Weatherly

By Dorin-Mihai Manea

Objective

A **modern** web application for checking the weather.

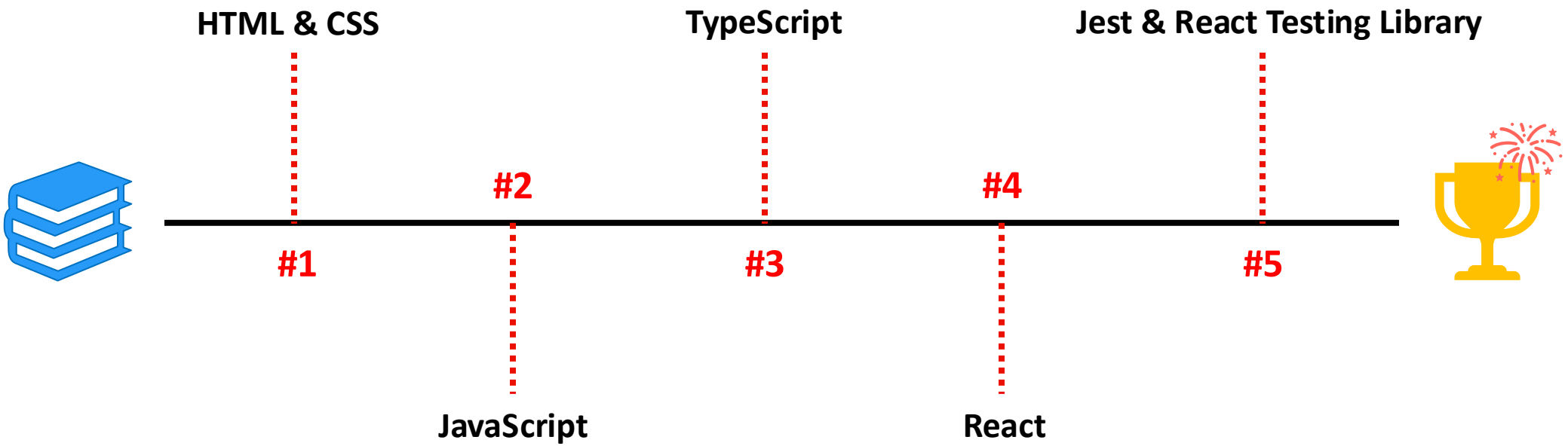
simple
mobile friendly
polished

CROSS-DEVICE COMPATIBLE

INTUITIVE AND CONSISTENT DESIGN

SCALABLE AND MAINTAINABLE

Tech Stack



Practice 1

This milestone lays the foundation of the project, including:

- HTML skeleton: *Well-structured* and *semantically meaningful*;
- CSS styles: Designed for a *polished, responsive layout*;
- Resources: images, icons.

While not yet interactive, this stage focuses on creating a *cross-device compatible* template with media queries, ensuring it works beautifully on all screens .

Challenges

- Using flexbox and grid appropriately;
- Making the footer stick to the bottom of the page no matter the size of the main content';
- Adjusting the background to cover the entire viewport no matter the window dimensions;
- Media queries.

Practice 2

Using Vanilla JavaScript and asynchronous programming, I added *interactive features* that bring functionality to the website.

- *Geolocation* – *W*hen loaded, the app attempts to geolocate the user and displays the weather for their current location. Default location in case of error: Bucharest.
- *Check the weather for any location* – By using the search bar, provided the data is available from the OpenWeatherMap API.
- *Loading spinner* – During network issues, a loading spinner keeps users informed, encouraging them to check their connection and try reconnecting.

Challenges

- Learning a new programming language;
- Asynchronous mechanisms – `Promise.all`, etc.
- Modularizing code into functions and multiple files;
- Error handling – input, API fails, network issues;
- Reverse geocoding – used for more precise geolocalisation.

Practice 3

Leveled up the project by converting all JavaScript scripts to TypeScript, introducing benefits like:

- *Strong typing;*
- *Interfaces;*
- *Decorators.*

This transition ensures better *scalability*, improved *code clarity*, and *fewer runtime errors* 🚫🐞.

Challenges

- Benefiting from all the advantages TypeScript provides across a very modularized project :-)

Practice 4

- Transitioned to a ***component-based architecture*** using React .
- A more *modular*, *scalable*, and *maintainable* codebase.
- Leveraged React hooks to manage *state* and *side effects* -> This shift *improves performance* and enables *faster development* by *reusing components across the app*.

Challenges

- Transitioning to React can be burdensome even if the codebase is small, but from then on it really simplifies things;
- Using the appropiate hooks -> implemented a bit of *memoization*;
- Spliting the large CSS file for styles into multiple modules.

Practice 5

Implemented *unit testing* using Jest and React Testing Library to ensure individual components function correctly. I also included an encapsulating `ErrorBoundary` component to gracefully handle all the unaccounted-for errors.

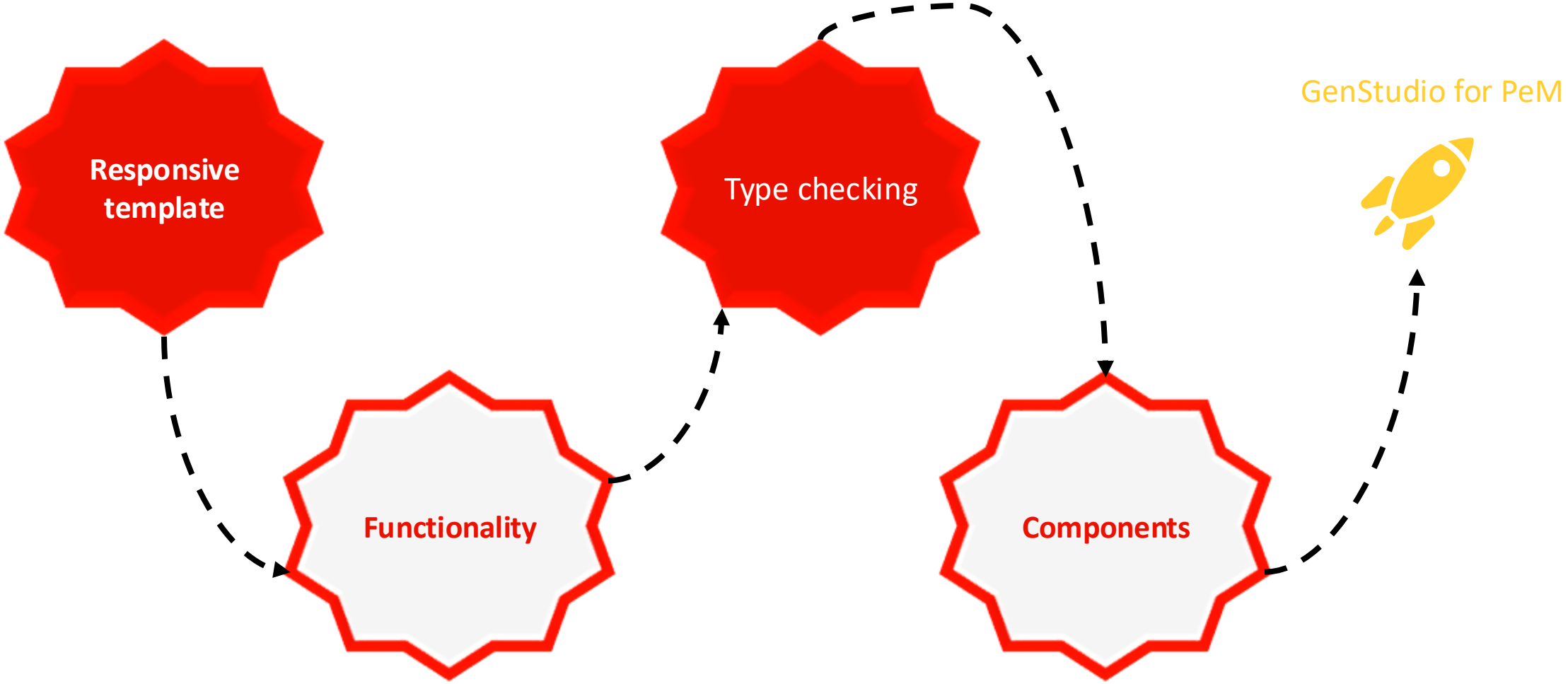
It helps maintain a *high-quality codebase* while *making refactoring easier and safer*.

Challenges

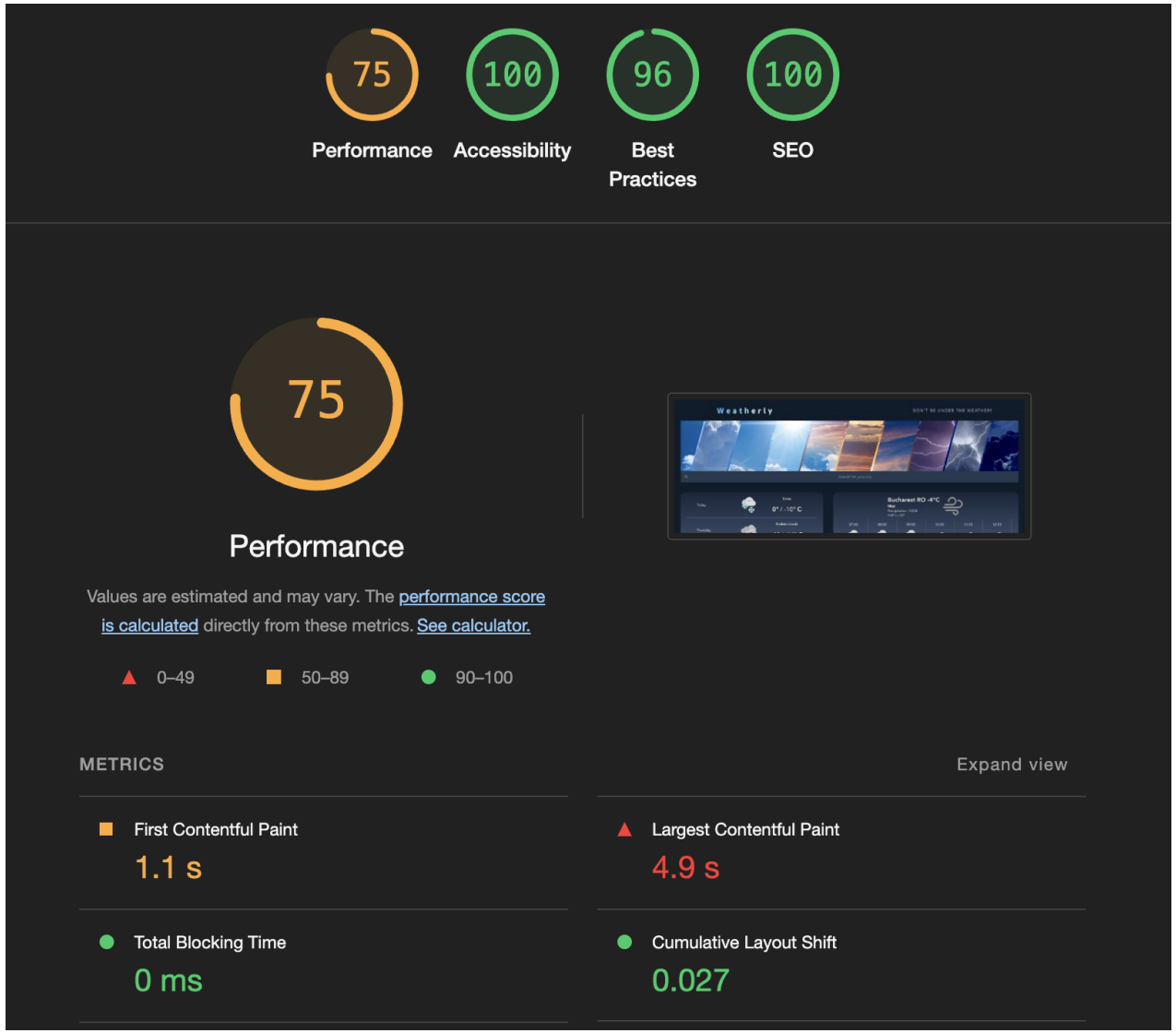
- The setup with all the appropiate options in the configuration files was somewhat of a hassle;
- Unit testing for every component and helper function;
- Covering edge cases with mocks, such as no data, invalid inputs, API failures, network failures.

Demo

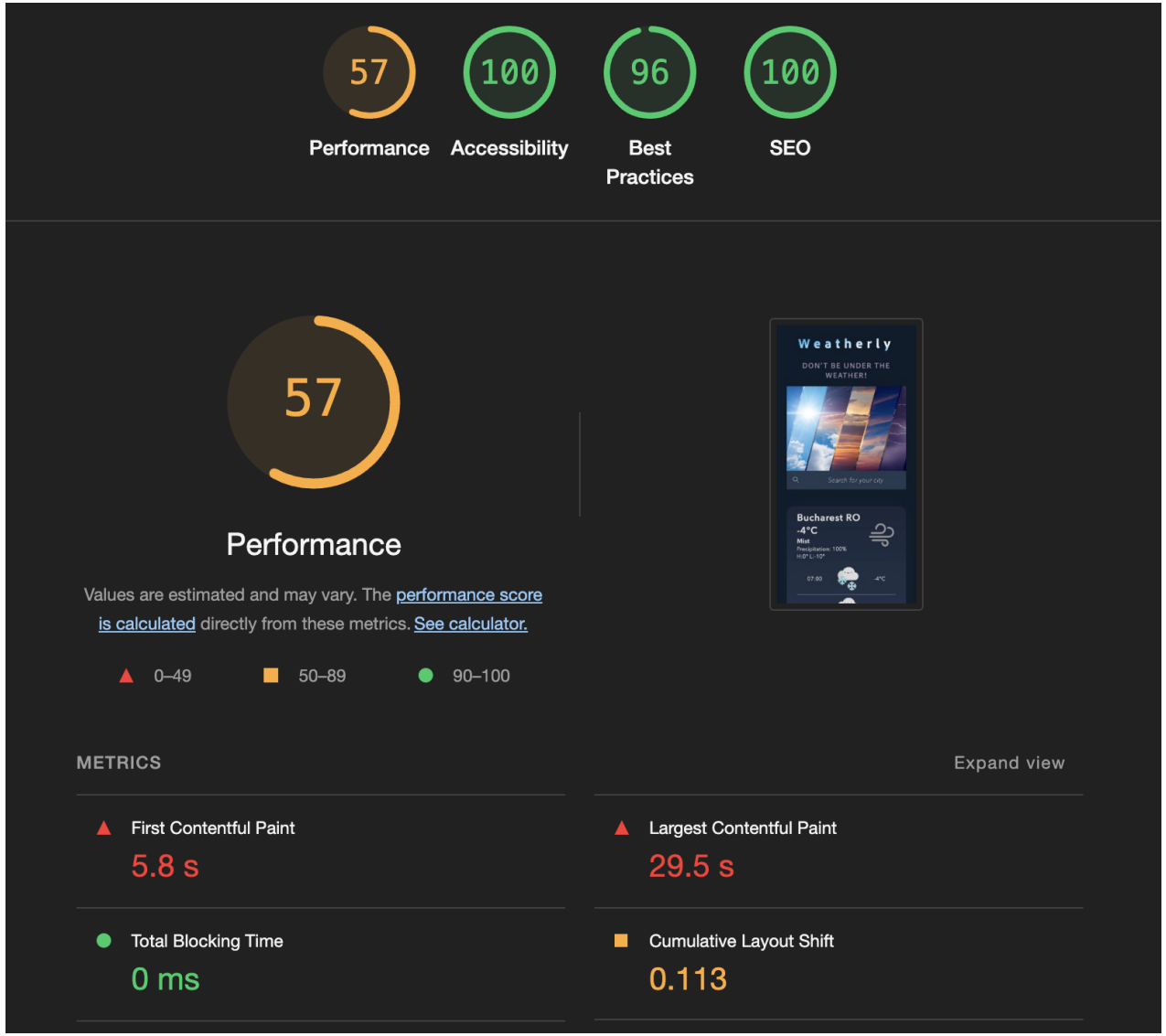
Results



Lighthouse - Desktop



Lighthouse - Mobile



Istanbul Report – Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	93.35	89.74	96.61	94.36	
src	82.85	84.61	91.66	82.6	
App.tsx	90.9	83.33	87.5	90.74	24-27,55
ErrorBoundary.tsx	100	100	100	100	
main.tsx	0	100	100	0	1-7
setupTests.ts	0	100	100	0	1
src/components/CurrentWeather	100	100	100	100	
CurrentWeather.tsx	100	100	100	100	
src/components/CurrentWeather/WeatherDetailsList	100	100	100	100	
WeatherDetailsList.tsx	100	100	100	100	
src/components/DailyForecast	100	100	100	100	
DailyForecast.tsx	100	100	100	100	
src/components/DailyForecast/ForecastCard	100	100	100	100	
ForecastCard.tsx	100	100	100	100	
src/components/Footer	100	100	100	100	
Footer.tsx	100	100	100	100	
src/components/Footer/SocialMediaList	100	100	100	100	
SocialMediaList.tsx	100	100	100	100	
src/components/Header	100	100	100	100	
Header.tsx	100	100	100	100	
src/components/Header/SearchBar	100	100	100	100	
SearchBar.tsx	100	100	100	100	
src/components/LoadingSpinner	100	100	100	100	
LoadingSpinner.tsx	100	100	100	100	
src/components/TodayForecast	100	100	100	100	
TodayForecast.tsx	100	100	100	100	
src/components/TodayForecast/HourlyForecastCard	100	100	100	100	
HourlyForecastCard.tsx	100	100	100	100	
src/components/TodayForecast/RightNowForecastCard	100	100	100	100	
RightNowForecastCard.tsx	100	100	100	100	
src/utils	92.72	88.67	96	95.83	
const.ts	100	100	100	100	
fetch-data.ts	92.68	90.9	91.66	92.3	50-51,90
helpers.ts	91.8	88.09	100	98.03	64
types.ts	100	100	100	100	

```
Test Suites: 15 passed, 15 total
Tests:       68 passed, 68 total
Snapshots:   1 passed, 1 total
Time:        3.778 s
```

Next steps/ Future work

- *Autocomplete suggestions* in the search bar, displaying the city, country, and, for the US, the state;
- *Performance improvement* – reverse geocoding is slow;
- Toggle for Celsius;
- Hooks could be improved, media queries could be re-thought from bottom-up, etc…

Q & A

Adobe