OPEN WINDOW

frots

project presentation

development 200 term 1

wolf botha | 21100255

developer

wolf botha

2 1 1 0 0 2 5 5

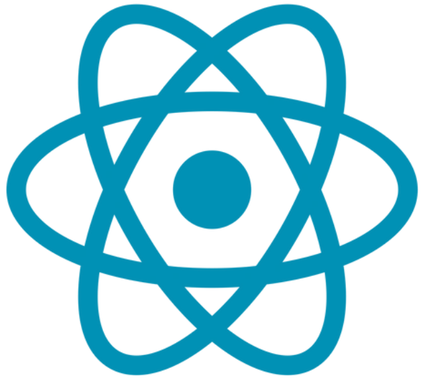
fun fact: my name, Wolf, is an abbreviated version of the common German name, “Wolfgang”.

the project

what is fruts?

Fruts is an interactive, chart-based website that provides detailed insights into the nutrition & price trends of various fruits. Apart from fun facts and info about each fruit on the home page, the platform offers two additional pages; the first, a comparative view where 2 fruits' nutritional info are compared with engaging visualisations. Secondly, users can explore the historical pricing of fruits in Europe on the timeline page. The site is a great tool for health enthusiasts, nutritionists, and anyone with a love for fruit.

tech stack A



react

allows for modular and maintainable code, with a virtual dom for a fast user experience

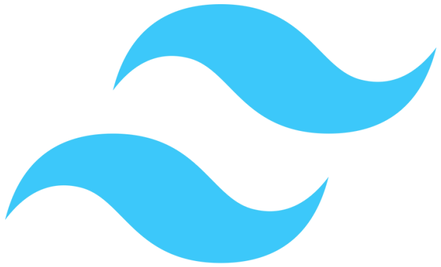
bootstrap

multiple pre-built, responsive -designed components, that reduces and simplifies development

chart.js

selected for its lightweight and visually appealing chart components, that integrate well with React.

tech stack B

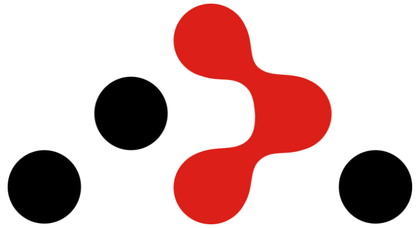


tailwind

better control over stylisation and allows for rapid but flexible styling, especially alongside bootstrap.

axios

used for making HTTP requests to fetch data from the two selected APIs.



react router

implemented for dynamic routing between different pages, providing a smooth single-page experience.

purpose

why?

the website was created with goal of providing easy and fun nutritional info to the public, which not only increases awareness of different fruit, but also promotes healthier eating habits.

goals met

- an educational tool that provides detailed nutritional info & price trends (using APIs)
- use a technology (chart.js) to provide a fun & engaging user experience with visualisations.
- multi-page application that is accessible and simple to use.

requirements

meeting the brief

after selecting the appropriate apis and doing researching and planning, a landing page (with a breakdown of info and contextualisation), a comparison page (comparing 2 interactive objects, represented by 3 chart elements) and a timeline page (with a line graph with 5+ properties of objects displayed) have been created.

the platform uses component based UI, with react router navigation, axios calls, a unified visual aesthetic that has been public on a public GitHub repository.

functionality

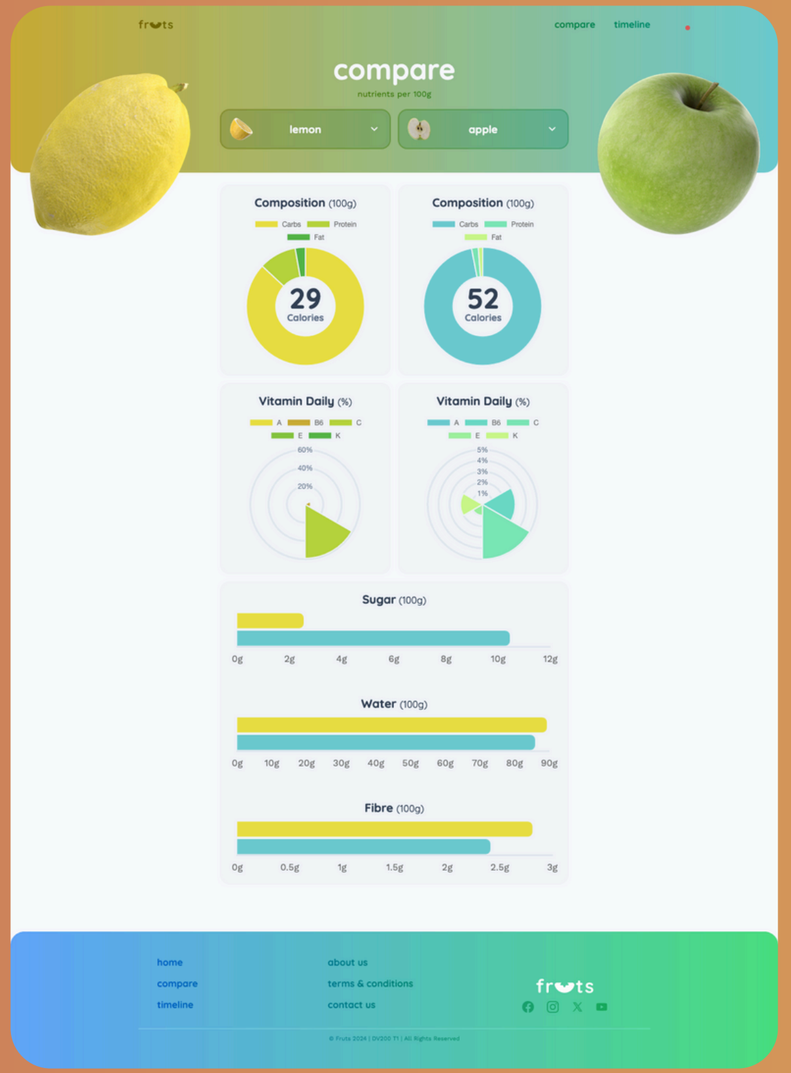
edamam api

this api is used on the comparison page, to get nutritional insights about 2 x different fruits.

An example of a section of a response:

Each key indicates another nutrient - such as 9.1mg of magnesium.

"CA": {
"label": "Calcium, Ca",
"quantity": 10.92,
"unit": "mg"
},
"MG": {
"label": "Magnesium, Mg",
"quantity": 9.1,
"unit": "mg"
},
"K": {
"label": "Potassium, K",
"quantity": 194.74,
"unit": "mg"
},
"FE": {

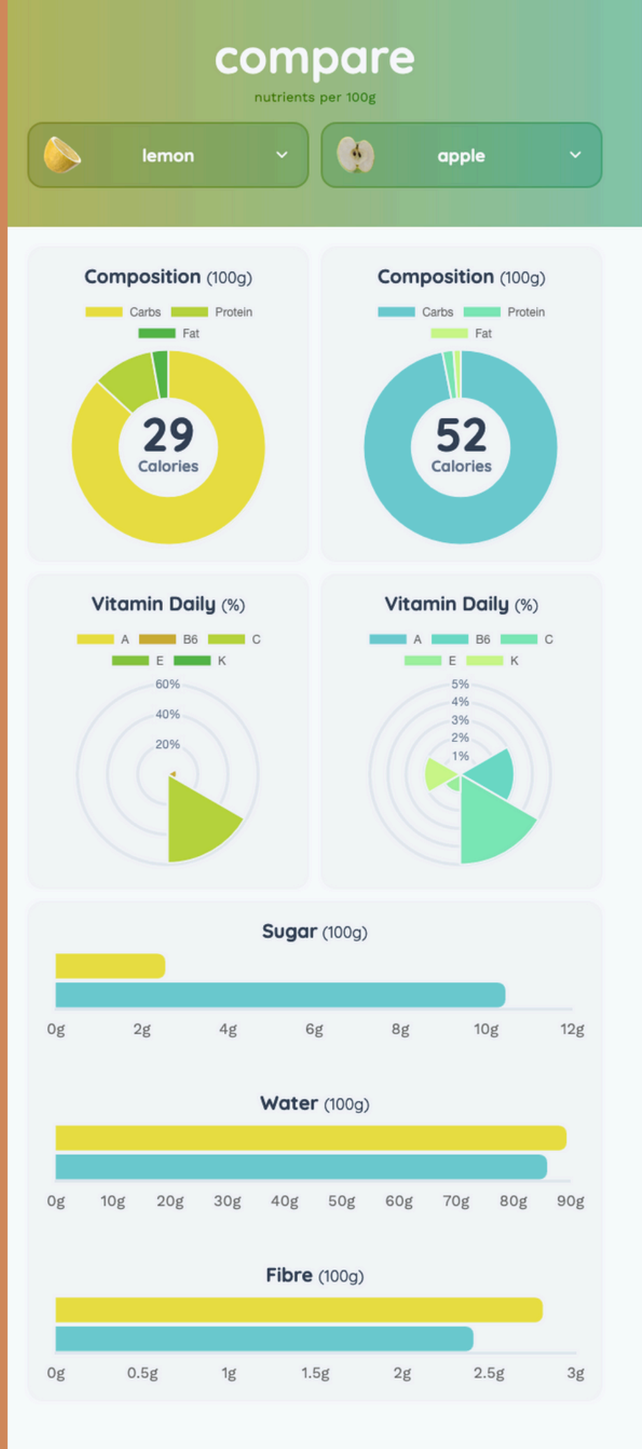


functionality

edamam api

when the dropdown changes to a different fruit, the data updates, and subsequently, the graphs.

When a single fruit changes, only that respective fruit's graphs refresh, except for the bottom bar graphs, which contain data about both fruits, and thus, require an update each time the fruits compared are changed.



functionality

EU Agri-Commissions API

this api is utilised on the timeline page, where the historical price of different fruits are visualised on the line graph.

An example of a section of a response:

In this array of objects, each objects is a data packet that contains info of different countries, fruits, the price of that fruit and during what beginDate & endDate.

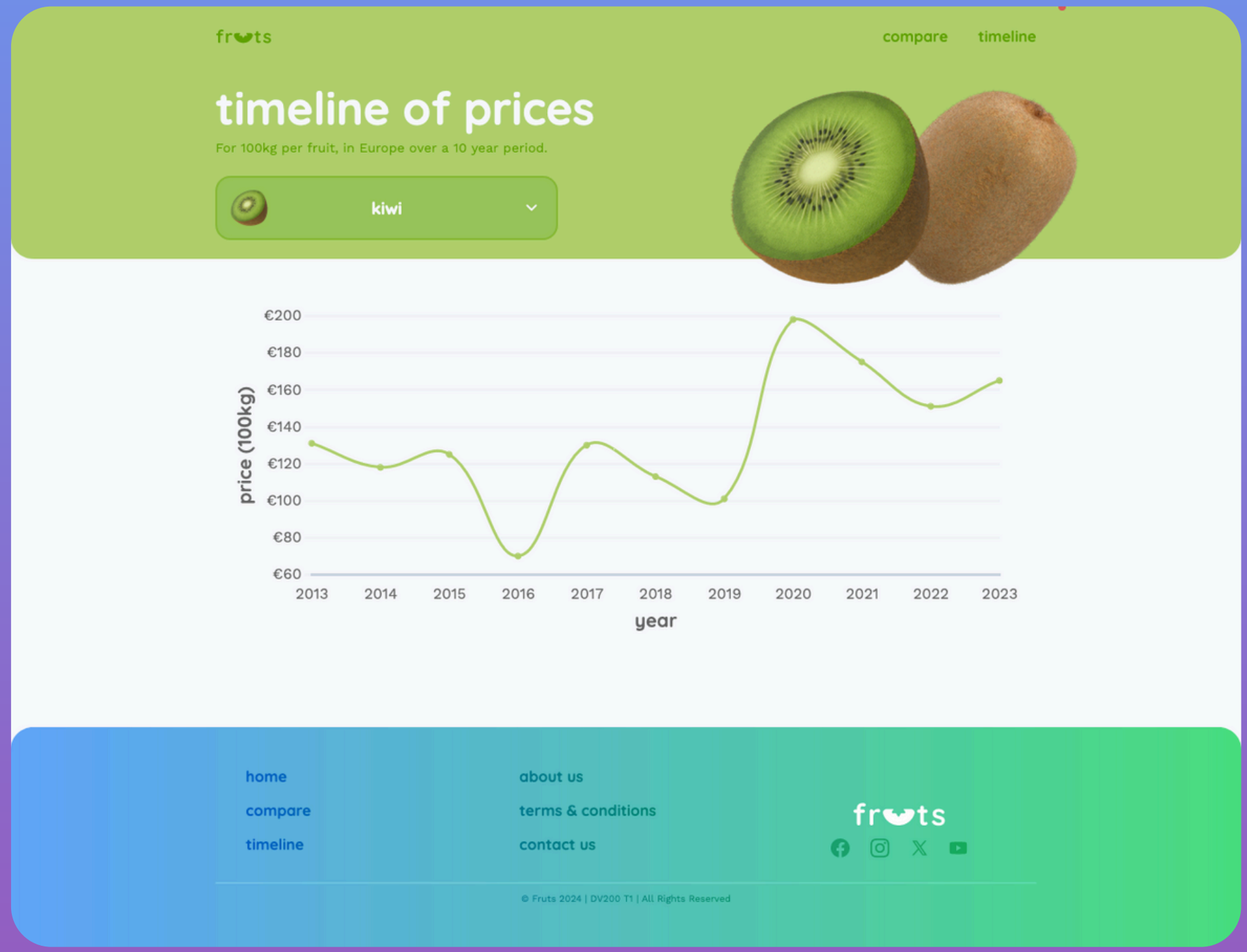
[{
"memberStateCode" : "PT",
"memberStateName" : "Portugal",
"beginDate" : "16/03/2020",
"endDate" : "22/03/2020",
"weekNumber" : 12,
"price" : "€50.00",
"unit" : "100KG",
"product" : "oranges",
"variety" : "Lanelate - Cat. I - Cal. 2-4",
"description" : "Portugal - oranges - Lanelate - Cat. I - Cal. 2-4"
}, {
"memberStateCode" : "PT",
"memberStateName" : "Portugal",
"beginDate" : "16/03/2020",
"endDate" : "22/03/2020",
"weekNumber" : 12,
"price" : "€50.00",
"unit" : "100KG",
"product" : "oranges",
"variety" : "navel",
"description" : "Portugal - oranges - navel"
}, {
"memberStateCode" : "PT",

functionality

EU Agri API

whenever a fruit type is selected from the dropdown, the api is called with the specific fruit, for a single entry of a fruit variant per year - for a total of 10 years.

the aforementioned data is then read through to the graph, which charts the line visually.



challenges & solutions A

data integration

integrating multiple different APIs into information that makes sense to chart.js to visually illustrate the information.

solution

through the implementation of custom backend functions and methods, the data can become standardised and easy to use, especially having the api documentation nearby.

challenges & solutions B

responsive design

ensuring the website’s layout and the chart.js visualisations remain accessible across different screen sizes.

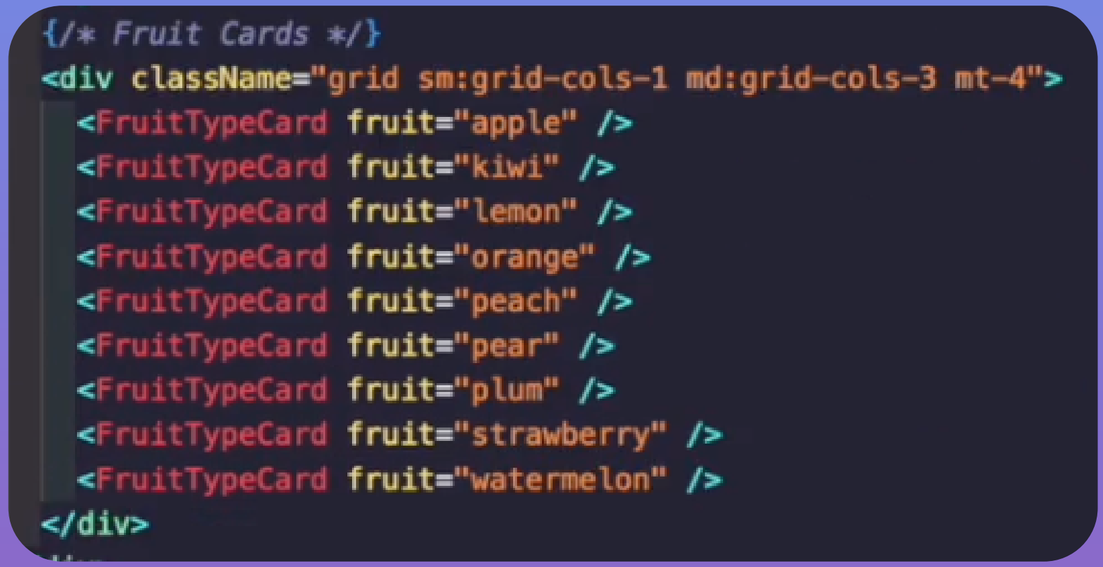
solution

utilising bootstrap and tailwindCSS improved the speed and efficiency of creating a responsive website.

explain code A1

component-based implementation

the “FruitTypeCard” custom component are reused several times on the landing page, each being the same component, but a unique “fruit” property being passed through to it, such as “apple” or “kiwi”.



```
{/* Fruit Cards */}
<div className="grid sm:grid-cols-1 md:grid-cols-3 mt-4">
  <FruitTypeCard fruit="apple" />
  <FruitTypeCard fruit="kiwi" />
  <FruitTypeCard fruit="lemon" />
  <FruitTypeCard fruit="orange" />
  <FruitTypeCard fruit="peach" />
  <FruitTypeCard fruit="pear" />
  <FruitTypeCard fruit="plum" />
  <FruitTypeCard fruit="strawberry" />
  <FruitTypeCard fruit="watermelon" />
</div>
```

explain code A2

component-based implementation

```
return (
  // Full Container (changes to gradient on hover)
  <div
    className={`group flex flex-col items-center transition-all duration-300
  >
    {/* Fruit Image */}
    <div className="flex justify-center group-hover:animate-wiggle">
      <FruitTypeCardImg fruit={props.fruit} />
    </div>

    {/* Name & Description */}
    <h3 className="text-center font-head font-bold p-1">{props.fruit}</h3>
    <FruitTypeCardParag fruit={props.fruit} />
  </div>
);
}
// ------------------------------
```

each property that is passed through, is also passed to the FruitTypeCard's subcomponents, being an image and description.

explain code A3

component-based implementation

finally, inside both the image and description subcomponents, a switch case takes the passed property, and changes the image src or paragraph text, respectively.

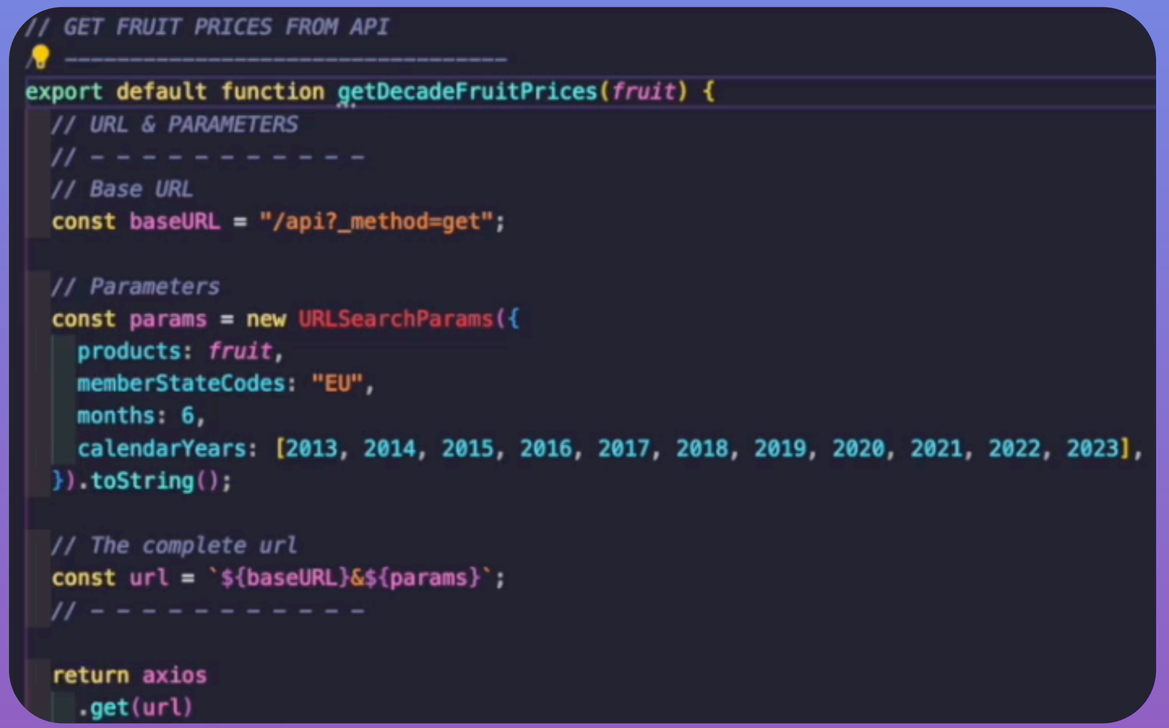
```
// Return different fun facts / information based on fruit prop passed
switch (props.fruit) {
  case "apple":
    description =
      "With fibre and Vitamin C; perfect for heart health and digestion, and keeps the doctors away.";
    break;
  case "kiwi":
    description =
      "Bursting with Vitamin C & K, kiwi is a powerhouse for immune support and gut health.";
    break;
  case "lemon":
    description =
      "High in Vitamin C and flavonoids, great for detoxification and enhancing skin health.";
    break;
  case "orange":
    description =
      "Your zesty dose of Vitamin C, excellent for immune function and antioxidant protection.";
    break;
  case "peach":
    description =
      "Juicy and sweet, peaches offer vitamins A and C, promoting skin health and digestion.";
    break;
  case "pear":
```

explain

code B1

populating line chart with api data

the EU agri-data api does not require any authentication, so simply using axios with a url (customised with parameters), we can easily get info in the form of a JSON file.



```
// GET FRUIT PRICES FROM API
// ------------------------------
export default function getDecadeFruitPrices(fruit) {
  // URL & PARAMETERS
  // - - - - - - - - - - - -
  // Base URL
  const baseURL = "/api?_method=get";

  // Parameters
  const params = new URLSearchParams({
    products: fruit,
    memberStateCodes: "EU",
    months: 6,
    calendarYears: [2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023],
  }).toString();

  // The complete url
  const url = `${baseURL}&${params}`;
  // - - - - - - - - - - - -

  return axios
    .get(url)
```

explain

code B2

populating line chart with api data

the JSON file is then formatted with custom service functions, that filter the data to only a single entry per year, and only a single fruit variant.

```
    memberStateCodes: "EU",
    months: 6,
    calendarYears: [2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023],
  }).toString();

  // The complete url
  const url = `${baseURL}&${params}`;
  // - - - - - - - - - - -

  return axios
    .get(url)
    .then((res) => {
      const filteredData = filterTo1PerYear(filterToSingleVariety(res.data));
      return filteredData;
    })
    .catch((err) => {
      console.log(`EuAgriData Error:`, err);
      throw err;
    });
}
// ------------------------------
```

explain code B3

populating line chart with api data

```
// EFFECT
// - - - - - - - - - - -
// When the state of dropdown is changed
useEffect(() => {
  // Start the loading screen
  setIsLoading(true);
  // Get the prices from the API
  getDecadeFruitPrices(fruitNameToPlural(selectedFruit.name))
    .then((arr) => {
      // Store in state for graph
      setFruitPriceData(arr);
      setIsLoading(false);
    })
    .catch((err) => {
      console.log("Failed to get fruit prices: ", err);
    });
}, [selectedFruit]);
// - - - - - - - - - - -
```

on the timeline page (the page which the chart is on), whenever the dropdown changes, the useEffect hook is triggered. this will call the previously mentioned “getDecadeFruitPrices” function, and store the returned array in the “FruitPriceData” state. the state’s data is then passed through to the chart component on that page.

explain

code B4

populating line chart with api data

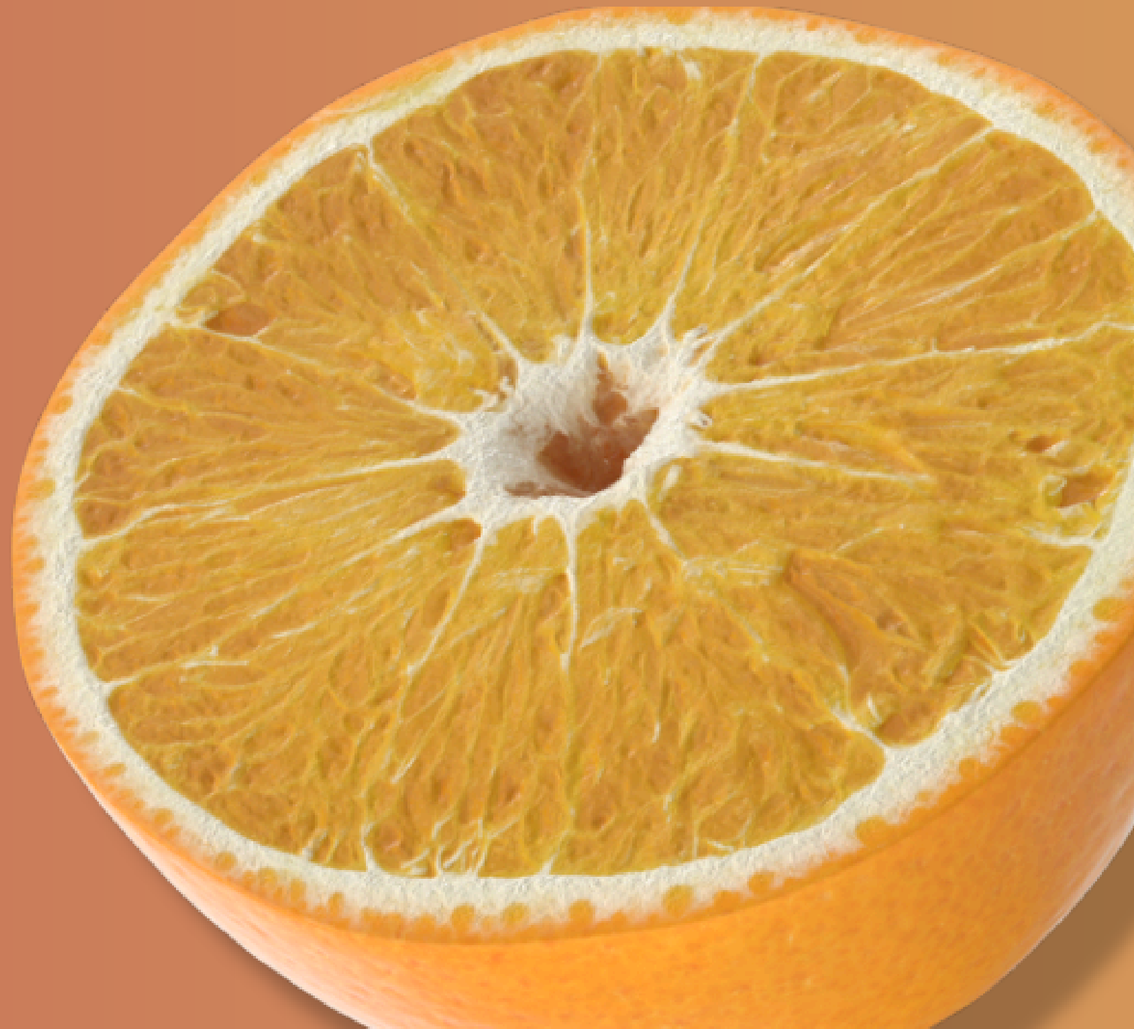
inside the line chart component, another useEffect refreshes the chart when data changes, and will map each object's price in the array to a variable called "chartData", which is what the chart will use to populate its points on the line, numerically.

```
// When data changes
useEffect(() => {
  // Map props.dataArr properties & extract the prices
  if (props.dataArr) {
    const newPricesArr = props.dataArr.map((obj) => obj.price);
    const chartData = euroPricesToNum(newPricesArr);
    setPricesArr(chartData);
  }
}, [props.dataArr]);
// - - - - - - - - - - - - - -
```

live demo

Let’s take a step-by-step walkthrough of the website,
highlighting all the features & functionalities.

thank you



OPEN WINDOW

wolf botha 21100255