Guia Ágil

TechGuide - Alura

React

Nivel 1

- [ ] **HTML - Fundamentals**:

- HTML is a markup language that defines the structure of your content. HTML consists of a series of elements, which you use to make it appear a certain way, or act a certain way. The enclosing tags can make a word or image hyperlink to somewhere else, can italicize words, can make the font bigger or smaller, and so on.
- Learning which tags are required for basic HTML
- Creating a text paragraph
- Displaying an image
- Knowing the difference between 'h1', 'h2', 'h3', etc.
- Creating a hyperlinked text
- Creating a form with relevant fields
- Creating an ordered or unordered list of items
- Creating a list of items within a dropdown list
- Linking to a CSS file
- Creating a table
- Adding IDs and classes

- [ ] **CSS - Fundamentals**:
    - Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML. CSS can be used for very basic document text styling — for example, for changing the color and size of headings and links. It can be used to create a layout — for example, turning a single column of text into a layout with a main content area and a sidebar for related information. It can even be used for effects such as animation.
    - Learning the visual structure of a page, with 'margin' and 'padding'
    - Establishing the size with 'width' and 'height'
    - Learning about the position of an element ('static', 'relative' or 'absolute')
    - Learning about the display of an element ('block', 'inline', 'inline-block')
    - Learning how to position images in relation to text
    - Learning about alignment
    - Learning about font style
    - Learning the differences and advantages of using the different units of measurement in CSS (%, relative, etc)
    - Connecting to the elements (IDs, classes) of an HTML file
    - Changing the characteristics of an element when the mouse hovers over it
    - Learning box-sizing
    - Learning Flexbox
    - Learning Grid
- [ ] **JavaScript - Fundamentals**:
    - JavaScript is the world's most popular programming language and is one of the core technologies of the World Wide Web, alongside HTML and CSS. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles.
    - Knowing the primitive types

- Declaring variables, considering the difference between 'var', 'let' and 'const
- Using conditional structures ('if', 'else')
- Know the assignment and comparison operators ('=', '==', '===')
- Using repetition structures and loops ('while', 'for')
- Using functions, passing parameters and arguments
- Manipulating arrays and lists
- Getting data from an API
- Making asynchronous calls using 'Async/Await', 'Promise', etc

☐ **DOM - Fundamentals**:

- The Document Object Model (DOM) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects; that way, programming languages can interact with the page.
- Understanding how the DOM tree works
- Accessing and manipulating HTML and CSS elements
- Accessing the parents and children of an element
- Inserting a new element to the tree
- Removing an element from the tree
- Waiting for an event on a certain page element using addEventListener()

☐ **SPA Concepts**:

- A single-page application (SPA) is a web application or website that interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the default method of a web browser loading entire new pages. The goal is faster transitions that make the website feel more like a native app.
- Understanding what an SPA is
- Establishing routes to other pages

- Knowing SPA frameworks
- Communicating with APIs

- [ ] **React - Components**:

- React lets you define components as classes or functions. Components defined as classes provide more features. They accept arbitrary inputs (called "props") and return React elements describing what should appear on the screen.
- Understanding how components work
- Knowing the Styled Components library
- Understanding the difference between class and functional components

- [ ] **React - Props**:

- Props are an object that is injected in components and provides some data that can be shared between other components in an uni-directional flow of data, from a parent to a child element. Props are read-only.
- Passing props
- Manipulating props

- [ ] **React Hooks - State**:

- The React `useState` Hook allows us to track state in a function component. The `useState` Hook can be used to keep track of strings, numbers, booleans, arrays, objects, etc.
- Control the state of components
- Manipulating variables
- Updating an element or component

- [ ] **Creating a React app**:

- Create React App is an officially supported way to create single-page React applications. It offers a modern build setup with no configuration.
- Structuring a new React project
- Creating a functional application from scratch

- [ ] **React Hooks - Effect**:
  - UseEffect() is a React Hook which allows you to handle side effects in your functional React components.
  - Executing a component only after rendering
  - Acessing the props of an element
  - Calling APIs

Nivel 2

- [ ] **React Hooks - Memo**:
  - The React Hook `useMemo` returns a memoized value. `useMemo` will only recompute the memoized value when one of the dependencies has changed.
  - Controlling the state of external variables
  - Avoiding expensive calculations on every render
- [ ] **React Hooks - Callback**:
  - The React `useCallback` Hook returns a memoized callback function.
  - Isolating resource intensive functions so that they will not automatically run on every render
- [ ] **React Hooks - Ref**:
  - The `useRef` Hook allows you to persist values between renders.
  - Storing a mutable value that does not cause a re-render when updated
  - Accessing a DOM element directly
- [ ] **React - Design System Libraries**:
  - A design system is a collection of reusable components, guided by clear standards, that can be assembled together to build any number of applications.
- [ ] **React Developer Tools**:

- React Developer Tools is used to inspect React components, edit props and state, and identify performance problems.
- Debugging applications

- [ ] **Front-end Semantic Versioning**:

- SemVer (Semantic Versioning) is a simple set of rules and requirements that dictate how version numbers are assigned and incremented.
- Organizing the dependencies of a project
- Avoiding the "dependency hell"

- [ ] **CSS-in-JS**:

- CSS-in-JS refers to a pattern where CSS is composed using JavaScript instead of defined in external files.
- Handling CSS code using JavaScript
- Using styled-components
- Knowing Styled-JSX

- [ ] **Styled Components**:

- Styled-components allow you to write actual CSS code to style your components using tagged template literals and the power of CSS.
- Handling CSS code in components

- [ ] **React - Webrouting**:

- Manipulating navigation between interfaces and components

- [ ] **TypeScript - Fundamentals**:

- TypeScript is a strongly typed programming language that builds on JavaScript.
- Understanding in depth what types are and the importance of typed programming
- Learning what TypeScript is, why it was created, how it works and its relationship with JavaScript

- Knowing the TypeScript tools (integration with the code editor, static checker and compiler)
- Writing code in TypeScript using its tools (interfaces, enum, decorators, etc.)

☐ **React Testing Library**:

- React Testing Library builds on top of DOM Testing Library by adding APIs for working with React components. It is a set of helpers that let you test React components without relying on their implementation details.
- Testing React components

☐ **Jest**:

- Jest is a JavaScript test runner that lets you access the DOM via jsdom. It provides a great iteration speed combined with powerful features like mocking modules and timers so you can have more control over how the code executes.
- Testing components

☐ **Cypress**:

- Cypress is a front-end testing tool that allows for setting up, writing, running, and debugging tests.

☐ **JavaScript - Callbacks and Promises**:

- A Promise is a proxy for a value not necessarily known when the promise is created. This lets asynchronous methods return values like synchronous methods - instead of immediately returning the final value, the asynchronous method returns a promise to supply the value at some point in the future.
- A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.
- An async function is a function declared with the async keyword, and the await keyword is permitted within it. The async and await keywords enable asynchronous, promise-based behavior to be written in a cleaner style, avoiding the need to explicitly configure promise chains.

- Understanding the concept of asynchronous programming
- Writing asynchronous code understanding the concept of promises in JavaScript
- Using JavaScript methods, keywords and objects for handling promises like 'Async/Await', '.then()', 'Promise', etc
- Learning in which situations you need to use asynchronous programming
- Calling APIs with `fetch()`

- [ ] **JavaScript - Error handling**:

- Error handling refers to the response and recovery procedures from error conditions present in a software application. In other words, it is the process comprised of anticipation, detection and resolution of application, programming or communication errors.
- Knowing and handling the most common exceptions
- Knowing the types of errors and in which situations they can occur
- Using 'try' and 'catch' for error handling
- Learning on what occasions and how to use `throw`
- Creating specific exceptions according to your application's needs

- [ ] **Babel - Fundamentals**:

- Babel is a JavaScript compiler. It is a toolchain that is mainly used to convert ECMAScript 2015+ code into a backwards compatible version of JavaScript in current and older browsers or environments.

Nivel 3

- [ ] **Lottie**:

- Lottie is a library that parses Adobe After Effects animations exported as json with Bodymovin and renders them natively on mobile and on the web.

- [ ] **Framer Motion**:

- Framer Motion is a production-ready motion library for React from Framer.

- Creating animations

- [ ] **Service Workers**:

- Service workers essentially act as proxy servers that sit between web applications, the browser, and the network (when available). They are intended, among other things, to enable the creation of effective offline experiences, intercept network requests and take appropriate action based on whether the network is available, and update assets residing on the server. They will also allow access to push notifications and background sync APIs.

- [ ] **React Hooks - Form**:

- React Hooks Form is a library that provides form validation.

- [ ] **Lodash**:

- Lodash is a modern JavaScript utility library delivering modularity, performance & extras. Lodash's modular methods are great for iterating arrays, objects, & strings; manipulating & testing values; and creating composite functions.

- [ ] **GraphQL**:

- GraphQL is a new API standard that provides a more efficient, powerful and flexible alternative to REST. It was developed and open-sourced by Facebook and is now maintained by a large community of companies and individuals from all over the world.
- Understanding how GraphQL is used in API development
- Creating APIs using GraphQL libraries and frameworks

- [ ] **Apollo Client**:

- Apollo Client is a comprehensive state management library for JavaScript that enables you to manage both local and remote data with GraphQL.

- [ ] **Redux Saga**:

- Redux Saga is a library that aims to make application side effects (i.e. asynchronous things like data fetching and impure things like accessing the browser cache) easier to manage, more efficient to execute, easy to test, and better at handling failures.

- [ ] **NextJS - Fundamentals**:

- Next.js is an open-source web development framework created by Vercel enabling React-based web applications with server-side rendering and generating static websites.
- Building web interfaces
- Decreasing page load times
- Rendering server-side pages
- Improving performance in React
- Building API routes with serveless functions
- CSS-in-JS

Habilidade Auxiliar: Infrastructure and Back-end

- [ ] **Git & GitHub - Fundamentals**:

- Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- GitHub is a hosting service for software development and version control using Git.
- Creating a repository
- Cloning a repository
- Committing, pushing and pulling to and from the repository
- Reversing a commit
- Creating branches and pull requests
- Handling merge and conflicts

- [ ] **HTTP - Fundamentals**:
  - HTTP stands for Hyper Text Transfer Protocol. Communication between client computers and web servers is done by sending HTTP Requests and receiving HTTP Responses.
  - Understanding the difference between HTTP verbs
  - Testing requests and checking the status codes in the browser
  - Learning how to make a HTTP request on the command line with WGET
  - Downloading an image with WGET
  - Performing a POST
- [ ] **JSON**:
  - JSON stands for JavaScript Object Notation. It is a text format for storing and transporting data.
  - Creating an object
  - Transforming an object into a string
  - Transforming a string into an object
  - Manipulating an object
- [ ] **Command Line - Fundamentals**:
  - CLI is a command line program that accepts text input to execute operating system functions.
  - Knowing the most important commands
- [ ] **Cloud - Fundamentals**:
  - Cloud, or cloud computing, is the distribution of computing services over the Internet using a pay-as-you-go pricing model. A cloud is composed of various computing resources, ranging from the computers themselves (or instances, in cloud terminology) to networks, storage, databases, and everything around them. In other words, everything that is normally needed to set up the equivalent of a server room, or even a complete data center, will be ready to use, configured, and run.

- Knowing the difference between IaaS, PaaS and SaaS
- Knowing the largest cloud providers
- Specializing in a specific provider of your choice

- [ ] **YARN**:

- Yarn is a package manager for your code. It allows you to use and share code with other developers. Code is shared through something called a package (sometimes referred to as a module). A package contains all the code being shared as well as a package.json file which describes the package.
- Managing packages
- Managing dependencies
- Installing packages offline
- Commands
- The yarn.lock file

Habilidade Auxiliar: UX and Design

- [ ] **Design Systems**:

- A design system is a collection of reusable components, guided by clear standards, that can be assembled together to build applications.
- Creating and maintaining libraries that will be consumed and used as a standard for building a project

- [ ] **Figma - Fundamentals**:

- Figma is a collaborative web application for interface design. The feature set of Figma focuses on user interface and user experience design, with an emphasis on real-time collaboration, utilising a variety of vector graphics editor and prototyping tools.
- Creating page layouts and components

- [ ] **Design components**:

- Knowing the components that describe a layout or interface

- [ ] **Color systems**:

- Defining a color palette that makes sense for a given interface

- [ ] **How to use Fonts**:

- Choosing the most appropriate font for a given project
