Hosting: Netlify, Vercel, GitHub Pages, Firebase Hosting

II. Domains: Namecheap, Google Domains, GoDaddy, Registro.br

I. SSL Cefiificates

Frontend Developer

I. Build websites for yourself or other people
II. Create mobile friendly layouts
III. Create CSS animations and effects
IV. Work with a CSS framework
V. Add dynamic page functionalities
VI. Build client side apps with JavaScript and JS libraries
VII. Use your browser's dev tools
VIII. Utilize Git for version control
IX. Deploy and maintain frontend projects

What's next?

I. Frontend JavaScript Framework
   II. (React, Vue, Angular, etc.)

III. Server-Side Language
   IV. (Python, Ruby, PHP, etc.)

Frontend Frameworks

Build powerful single-page applications
with organized and interactive UIs

- Vue : Easiest to learn, really gaining traction Fairly easy to
- React : learn, still the most popular
- Angular : Steep learning curve, used more in enterprise

State Management

I. Vue : Vuex
II. React : Redux, Context API with Hooks, Mobx
III. Angular : RxJs, NGRX, Services

Server-Side Rendering

I. Better SEO
II. File System Routing
III. Automatic Code Splitting
IV. Static Expofiing
V. Nuxt.js
VI. Next.js
VII. Angular Universal

Static Site Generators

I. Better SEO
II. File System Routing
III. Data fetching during
IV. build time
V. Gatsby.js, Next.js
VI. Nuxt.js, Gridsome, VuePress
VII. Scully
VIII. Eleventy, Hugo

TypeScript

I. Superset of JavaScript
II. Static Types
III. Modules, Decorators, Classes
IV. Compiles into JavaScript code

Frontend Developer

I. Familiar with a popular frontend framework
II. Build advanced frontend apps and interfaces
III. Smooth frontend workflow
IV. Interact with backend APIs and data
V. Manage application and component-level state
VI. Bonus: server-side rendering (SSR), static site generation (SSG)

The Great Divide

css-tricks.com/the-great-divide

The divide is between people who self-identify as a (or have the job title of) front-end developer, yet have divergent skill sets.

On one side, an army of developers whose interests, responsibilities, and skill sets are heavily revolved around JavaScript.

On the other, an army of developers whose interests, responsibilities, and skill sets are focused on other areas of the front end, like HTML, CSS, design, interaction, patterns, accessibility, etc.

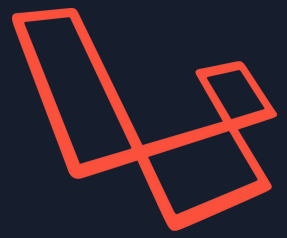
Server-Side Language

To be a full stack or backend developer, you will need to learn a server-side programming language

I. Node.js
II. Python
III. PHP
IV. Ruby
V. Go
VI. Rust
VII. Java
VIII. C#

Server-Side Frameworks

I. Node.js : Express, Koa, Feathers.js
II. Python : Django, Flask
III. PHP : Laravel, Symfony
IV. Ruby : Ruby on Rails
V. Java : Spring MVC, Spring Boot
VI. C# : ASP.NET



Databases

Most apps need a place to store data

I. Relational Databases : PostgreSQL, MySQL, SQL Server
II. NoSQL : MongoDB, CouchDB
III. Cloud Databases : Cloud Firestore
IV. Lightweight & Cache : Redis, SQLite

REST API

Standard way for our frontend and backend to communicate with each other

I. Verb + location
II. GET /users
III. POST /users
IV. GET /users/:id
V. PUT /users/:id
VI. DELETE /users/:id

GraphQL

Query language for our API, alternative to REST

I. Single endpoint
II. We ask for what we want, and we get that exact data
III. Simple syntax, similar to JSON
IV. Fairly easy to implement, has implementations in different programming languages

Content Management

Add content to our apps, great for clients to be able to update their own content

I. Traditional CMS : Wordpress, Drupal
II. Headless CMS : Strapi, ContenVul, Sanity, Prismic.io

Deployment & DevOps

I. App hosting : DigitalOcean, Linode, Heroku, AWS, Azure, Google Cloud PlaVorm

II. Web servers : Caddy, Nginx, Apache

III. Virtualization: Docker

IV. Load balancing, monitoring, security, etc.

Full Stack Developer

I. Create and deploy powerful, database-driven Web apps
II. Build user interfaces using your choice of frontend technologies
III. Fluent in a server-side programming language
IV. Setup dev environments and workflows
V. Build backend apps and APIs
VI. Work with databases (relational or NoSQL)
VII. Deploy to production

Progressive Web Apps

Regular Web apps but with a native native feel in terms of experience, layout, and functionality

I. Responsive
II. Offline content
III. Installable
IV. Splash screen
V. Secure (over HTTPS)
VI. Reliable, fast, and engaging

JAMstack

I. JavaScript + APIs + Markup
II. No restriction on frameworks on libraries
III. Websites are served as static HTML files generated by static site generators
IV. High performance, generated at deploy time
V. Cheaper and easy to host and scale

Firebase

I. Web and mobile app development plaVorm
II. Backend as a Service
III. Authentication, Database, Storage, Functions, Notifications, etc.

Phew!

I. Don't get overwhelmed
II. Figure out what exactly you want to do
III. Learn one thing at a time
IV. The more you learn, the easier it is to learn more and fit all these technologies together

About Me

Weder Mariano de Sousa

Specialist Senior Java - GFT

Technician **System Development**

Graduated **Computer Science**

Post Graduate in **Midias UFG**

Post Graduate in **Information Security**

GOJava

AWS User Group Goiânia

Q & A

aws certified Cloud Practitioner FOUNDATIONAL

aws certified Developer ASSOCIATE

AWS community builder Serverless

https://www.linkedin.com/in/wedermarianodesousa/

https://github.com/weder96

https://twitter.com/weder96

https://dev.to/weder96

aws Technical Accredited PARTNER

aws Cloud Economics Accredited PARTNER

aws Business Accredited PARTNER

aws TECHNOLOGY File Storage SPECIALIZED

aws TECHNOLOGY Storage Data Migration SPECIALIZED

aws TECHNOLOGY Storage Data Protection & Disaster Recovery SPECIALIZED

aws Well-Architected PROFICIENT

aws TECHNOLOGY Amazon Braket

aws TECHNOLOGY Architecting

aws TECHNOLOGY Migration Foundations

aws TECHNOLOGY Amazon EKS

aws TECHNOLOGY Networking Core

aws TECHNOLOGY Serverless

aws TECHNOLOGY Events and Workflows

aws TECHNOLOGY Compute

aws TECHNOLOGY Cloud Essentials

THANK YOU

Weder Sousa

https://www.linkedin.com/in/wedermarianodesousa/

https://github.com/weder96

https://twitter.com/weder96

https://dev.to/weder96

HISTORY OF THE INTERNET 1960 TO 2020