**Curriculum**

* Data structures and algorithms foundation
* Data structures and algorithms advance
* Fontend basics for beginners
  + HTML
    - Elements and structure
      * Document
      * Elements
        + Div
        + Span
        + Picture

Rendering optimizations

SVGs and WebP

Sprites

Font icons

Lazy loading

* + - * + List
      * Assignments
        + MCQs to ensure understanding of basic tags
        + Make a one page website for your startup, which will be selling homemade organic cosmetic

Displays “Organic Homemade Cosmetic Shop” inside h1 title

Show a h2 title with text “Ingredients used in organic cosmetics”

Ingredient 1

Ingredient 2

Ingredient 3

Ingredient 4

Ingredient 5

Each ingredient name should have a link to the wikipedia page of the item, clicking on the link should open it in the new page.

Add a link with text, “available items”, and it should link to “/cosmetics.html”

**Step by step guide**

add the <!DOCTYPE html> declaration as the very first line of code at the top of the index.html file.

Add head element

Add body element

Platform support must have: Side by side view of html code being written

Great to have, judge to check the correctness of each step.

* + - Tables
      * MCQs
      * Assignment
        + Create a page with a table showing list of cosmetics available in your online store, file should be named cosmetics.html.
        + Link to page showing ideal outcome
        + Step by step guide
        + Hints associated with each step.
        + Judge to show passing/failing test cases
    - Forms
      * MCQs
      * Assignment
        + Create a page to order the cosmetics
        + The page should have following fields

Name of the person ordering the cosmetics

Name of the product, the customer wants to buy

Quantity, ensure min value of 1 and max value of 5

Address for delivery

Pin code

Submit button to place the order

Submit action should take to thank\_you.html

Create a new file, thank\_you.html, with a thank you message.

* + - * + Step by step instructions, with hint associated with each
      * (Later in curriculum)
        + Methods
        + Names
        + ID
        + Security aspect of GET vs POST (even via SSL)
    - Semantic HTML
      * MCQs
      * Assignment
        + Create a product page, for one of the cosmetics, use semantics, with audio, video, description of images etc.

* CSS
  + Adding CSS to HTML
    - Via external file
    - Assignment on code built so far, load a pre provided external css file to current html
  + Selectors and specificity
    - Assignment
  + Box Model
    - Flexbox
    - CSS Grid
  + Responsiveness
  + CSS Reset
  + CSS Animations, and CSS variables
  + Shadow DOM (Web Components)
* HTTP Protocol basics
  + GET/POST/PUT/DELETE, when to use what?
  + HTTP and HTTPS
* HTTP Advance
  + Caching
  + Compression
  + W3 consortium
  + RFCs
    - Reading
    - Contributing
* JS Language Constructs
  + Codesandbox (or alternatives IDEs)
  + Language Documentation - MDN
  + Functional Language
  + Types Supported in JS
    - Primitives and internal functions
      * Numbers
      * Iterables
      * Array
      * String
      * Maps
      * Set, WeakMap, WeakSet
      * Date/time
    - Variables
    - Objects
      * New Operator / constructor / classes
        + Static variables in functional and classes
        + Private vs protected properties/methods
        + Setters and getters
        + Mixins
        + Extending built-ins
      * Overriding default methods like toString to generate intended consequences. (Type casting on our own terms).
      * Object in heap vs stack
      * Garbage Collection
      * Accessing non-existent properties. (?.)
  + This keyword
    - Global Object (Scope)
    - Call, Bind, Apply & Polyfills
  + Arrow Functions
  + Functional Scope
    - Currying
    - Recursion in JS
    - Prototypes / prototypal inheritance
    - Closures
      * Private variables in JS, using closures
    - Classes in JS
  + Pass by Reference vs. Value, deep cloning vs shallow cloning
  + Basic Operators
  + Statements vs. Expressions vs Comments
  + “use strict”
  + Loops, Conditions, Switch
    - :? vs ??
    - Short circuit evaluation
  + Debugging JS
    - Alert
    - Console
    - Debugger;
    - Error Handling
      * Custom Errors
    - Dev Tools
      * Conditional breakpoints
      * Watch
* Browsers & the Event Model
  + Single Threaded Nature of the Browser
  + DOM Manipulation
    - Render Tree
    - CSSOM
    - DOM built-in methods (positions, width, color getters and setters)
    - Page Lifecycle Events (onload etc.)
    - Loading external scripts - async, defer
    - Tapping into loading errors on external resources (IMG, CSS, Scripts etc.)
  + Browser Event Loop
    - Events Delegation, bubbling, capturing
      * Custom Events
      * Mouse/Pointer & Keyboard Events
      * Drag & Drop
      * Scrolling
    - Default actions (form submits etc)
  + Jank
* Browser Specific
  + Accessibility
  + SEO
  + Semantic Web
  + FOUC
* JS Advanced
  + Server side rendering, and micro frontends
  + Debounce & Throttling
  + Hoisting
  + Closures
  + JSON for interfacing with external systems.
  + Functions
    - Limits on Recursion & the Stack
    - Callback Hell
    - Memoisation
    - … operator
    - Immediately Invoked Functional Expressions + Named Functional Expressions
    - Block scope vs Functional scope - var vs let vs const
  + Objects
    - Prototypal Inheritance
  + Settimeout
  + JS Event Loop
  + Promises
    - Promise API
    - Promise error handling
    - Microtask Queue
    - Promise Chaining
    - Async/Await
    - Generators
    - Async Iteration & Generators
    - Transform Callbacks to Promises -> Avoiding Callback Hell
  + Currying
  + Web Workers, Service Worker (FE2)
* React for Frontend
* Testing Frontend
  + Postman
  + Jest, Mocha and the likes
* Bad code that works
  + 100x coder trap avoidance
* Deployment
  + GIT
  + Initialising any codebase
  + ES6 Modules, chunking code into multiple files
  + Linting (Prettier)
  + Pre-commit hooks
  + Pre-processors / Transpiler (for HTML, CSS, JS)
    - Css-in-JS and Module CSS
  + Polyfills / Creating Polyfills
  + Cache Bursting with CDNs
  + Questions / Assignments
    - How does JSX code work in the browser?
    - Is there any way to make JSX work in the browser?
* Frontend Design Patterns
* WebSocket
* Web Frontend Optimisation
* Browser Network Layer
  + Multiple requests
  + Domain Sharding
  + Caching
  + CDNs
  + Static assets handling
  + Force refresh via code
* Web Security
  + JS Vuln
  + XSS
  + Eval
  + HTTP only cookie

**Bar Raiser**

* JS advanced
  + Symbols in JS
  + Objects Advanced
    - Descriptors in JS
* Sense of web design aesthetics
* Framebursting
* fileReader API (read PDFs for example)
* Running ML/AI model in UI
* Testing
  + TDD
  + FE Testing frameworks
  + Mocking in the browser

**Sample Projects**

* Responsive Page With Navbar
* Infinite Scrolling
* Auto adjusting posts (stacking containers)
* Pagination
* Kanban Board
* Add a new SSL Certificate to a Domain.
* Make a broken website work
* Cross window communication (postMessage APIs, localStorage, sessionStorage, what else?)
* Strategies to improve performance of a web application (improve lighthouse score)

**Assuming covered elsewhere -**

* String, Character, Encoding UTF-8/16 etc. Emojis in DB, Browser, JSON etc.
* Regexp basics (no need to remember it all, just knowing something like this exists)