Lecture 18: APIs & Frontend

AC215

Shivas Jayaram

Announcements

- **Showcase Info Form - due today - 11/12**

  https://forms.gle/CewUpMnmYq2BxupW6

- **Optional React Zoom Session -**
  Friday 11/15 - Time TBD (will be recorded)

- **Late Days** - 2 days maximum for Milestone 4 or HW3 (No need to send an email) subject to your attendance record.

- No late days for final project Milestone 5

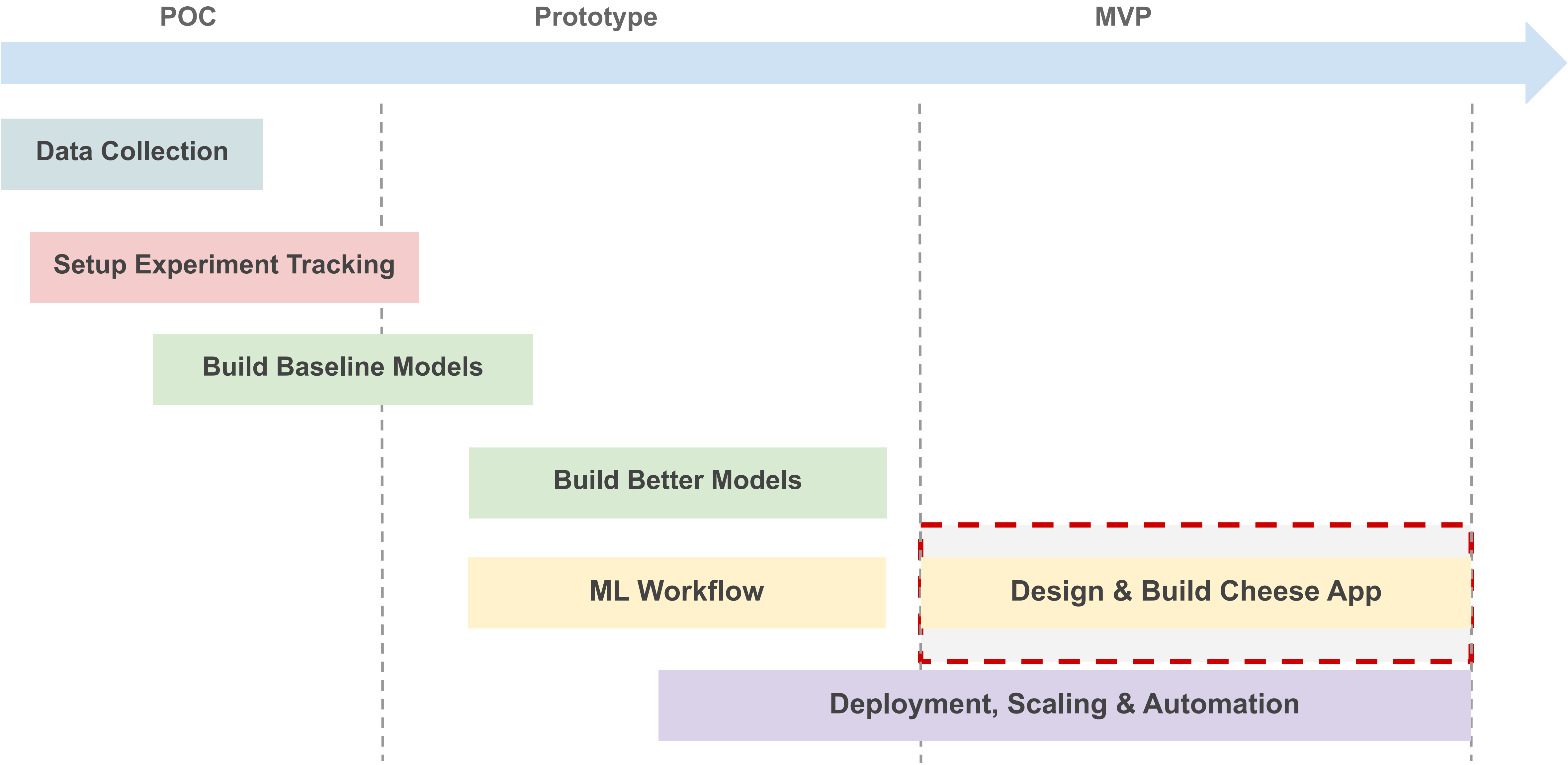
Outline

1. Recap
2. APIs
3. Frontend (Simple)
4. Frontend Frameworks
5. Frontend App (React)

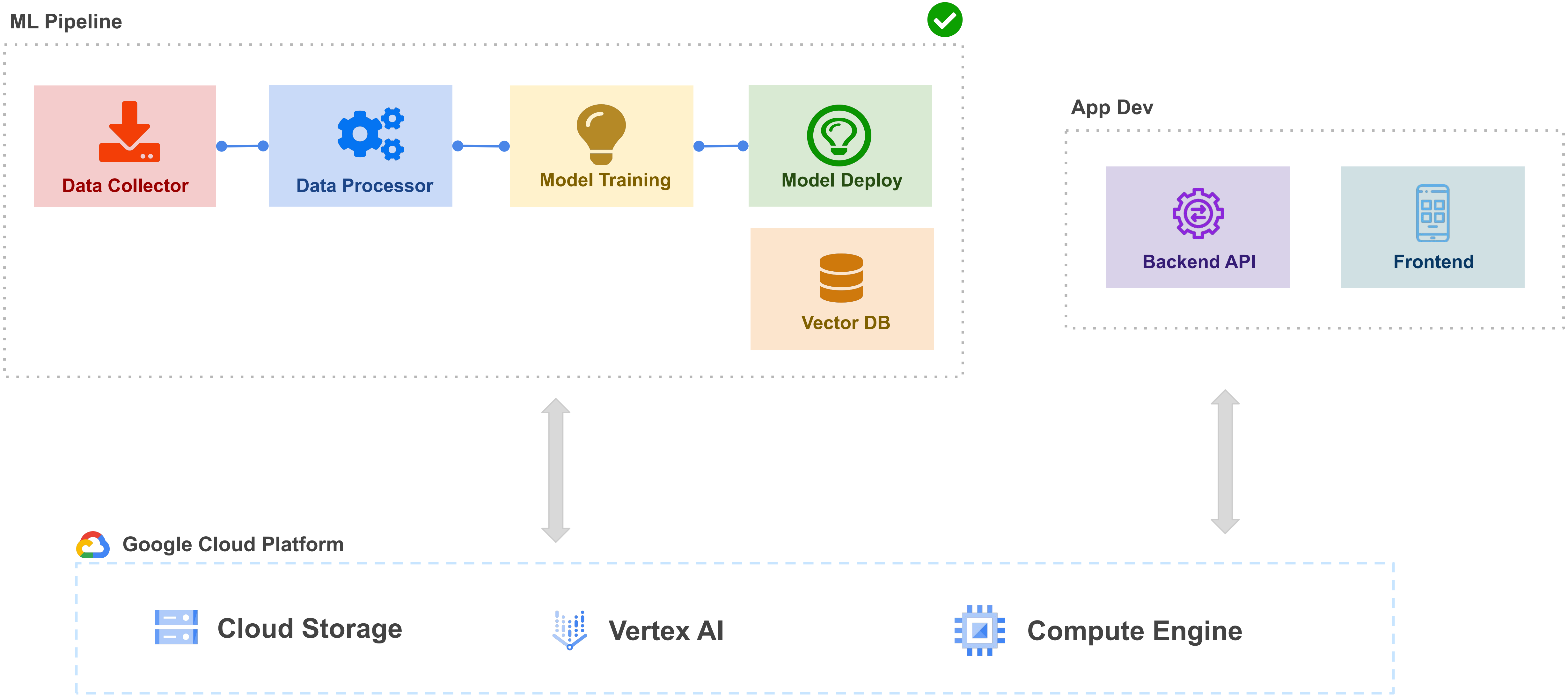
Outline

1. **Recap**
2. APIs
3. Frontend (Simple)
4. Frontend Frameworks
5. Frontend App (React)

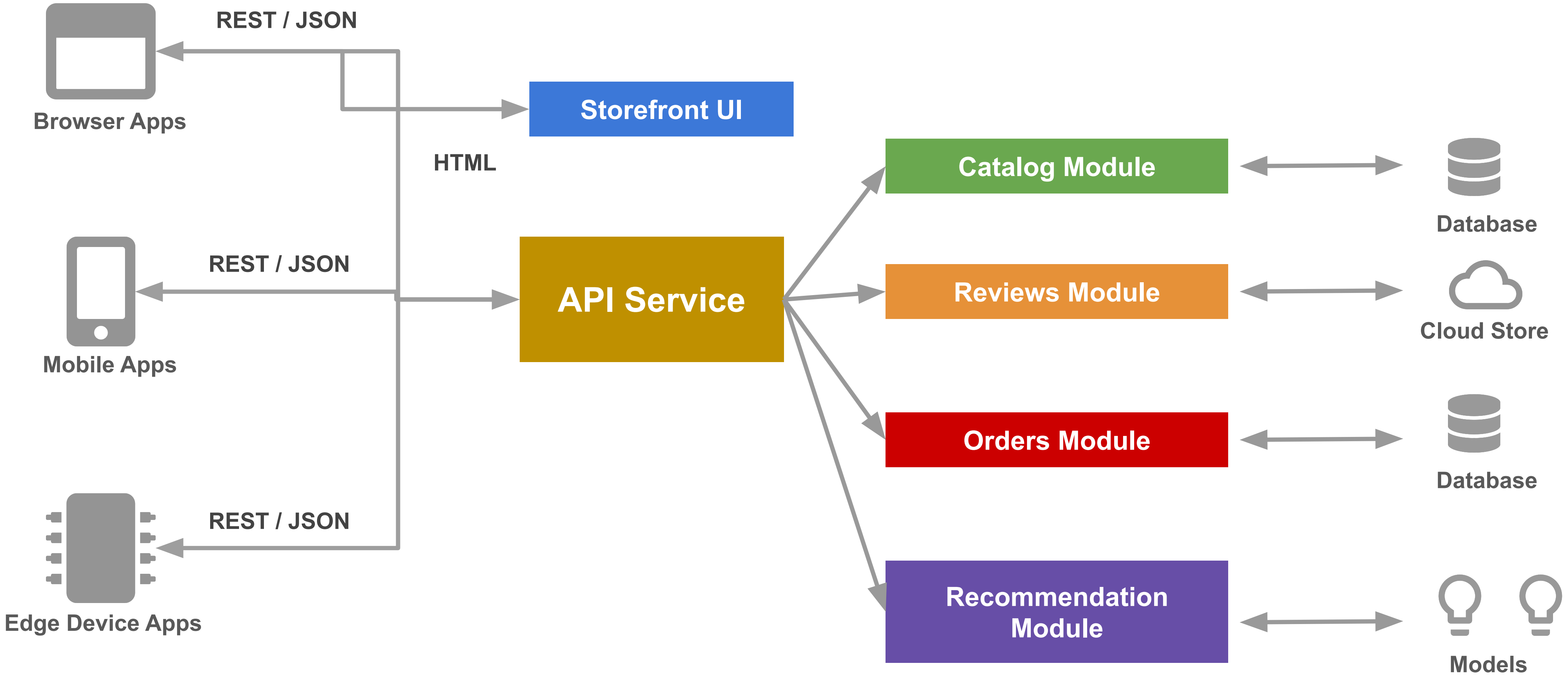
Recap: Cheese App Status


POC
Prototype
MVP
Data Collection
Setup Experiment Tracking
Build Baseline Models
Build Better Models
ML Workflow
Design & Build Cheese App
Deployment, Scaling & Automation

Recap: Cheese App Development



Recap: Microservice Architecture

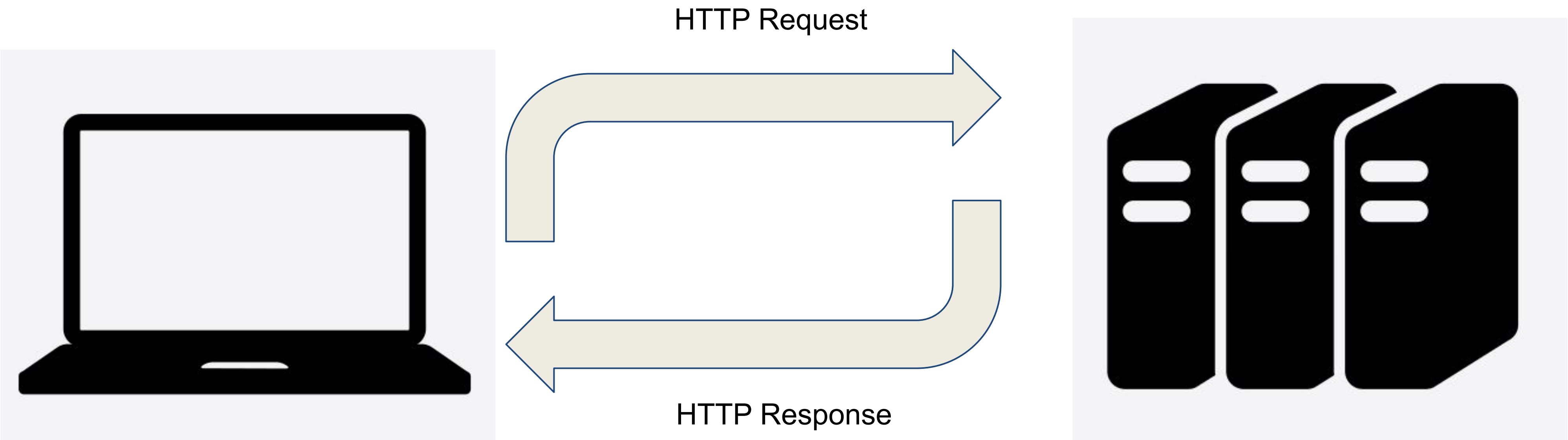


Outline

1. Recap
2. **APIs**
3. App Frontend (Simple)
4. Frontend Frameworks
5. Frontend App (React)

Review: What is HTTP?

- HyperText Transfer Protocol: method for **transporting information** where **client** (such as a web browser) makes **request** and web **server** issues a **response**
  - content can be anything from text to images to video
- HTTPS: encryption for **secure** communication over network
- Analogy: post office



HTTP Request

HTTP Response

Review: What is a port?

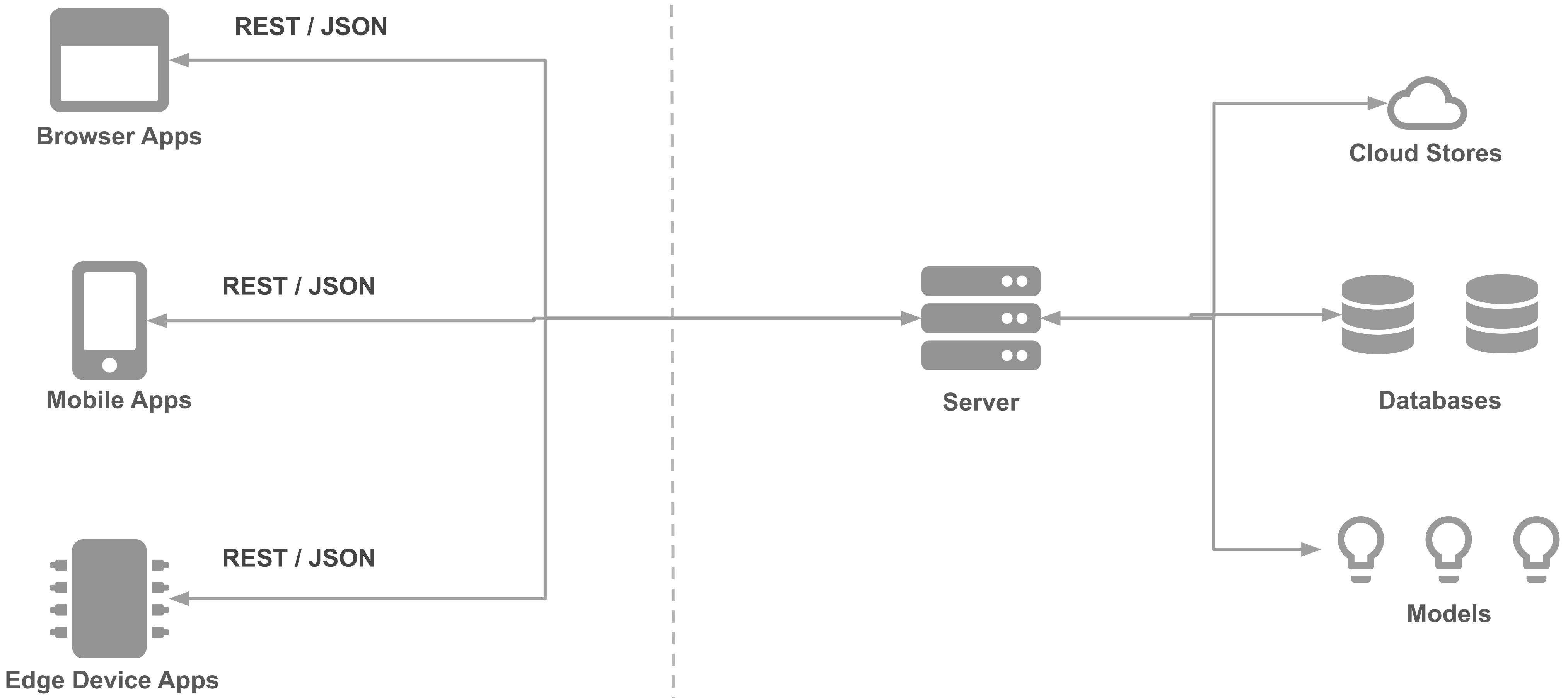
- **communication endpoint** where network connections start and end
- lets **computers differentiate** between different kinds of **data** (emails, webpages, etc.)
    - Port 22 = SSH
    - Port 25 = SMTP (email)
    - Port 80 = HTTP
    - Port 443 = HTTPS

What is an API

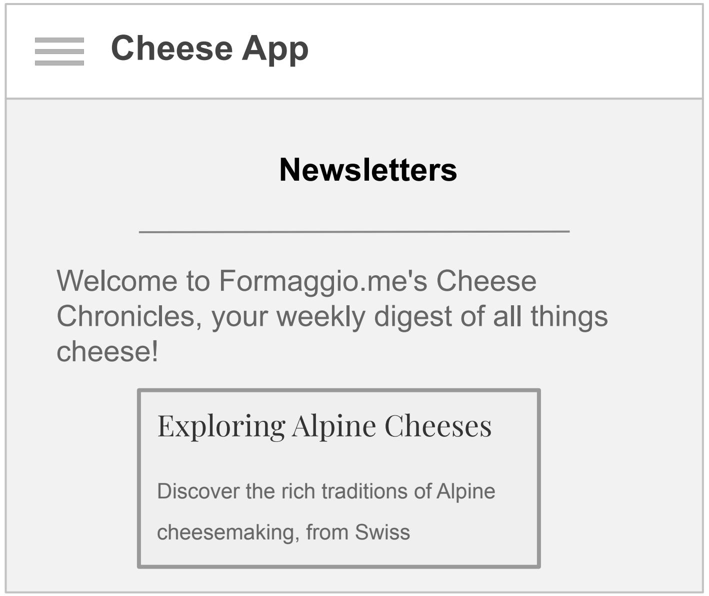
- API is **Application Programming Interface**
- **Web API** is an API that can be access using HTTP/S
- A **REST API** is a Web API that follows the HTTP method constraints - get, post, put, delete
- We will use **FastAPI** a Python framework to build REST APIs

APIs

We will be using the term **API** to refer to REST API, which will be used to connect to various components



Review: Screenflow & Wireframes



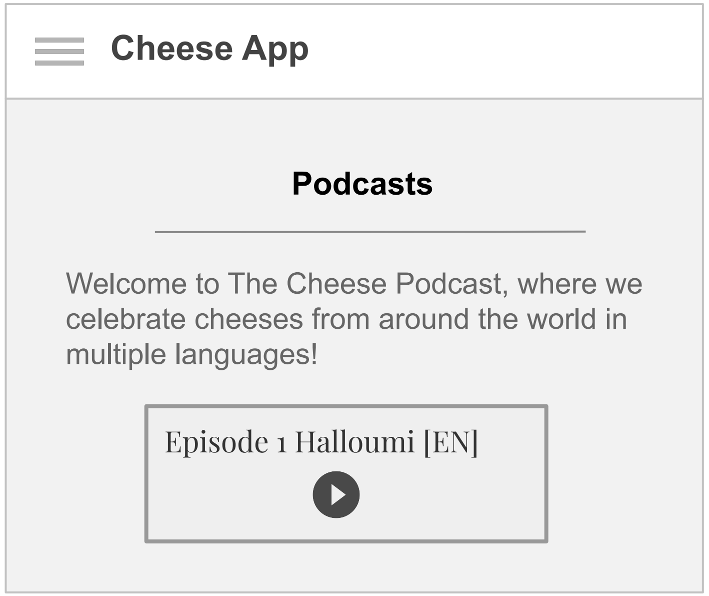
Cheese App

Newsletters

Welcome to Formaggio.me's Cheese Chronicles, your weekly digest of all things cheese!

Exploring Alpine Cheeses

Discover the rich traditions of Alpine cheesemaking, from Swiss

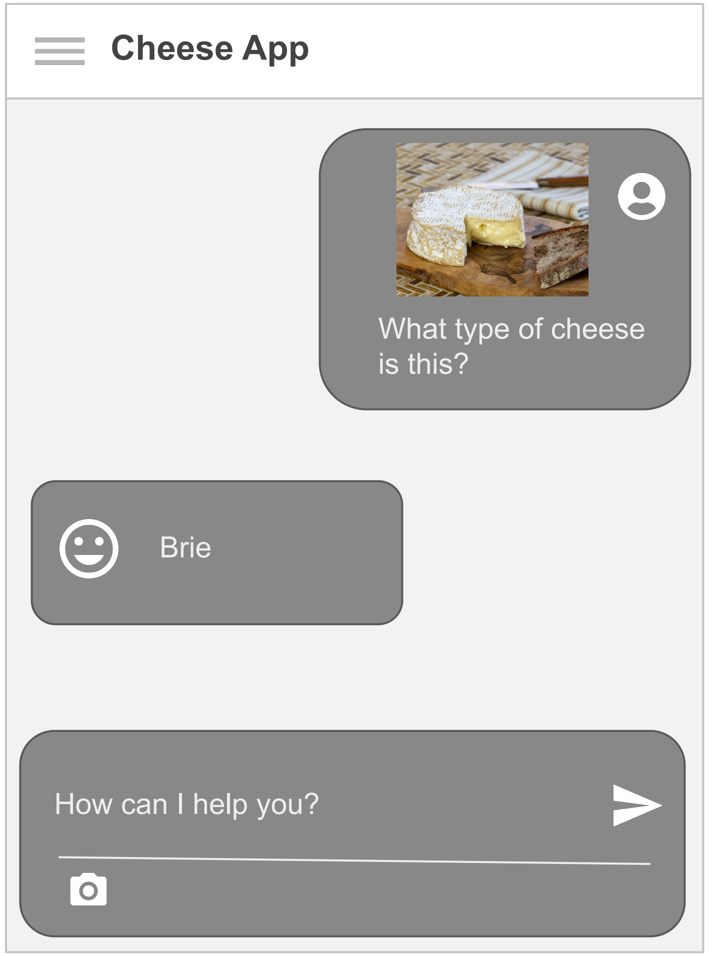


Cheese App

Podcasts

Welcome to The Cheese Podcast, where we celebrate cheeses from around the world in multiple languages!

Episode 1 Halloumi [EN] ▶



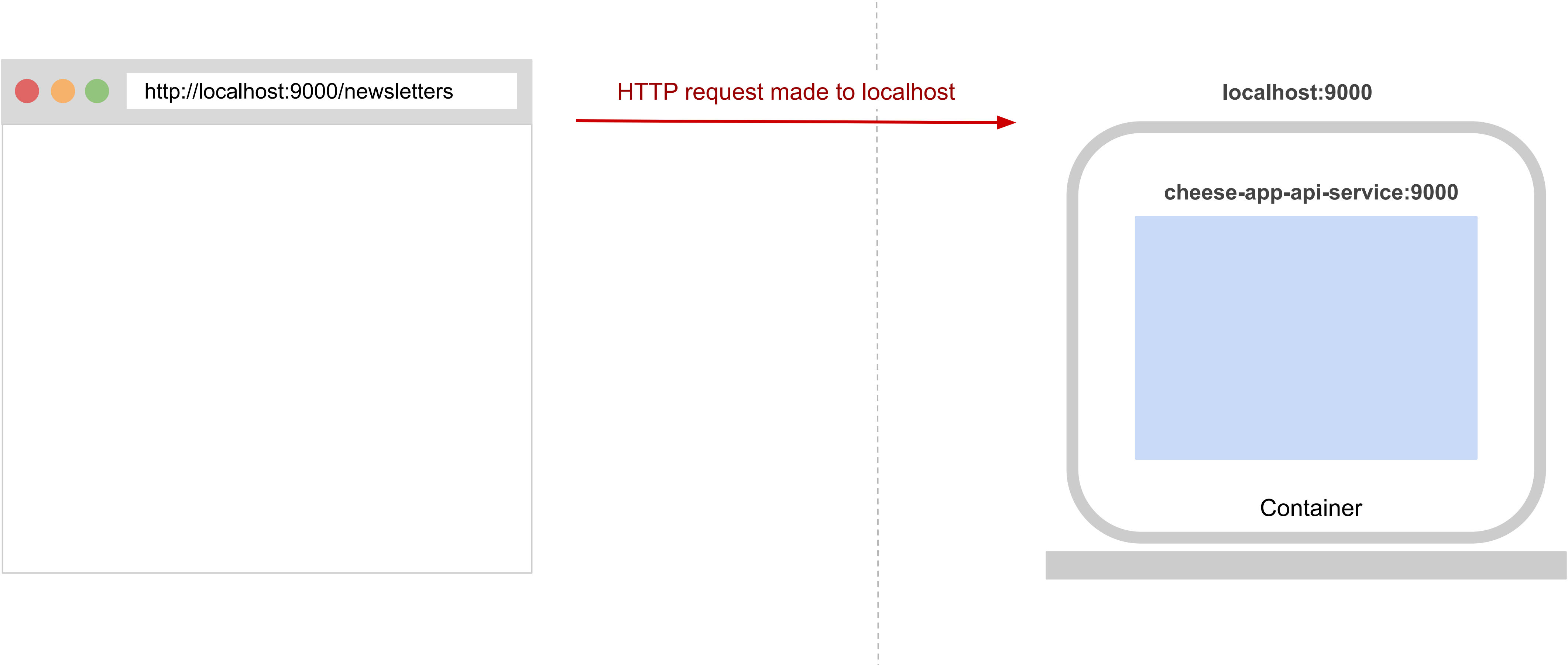
Cheese App

What type of cheese is this?

Brie

How can I help you?

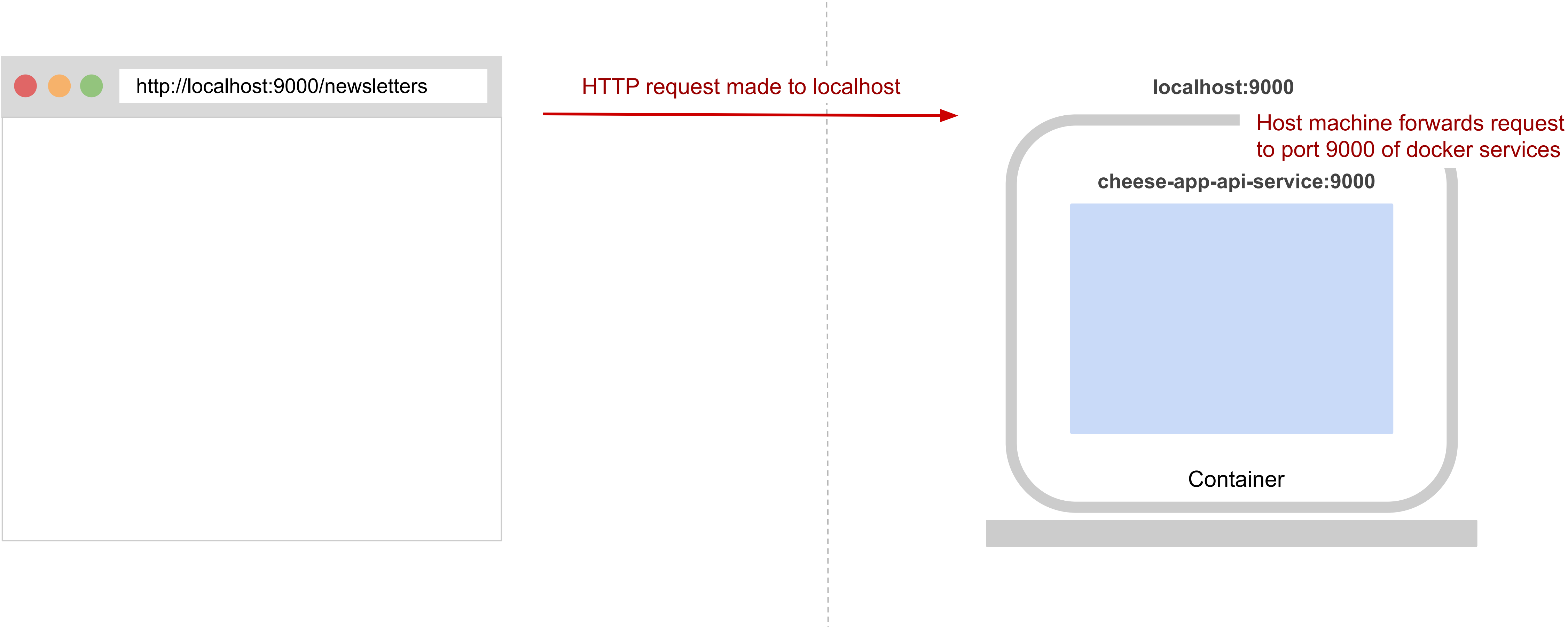
How does an API work


http://localhost:9000/newsletters
HTTP request made to localhost
localhost:9000
cheese-app-api-service:9000
Container

Browser

Local computer / Server

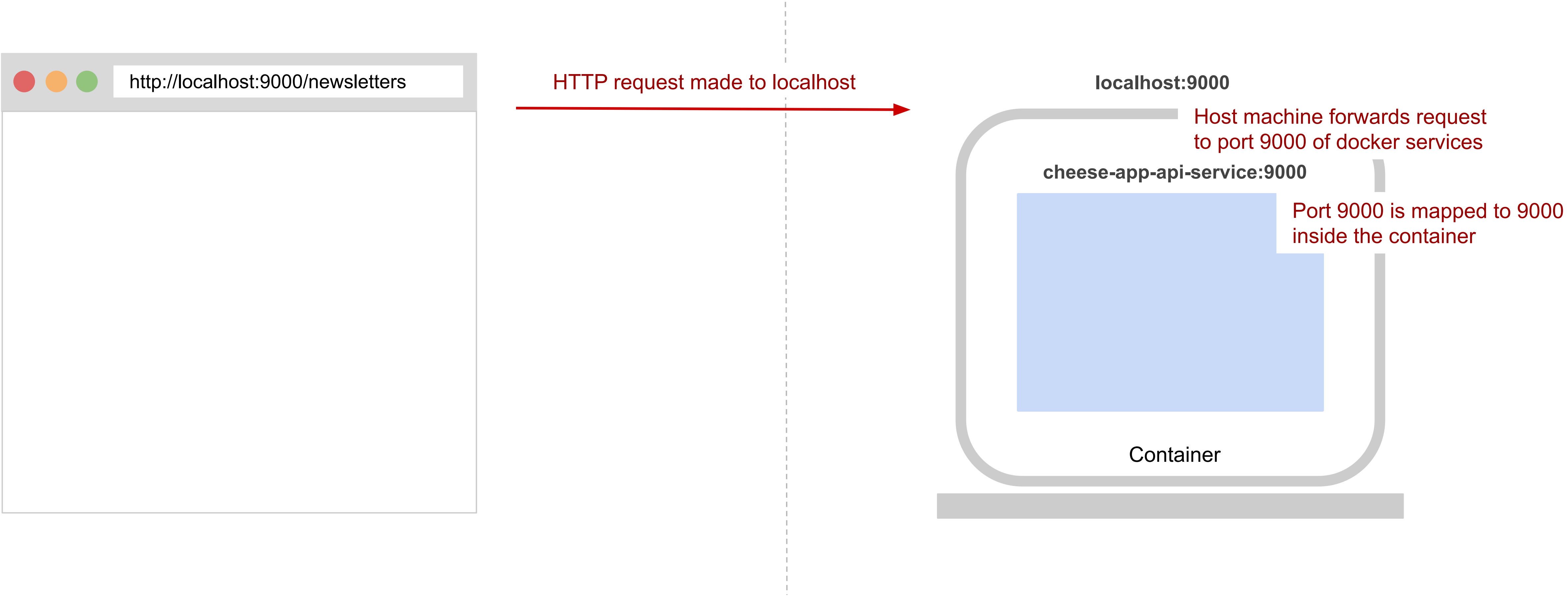
How does an API work


http://localhost:9000/newsletters
HTTP request made to localhost
localhost:9000
Host machine forwards request to port 9000 of docker services
cheese-app-api-service:9000
Container

Browser

Local computer / Server

How does an API work

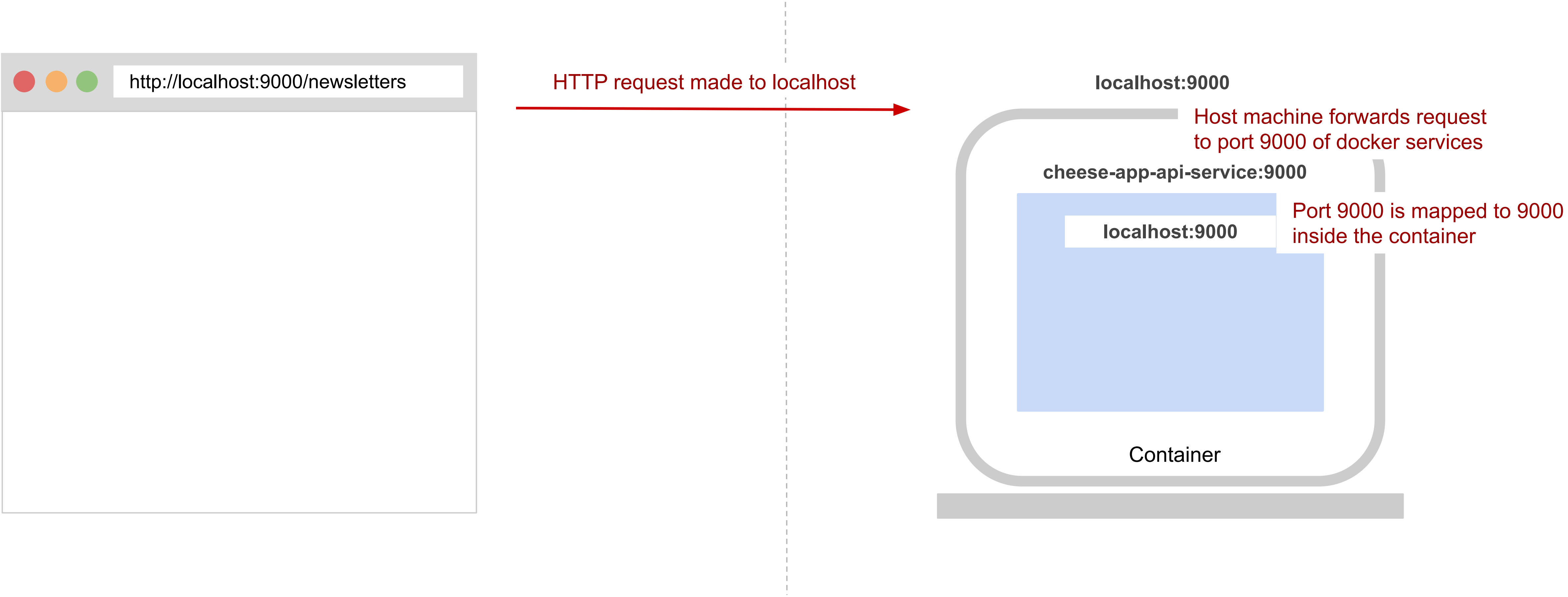


http://localhost:9000/newsletters
HTTP request made to localhost
localhost:9000
Host machine forwards request to port 9000 of docker services
cheese-app-api-service:9000
Port 9000 is mapped to 9000 inside the container
Container

Browser

Local computer / Server

How does an API work

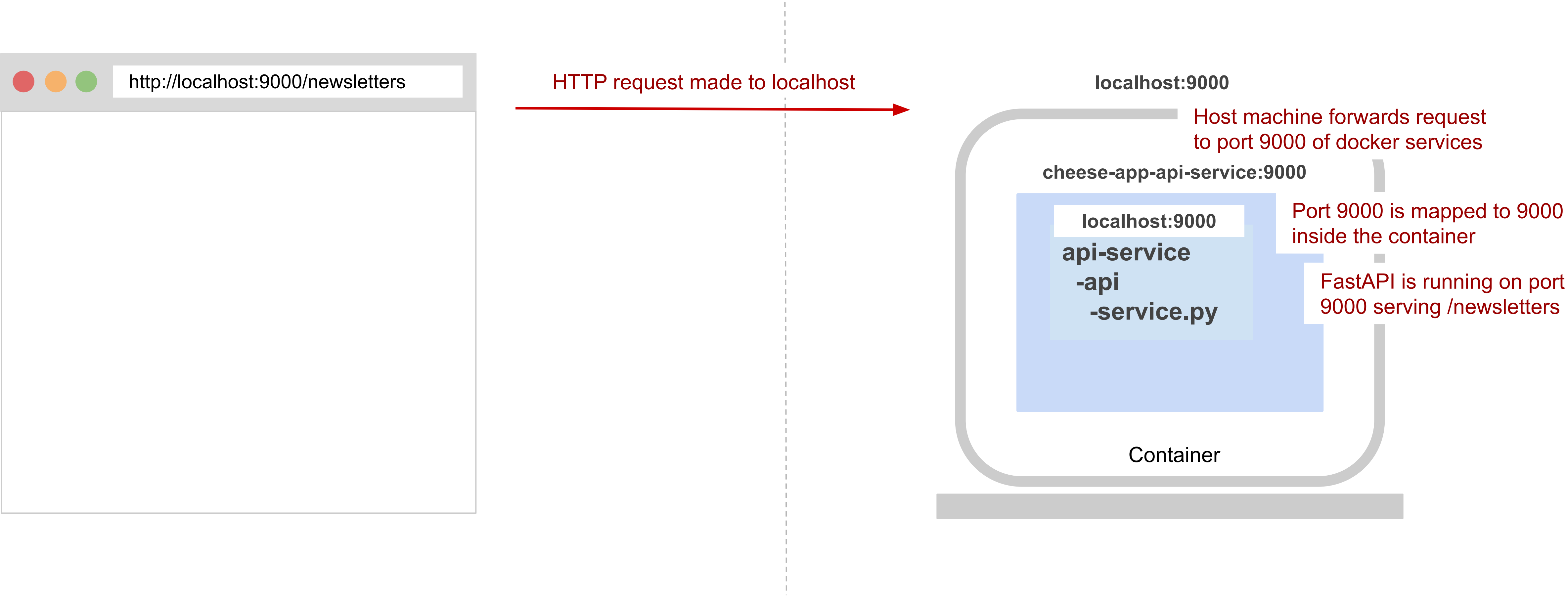


http://localhost:9000/newsletters
HTTP request made to localhost
localhost:9000
Host machine forwards request to port 9000 of docker services
cheese-app-api-service:9000
localhost:9000
Port 9000 is mapped to 9000 inside the container
Container

Browser

Local computer / Server

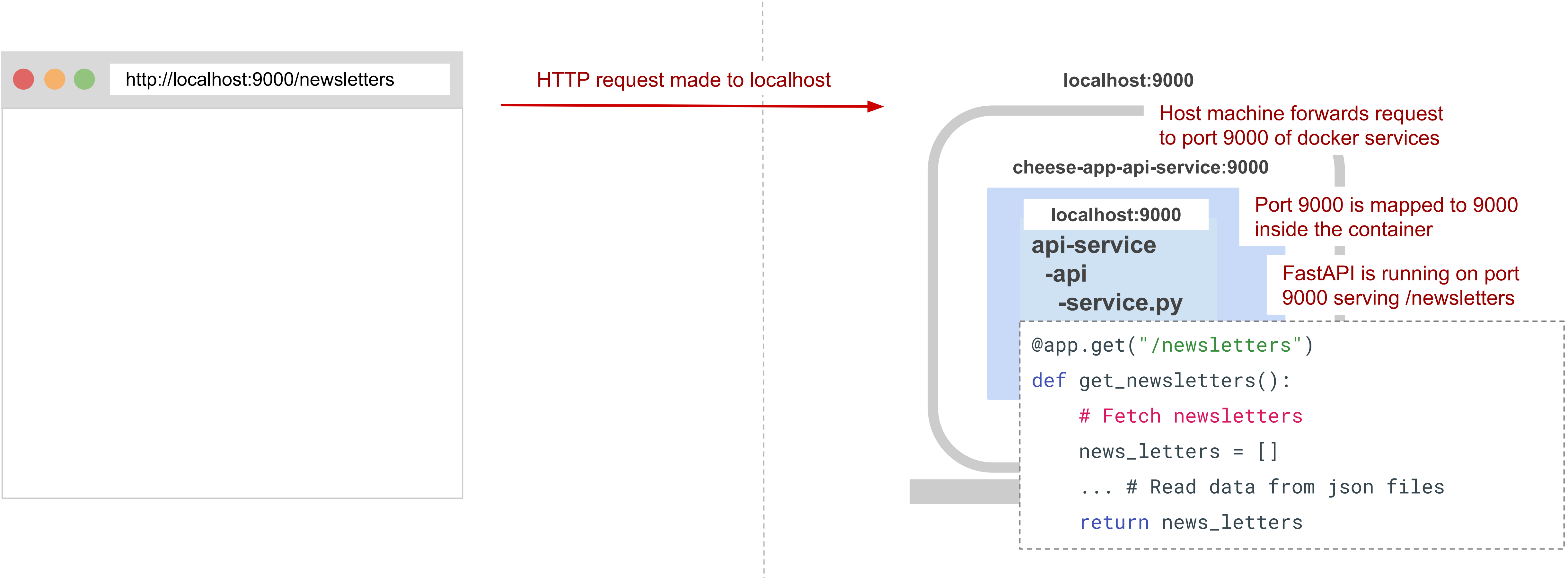
How does an API work


http://localhost:9000/newsletters
HTTP request made to localhost
localhost:9000
Host machine forwards request to port 9000 of docker services
cheese-app-api-service:9000
localhost:9000
api-service
-api
-service.py
Port 9000 is mapped to 9000 inside the container
FastAPI is running on port 9000 serving /newsletters
Container

Browser

Local computer / Server

How does an API work


http://localhost:9000/newsletters
localhost:9000
cheese-app-api-service:9000
localhost:9000
api-service
-api
-service.py

HTTP request made to localhost

Host machine forwards request to port 9000 of docker services

Port 9000 is mapped to 9000 inside the container

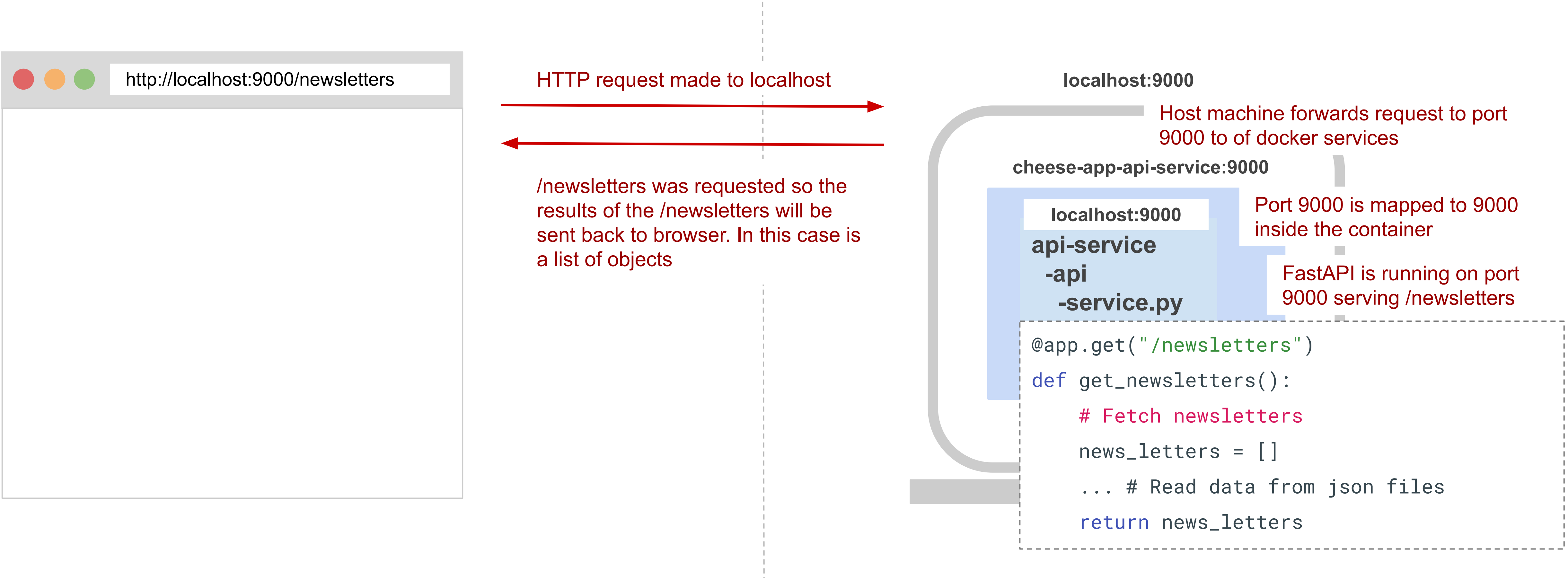
FastAPI is running on port 9000 serving /newsletters

```
@app.get("/newsletters")
def get_newsletters():
    # Fetch newsletters
    news_letters = []
    ... # Read data from json files
    return news_letters
```

Browser

Local computer / Server

How does an API work



http://localhost:9000/newsletters

HTTP request made to localhost

/newsletters was requested so the results of the /newsletters will be sent back to browser. In this case is a list of objects

localhost:9000

Host machine forwards request to port 9000 to of docker services

cheese-app-api-service:9000

Port 9000 is mapped to 9000 inside the container

localhost:9000

api-service
-api
-service.py

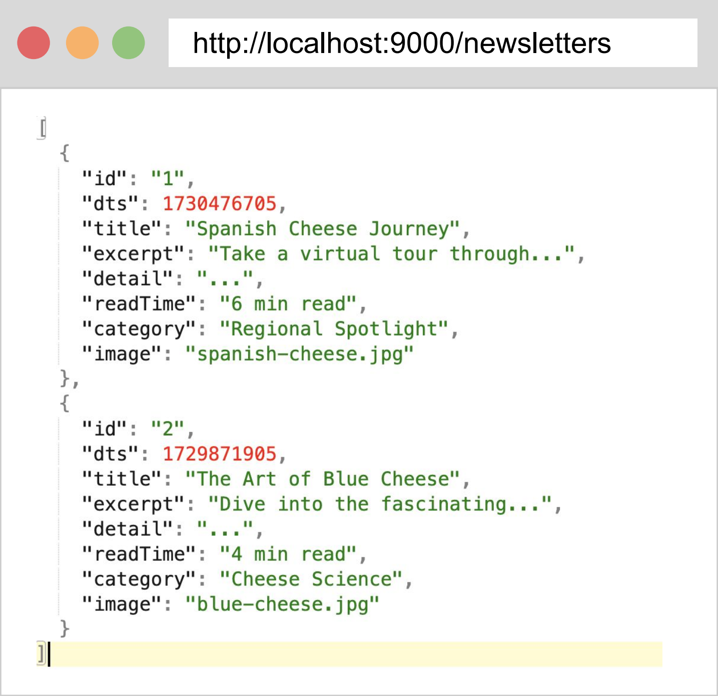
FastAPI is running on port 9000 serving /newsletters

```
@app.get("/newsletters")
def get_newsletters():
    # Fetch newsletters
    news_letters = []
    ... # Read data from json files
    return news_letters
```

Browser

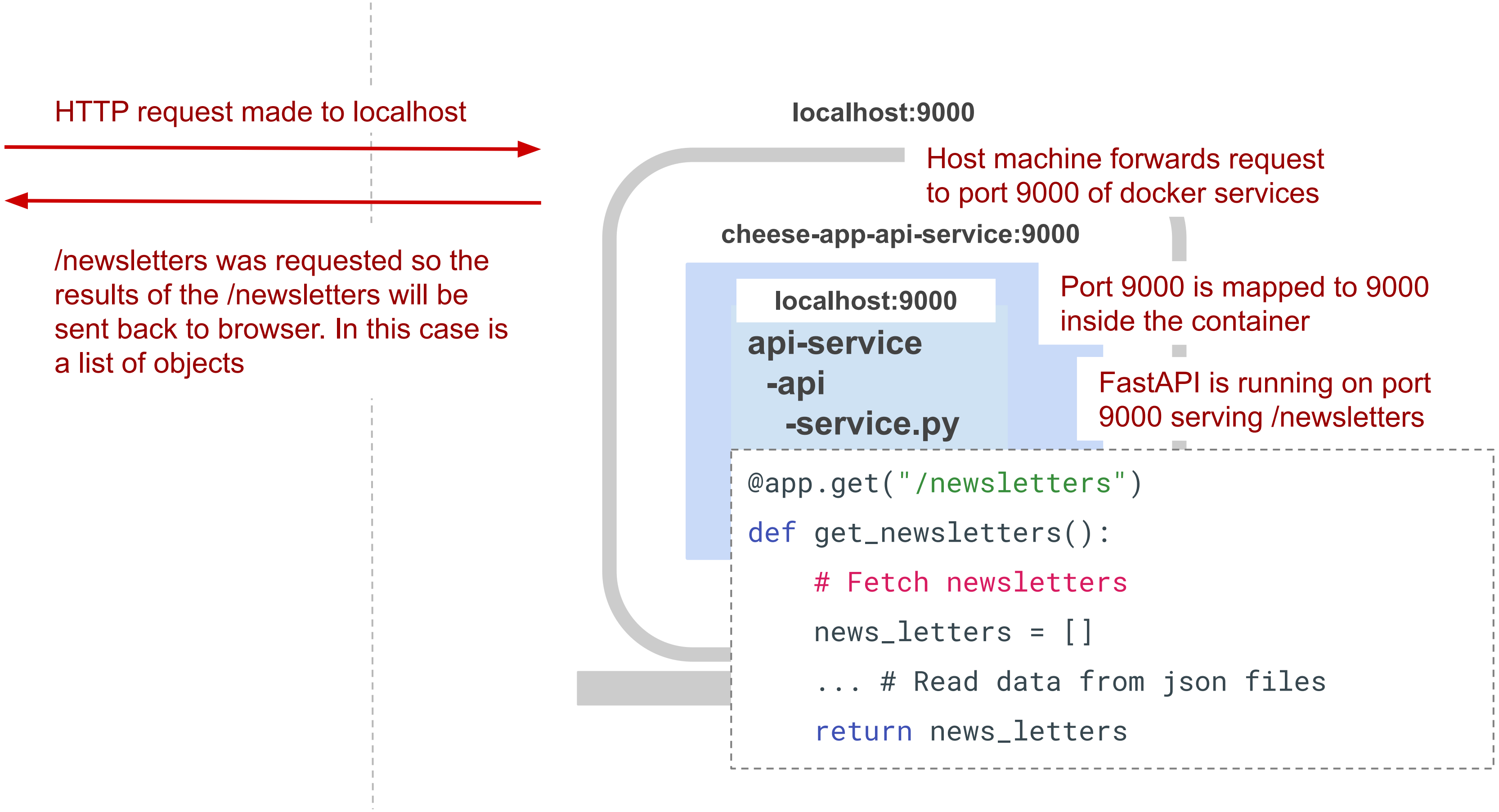
Local computer / Server

How does an API work



http://localhost:9000/newsletters

```
[
  {
    "id": "1",
    "dts": 1730476705,
    "title": "Spanish Cheese Journey",
    "excerpt": "Take a virtual tour through...",
    "detail": "...",
    "readTime": "6 min read",
    "category": "Regional Spotlight",
    "image": "spanish-cheese.jpg"
  },
  {
    "id": "2",
    "dts": 1729871905,
    "title": "The Art of Blue Cheese",
    "excerpt": "Dive into the fascinating...",
    "detail": "...",
    "readTime": "4 min read",
    "category": "Cheese Science",
    "image": "blue-cheese.jpg"
  }
]
```



HTTP request made to localhost

/newsletters was requested so the results of the /newsletters will be sent back to browser. In this case is a list of objects

Host machine forwards request to port 9000 of docker services

Port 9000 is mapped to 9000 inside the container

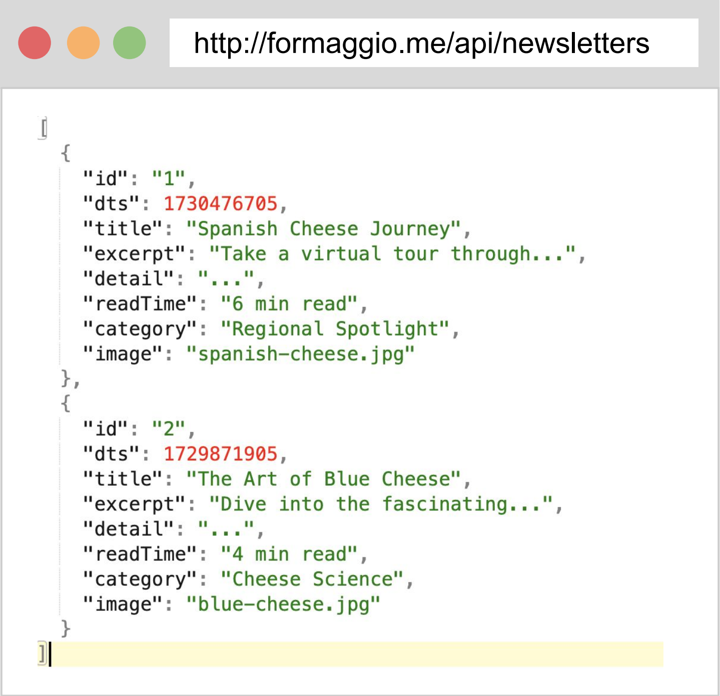
FastAPI is running on port 9000 serving /newsletters

```
@app.get("/newsletters")
def get_newsletters():
    # Fetch newsletters
    news_letters = []
    ... # Read data from json files
    return news_letters
```

Browser

Local computer / Server

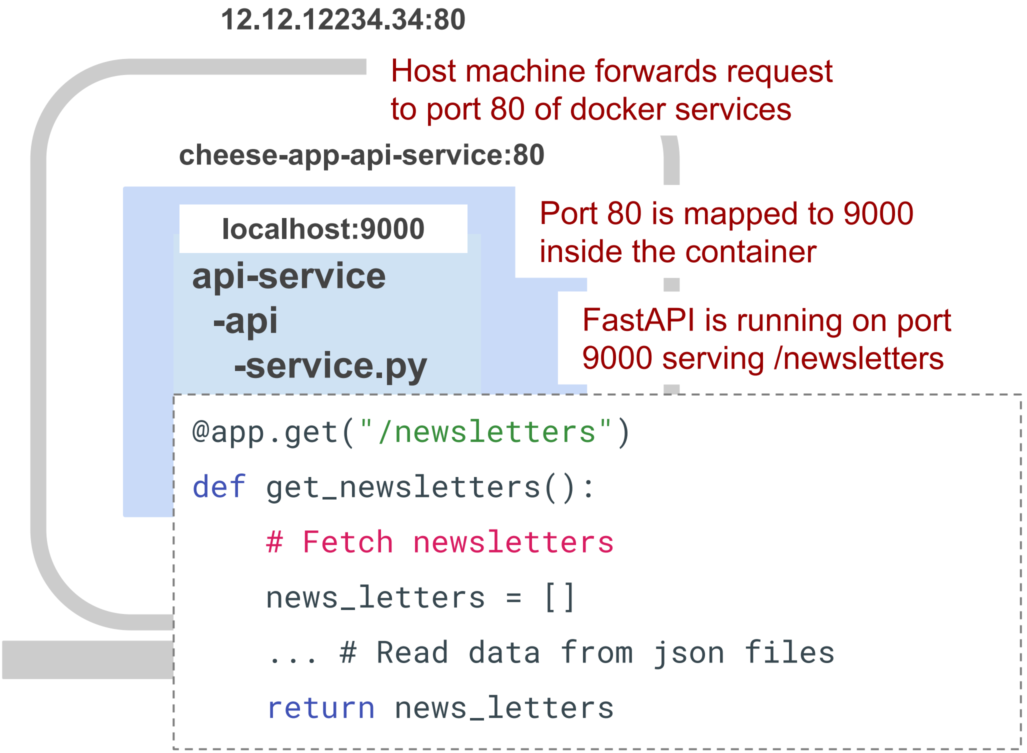
How does an API work (In Production)



```
[
  {
    "id": "1",
    "dts": 1730476705,
    "title": "Spanish Cheese Journey",
    "excerpt": "Take a virtual tour through...",
    "detail": "...",
    "readTime": "6 min read",
    "category": "Regional Spotlight",
    "image": "spanish-cheese.jpg"
  },
  {
    "id": "2",
    "dts": 1729871905,
    "title": "The Art of Blue Cheese",
    "excerpt": "Dive into the fascinating...",
    "detail": "...",
    "readTime": "4 min read",
    "category": "Cheese Science",
    "image": "blue-cheese.jpg"
  }
]
```

HTTP request made to server

/newsletters was requested so the results of the /newsletters will be sent back to browser. In this case is a list of objects



Host machine forwards request to port 80 of docker services

Port 80 is mapped to 9000 inside the container

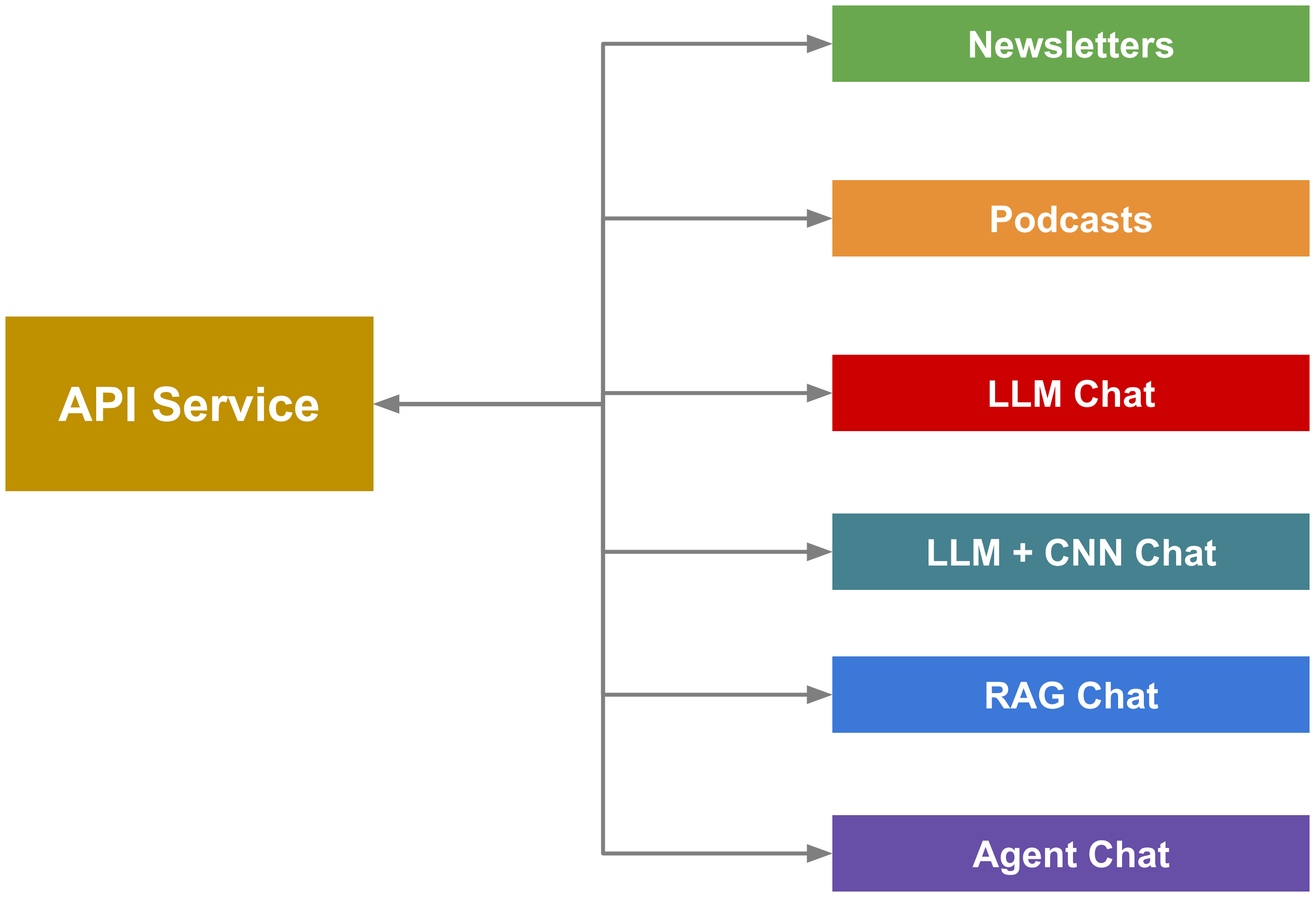
FastAPI is running on port 9000 serving /newsletters

```
@app.get("/newsletters")
def get_newsletters():
    # Fetch newsletters
    news_letters = []
    ... # Read data from json files
    return news_letters
```

Browser

GCP Server

Tutorial: APIs



Tutorial: APIs

Steps to build Cheese App **APIs**:

- Ensure vector database is running.
- Expose data using an API.
- For detailed instructions, please refer to the following link
  - Cheese App APIs. (https://github.com/dlops-io/cheese-app-v2#setup-environments)

Outline

1. Recap
2. APIs
3. **App Frontend (Simple)**
4. Frontend Frameworks
5. Frontend App (React)

App Frontend

HTML

- Is Hyper Text Markup Language (Remember Markdowns)
- Browsers use HTML to display web pages

CSS

- Cascading style sheets
- Used to format & style web pages

Javascript

- Programming language understood by browser

App Frontend

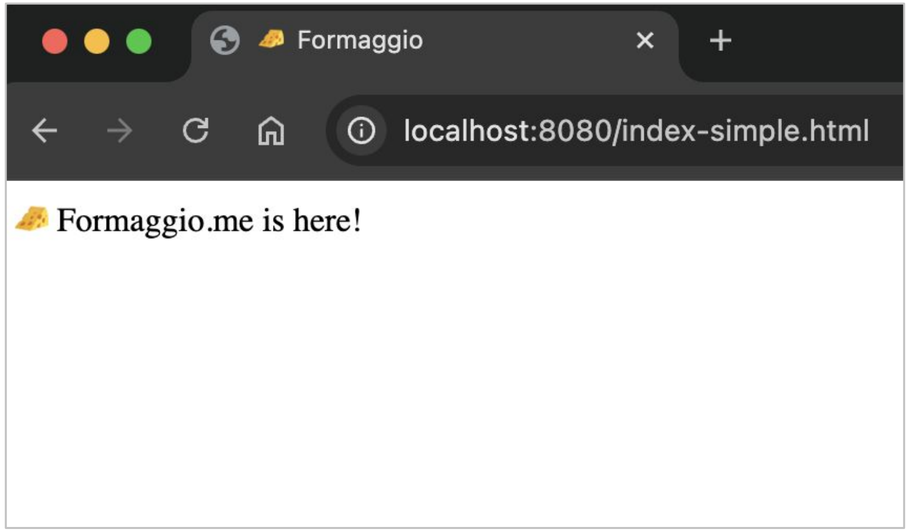
```html
<!DOCTYPE html>
<html>
<head>
    <title>🧀 Formaggio</title>
    <style>body{background-color: #efefef;}</style>
</head>
<body>
    🧀 Formaggio.me is here!
</body>
<script>
    var input_file =
document.getElementById("input_file");
</script>
</html>
```

Browser Title

Page Style

Web page details

Web page scripts (Javascript)

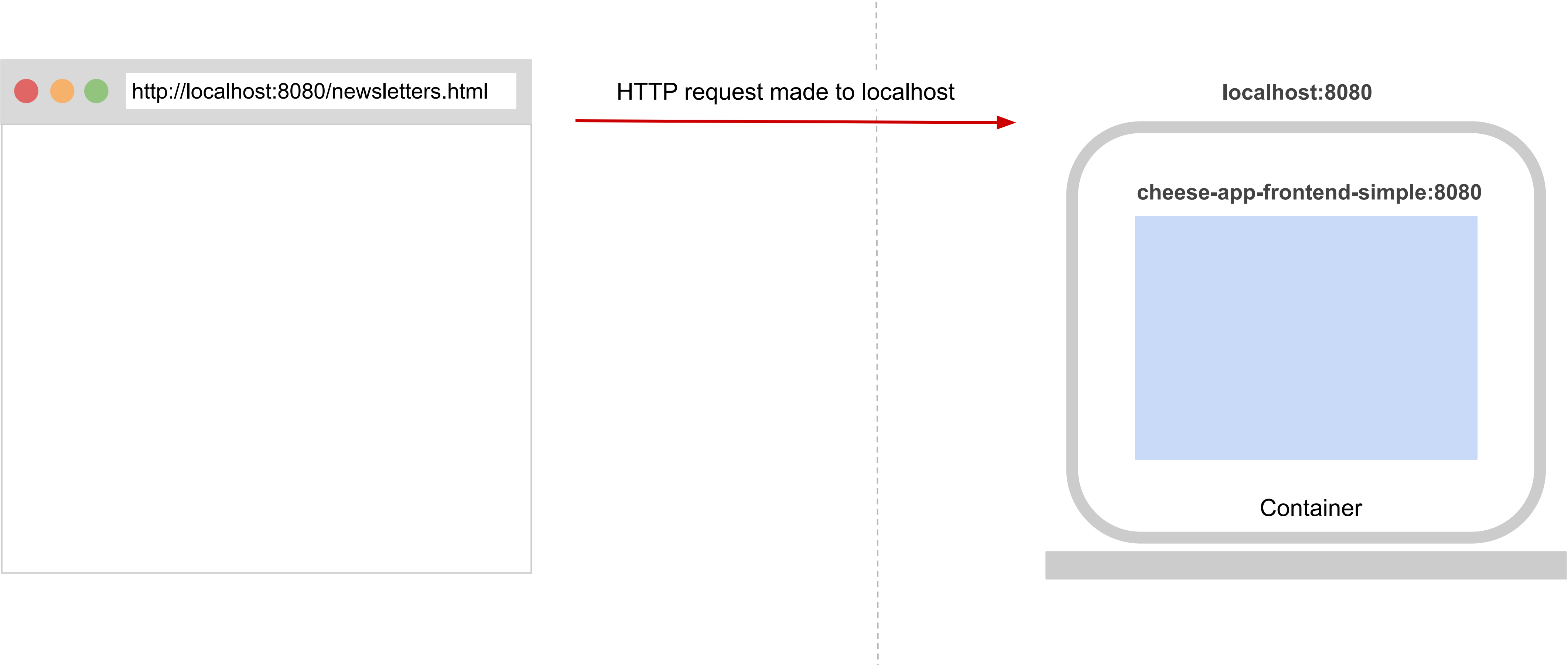


🧀 Formaggio

localhost:8080/index-simple.html

🧀 Formaggio.me is here!

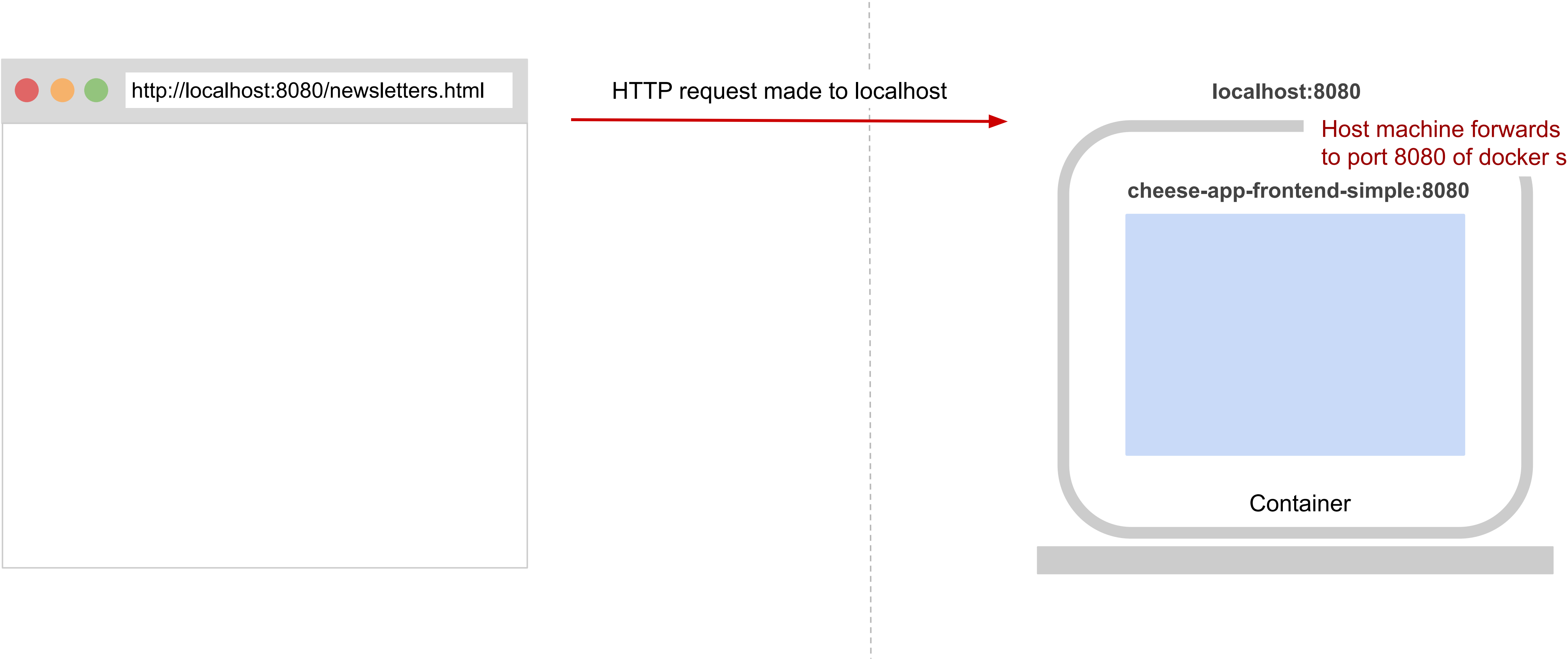
How does the App work


http://localhost:8080/newsletters.html
HTTP request made to localhost
localhost:8080
cheese-app-frontend-simple:8080
Container

Browser

Local computer / Server

How does the App work



http://localhost:8080/newsletters.html

HTTP request made to localhost

localhost:8080

Host machine forwards request to port 8080 of docker services

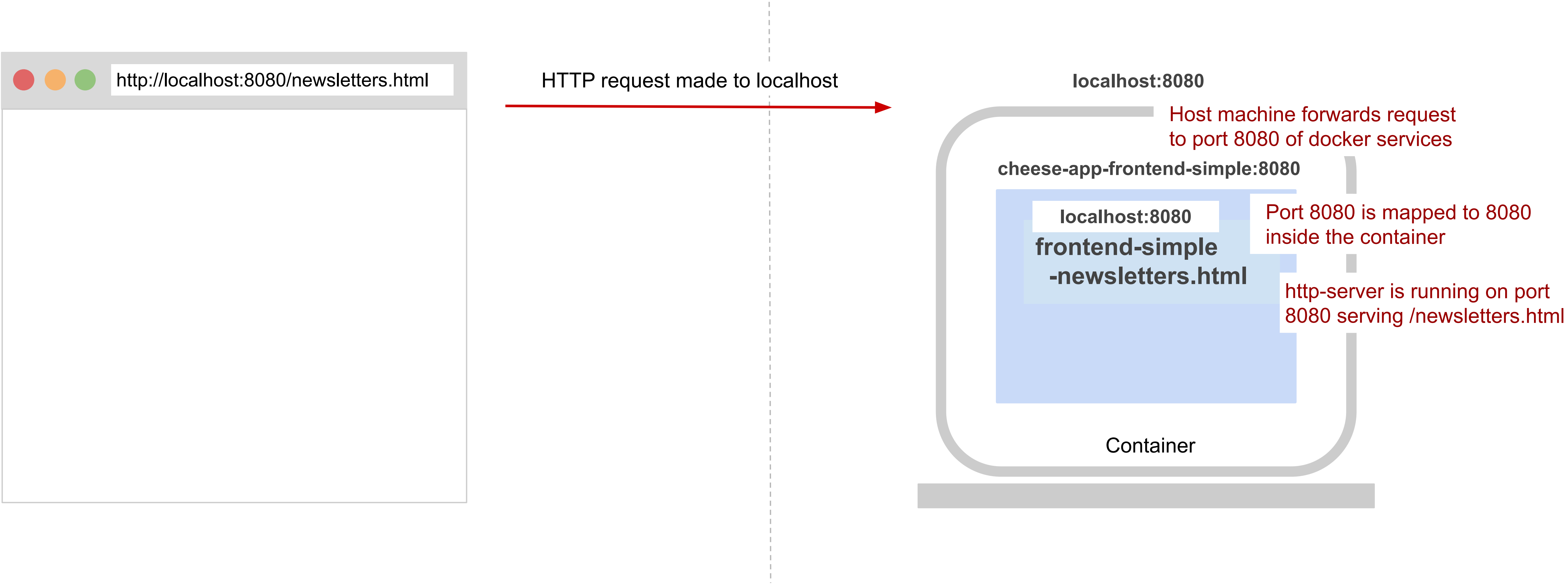
cheese-app-frontend-simple:8080

Container

Browser

Local computer / Server

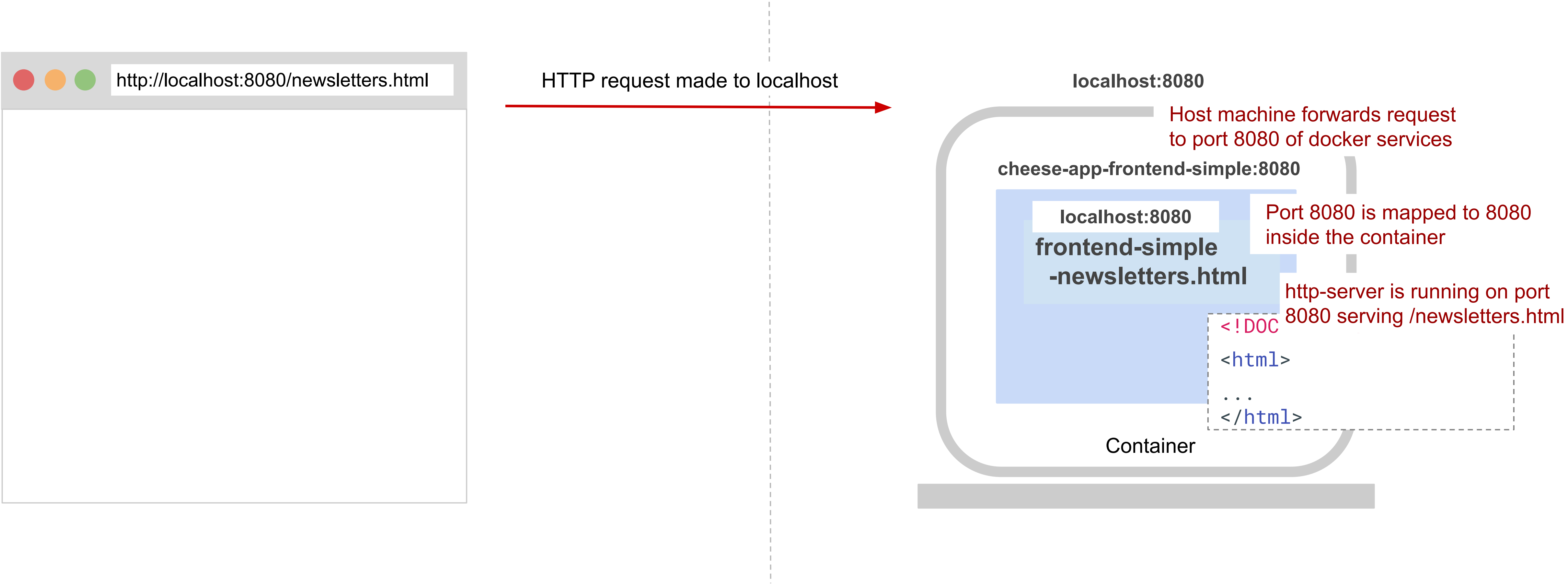
How does the App work


http://localhost:8080/newsletters.html
HTTP request made to localhost
localhost:8080
Host machine forwards request to port 8080 of docker services
cheese-app-frontend-simple:8080
localhost:8080
frontend-simple
-newsletters.html
Port 8080 is mapped to 8080 inside the container
http-server is running on port 8080 serving /newsletters.html
Container

Browser

Local computer / Server

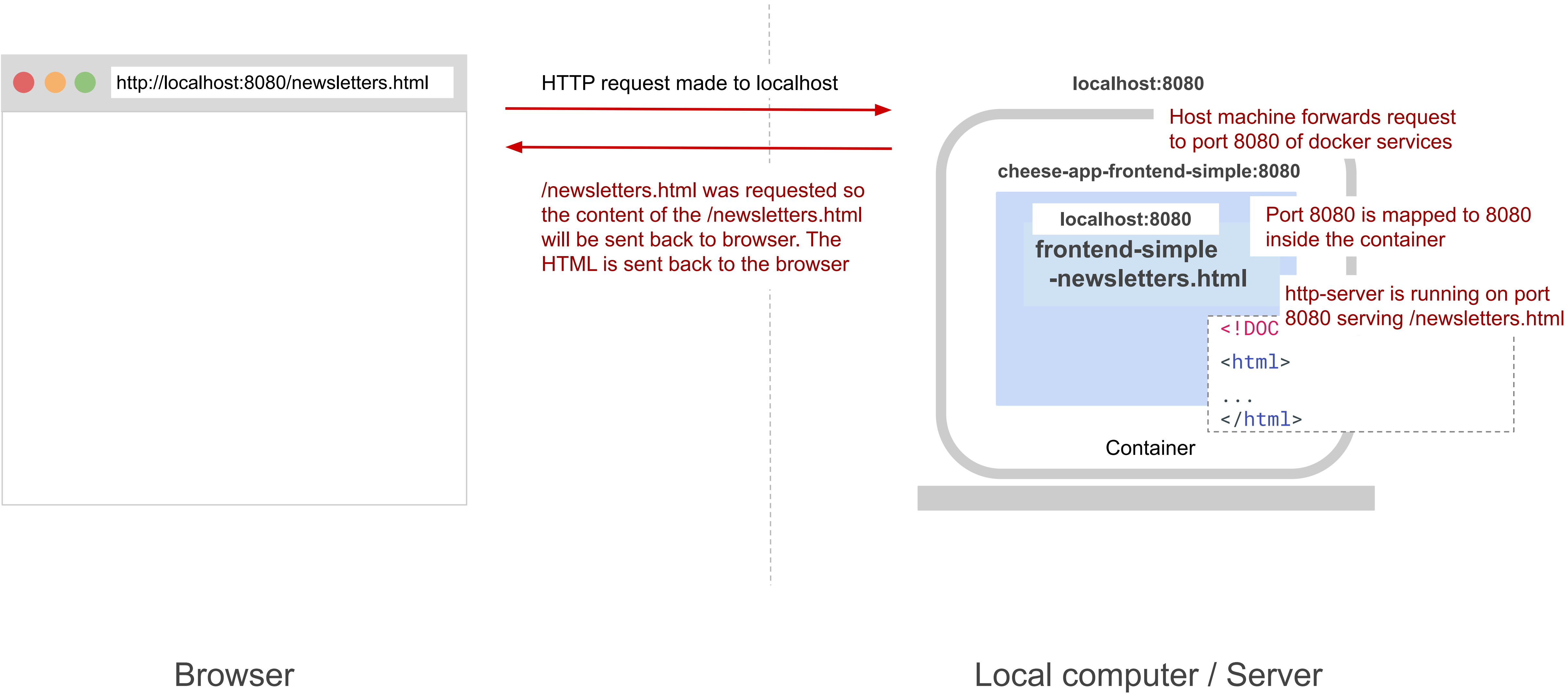
How does the App work


http://localhost:8080/newsletters.html
HTTP request made to localhost
localhost:8080
Host machine forwards request to port 8080 of docker services
cheese-app-frontend-simple:8080
localhost:8080
frontend-simple
-newsletters.html
Port 8080 is mapped to 8080 inside the container
http-server is running on port 8080 serving /newsletters.html
<!DOC
<html>
...
</html>
Container

Browser

Local computer / Server

How does the App work



http://localhost:8080/newsletters.html

HTTP request made to localhost

/newsletters.html was requested so the content of the /newsletters.html will be sent back to browser. The HTML is sent back to the browser

localhost:8080

Host machine forwards request to port 8080 of docker services

cheese-app-frontend-simple:8080

localhost:8080

frontend-simple -newsletters.html

Port 8080 is mapped to 8080 inside the container

http-server is running on port 8080 serving /newsletters.html

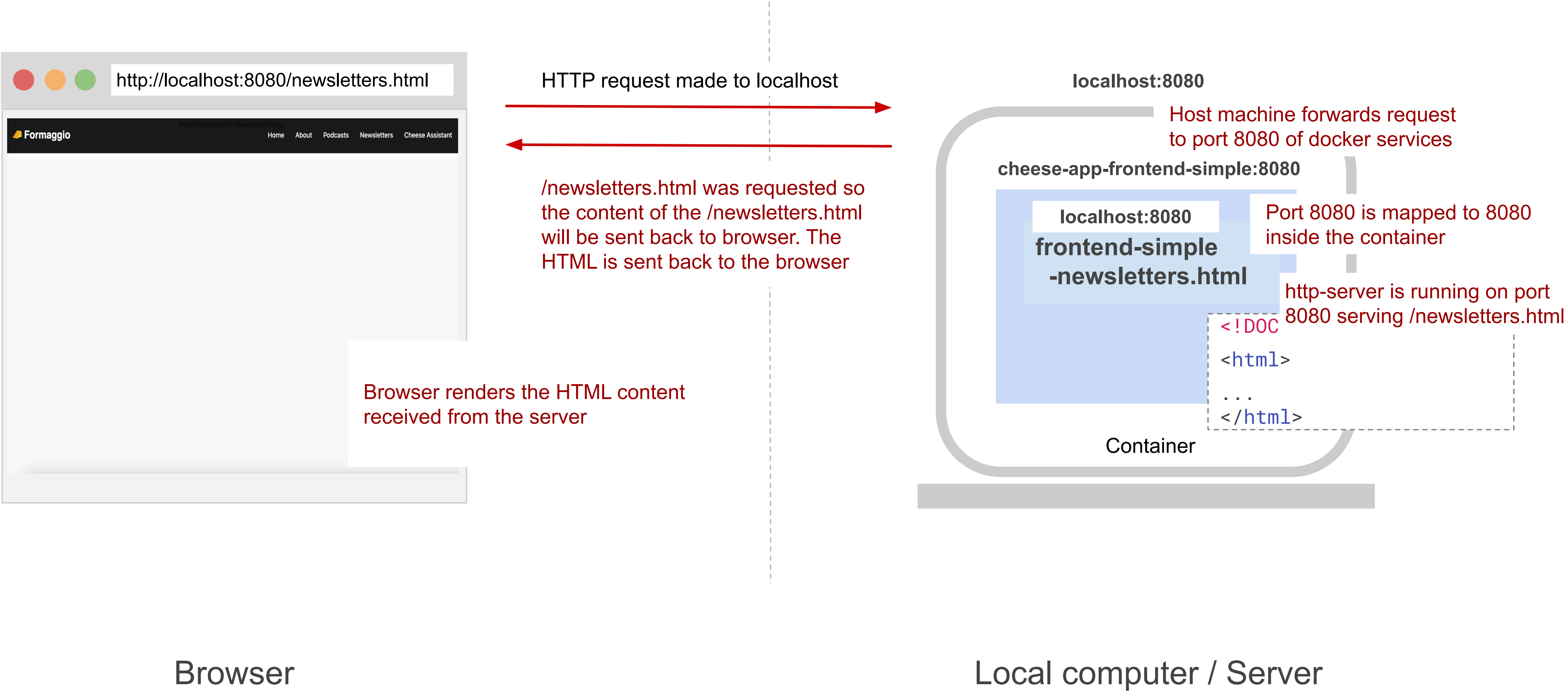
```
<!DOC
<html>
...
</html>
```

Container

Browser

Local computer / Server

How does the App work



HTTP request made to localhost

/newsletters.html was requested so the content of the /newsletters.html will be sent back to browser. The HTML is sent back to the browser

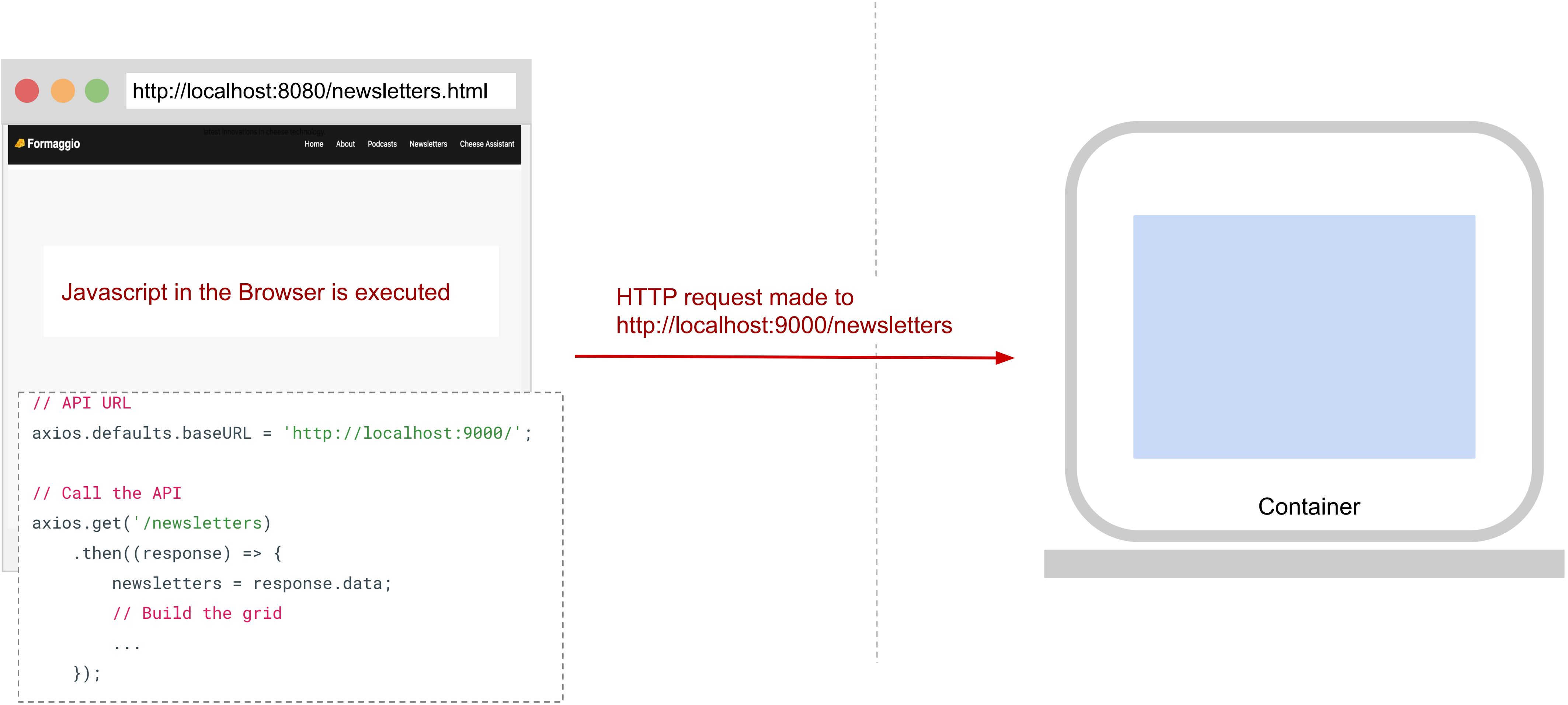
Host machine forwards request to port 8080 of docker services

Port 8080 is mapped to 8080 inside the container

http-server is running on port 8080 serving /newsletters.html

Browser renders the HTML content received from the server

How does the App work


http://localhost:8080/newsletters.html
latest innovations in cheese technology.
Formaggio
Home
About
Podcasts
Newsletters
Cheese Assistant

Javascript in the Browser is executed

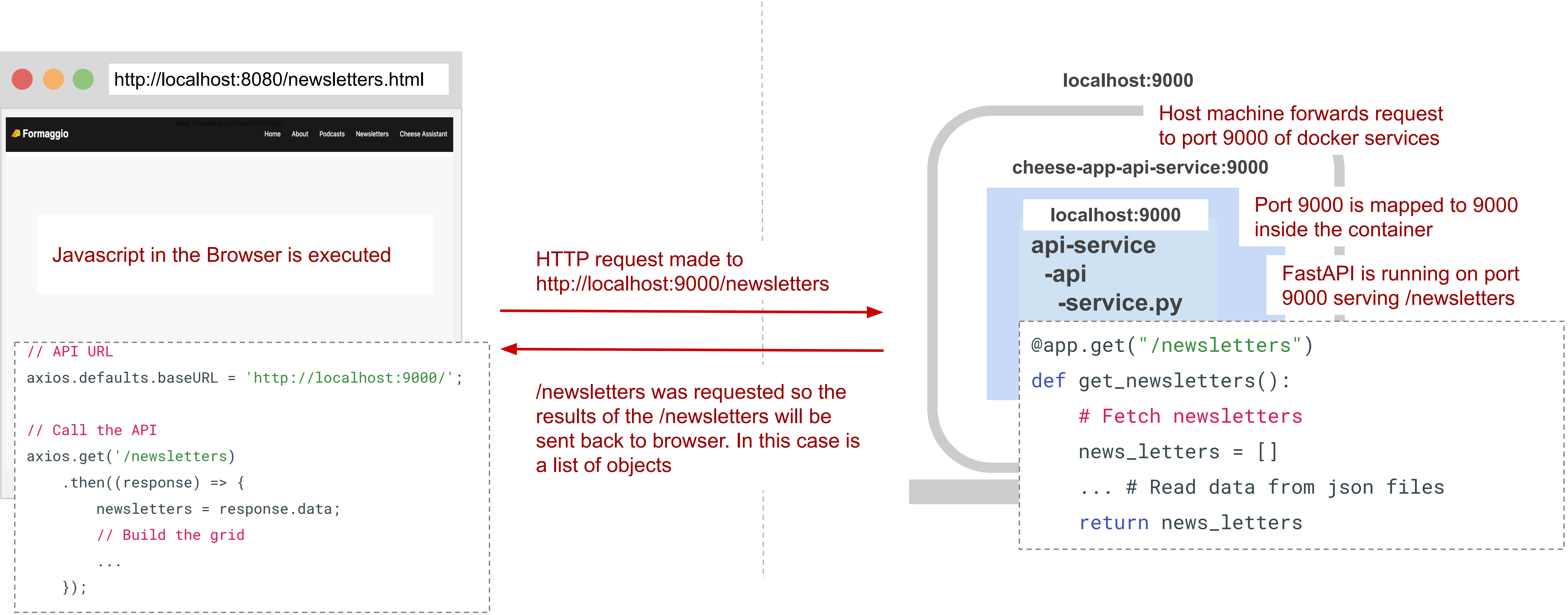
```
// API URL
axios.defaults.baseURL = 'http://localhost:9000/';

// Call the API
axios.get('/newsletters)
    .then((response) => {
        newsletters = response.data;
        // Build the grid
        ...
    });
```

HTTP request made to http://localhost:9000/newsletters

Browser

Local computer / Server

How does the App work



Javascript in the Browser is executed

```
// API URL
axios.defaults.baseURL = 'http://localhost:9000/';

// Call the API
axios.get('/newsletters)
    .then((response) => {
        newsletters = response.data;
        // Build the grid
        ...
    });
```

HTTP request made to http://localhost:9000/newsletters

/newsletters was requested so the results of the /newsletters will be sent back to browser. In this case is a list of objects

Host machine forwards request to port 9000 of docker services

Port 9000 is mapped to 9000 inside the container

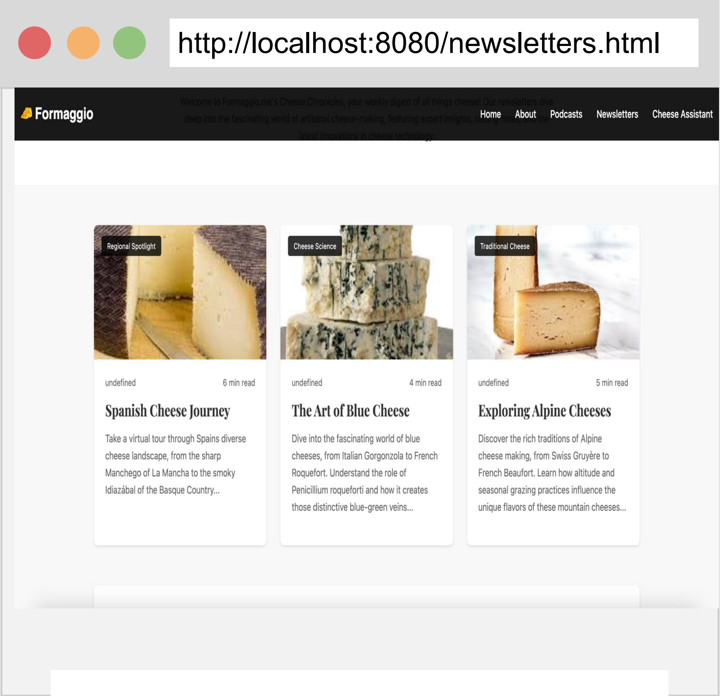
FastAPI is running on port 9000 serving /newsletters

```
@app.get("/newsletters")
def get_newsletters():
    # Fetch newsletters
    news_letters = []
    ... # Read data from json files
    return news_letters
```

Browser

Local computer / Server

How does the App work



http://localhost:8080/newsletters.html

Welcome to Formaggio.me's Cheese Chronicles, your weekly digest of all things cheese! Our newsletters dive deep into the fascinating world of artisanal cheese-making, featuring expert insights, tasting notes, and the latest innovations in cheese technology.

Regional Spotlight

undefined 6 min read

Spanish Cheese Journey

Take a virtual tour through Spains diverse cheese landscape, from the sharp Manchego of La Mancha to the smoky Idiazábal of the Basque Country...

Cheese Science

undefined 4 min read

The Art of Blue Cheese

Dive into the fascinating world of blue cheeses, from Italian Gorgonzola to French Roquefort. Understand the role of Penicillium roqueforti and how it creates those distinctive blue-green veins...

Traditional Cheese

undefined 5 min read

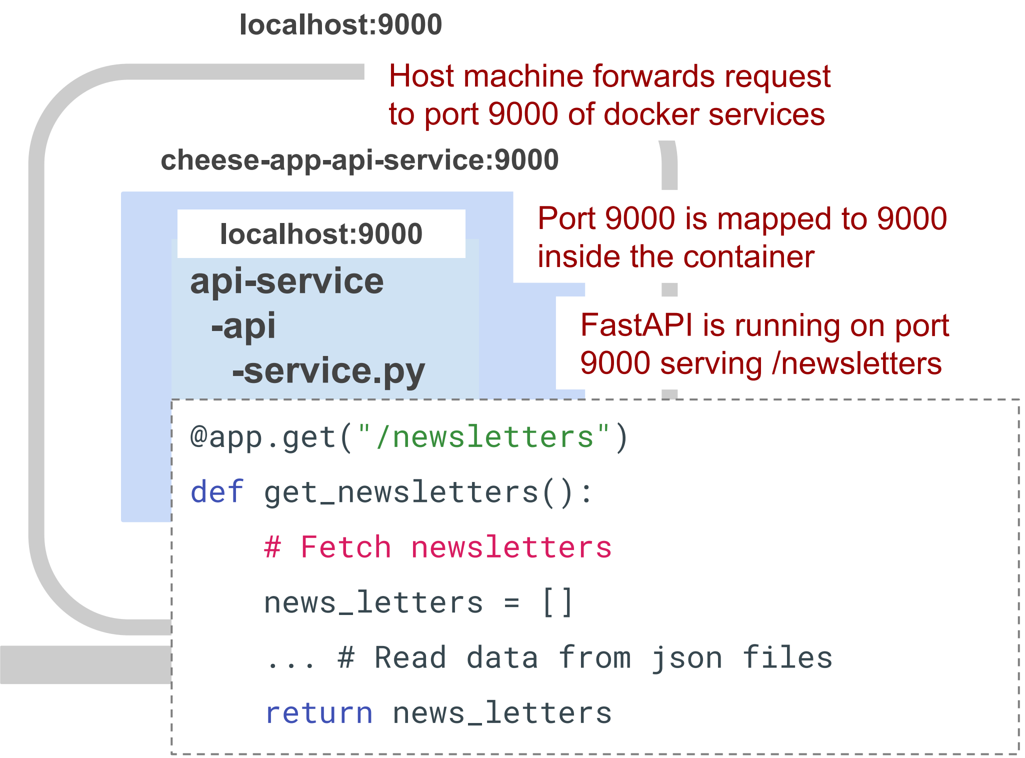
Exploring Alpine Cheeses

Discover the rich traditions of Alpine cheese making, from Swiss Gruyère to French Beaufort. Learn how altitude and seasonal grazing practices influence the unique flavors of these mountain cheeses...

Javascript displays the newsletters data in the html page.

HTTP request made to http://localhost:9000/newsletters

/newsletters was requested so the results of the /newsletters will be sent back to browser. In this case is a list of objects



Host machine forwards request to port 9000 of docker services

Port 9000 is mapped to 9000 inside the container

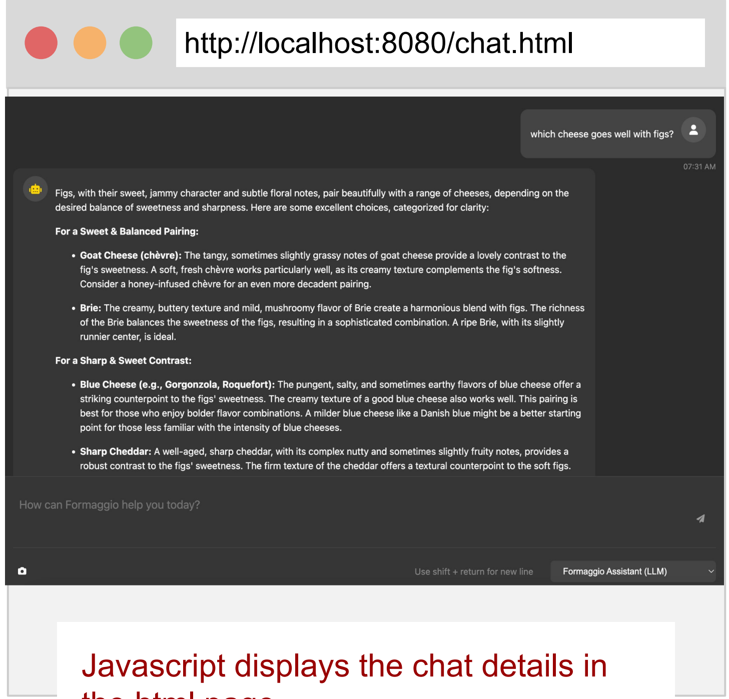
FastAPI is running on port 9000 serving /newsletters

```
@app.get("/newsletters")
def get_newsletters():
    # Fetch newsletters
    news_letters = []
    ... # Read data from json files
    return news_letters
```

Browser

Local computer / Server

How does the App work (Chat)



which cheese goes well with figs?

07:31 AM

Figs, with their sweet, jammy character and subtle floral notes, pair beautifully with a range of cheeses, depending on the desired balance of sweetness and sharpness. Here are some excellent choices, categorized for clarity:

For a Sweet & Balanced Pairing:

- **Goat Cheese (chèvre):** The tangy, sometimes slightly grassy notes of goat cheese provide a lovely contrast to the fig's sweetness. A soft, fresh chèvre works particularly well, as its creamy texture complements the fig's softness. Consider a honey-infused chèvre for an even more decadent pairing.
- **Brie:** The creamy, buttery texture and mild, mushroomy flavor of Brie create a harmonious blend with figs. The richness of the Brie balances the sweetness of the figs, resulting in a sophisticated combination. A ripe Brie, with its slightly runnier center, is ideal.

For a Sharp & Sweet Contrast:

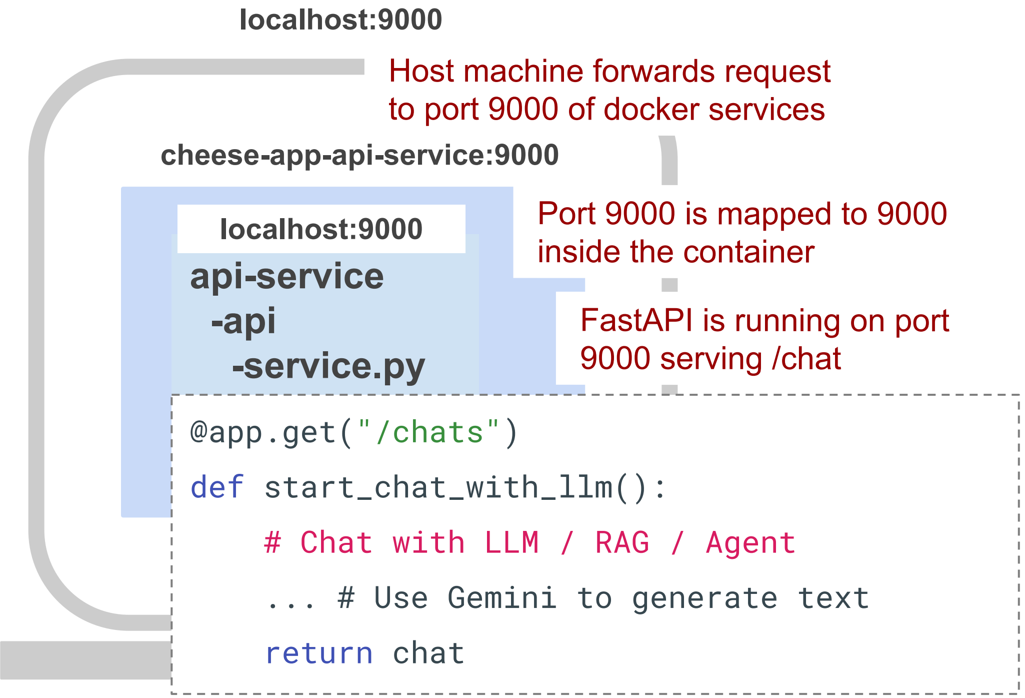
- **Blue Cheese (e.g., Gorgonzola, Roquefort):** The pungent, salty, and sometimes earthy flavors of blue cheese offer a striking counterpoint to the figs' sweetness. The creamy texture of a good blue cheese also works well. This pairing is best for those who enjoy bolder flavor combinations. A milder blue cheese like a Danish blue might be a better starting point for those less familiar with the intensity of blue cheeses.
- **Sharp Cheddar:** A well-aged, sharp cheddar, with its complex nutty and sometimes slightly fruity notes, provides a robust contrast to the figs' sweetness. The firm texture of the cheddar offers a textural counterpoint to the soft figs.

How can Formaggio help you today?
Use shift + return for new line
Formaggio Assistant (LLM)

Javascript displays the chat details in the html page.

HTTP request made to http://localhost:9000/chat

/chat was requested so the results of the /chat will be sent back to browser. In this case it is the chat details



Host machine forwards request to port 9000 of docker services

Port 9000 is mapped to 9000 inside the container

FastAPI is running on port 9000 serving /chat

```
@app.get("/chats")
def start_chat_with_llm():
    # Chat with LLM / RAG / Agent
    ... # Use Gemini to generate text
    return chat
```

Browser

Local computer / Server

Tutorial: Frontend Simple

Steps to run Cheese App **Frontend**:

- https://github.com/dlops-io/cheese-app-v2#frontend-app-simple

Outline

1. Recap
2. APIs
3. App Frontend (Simple)
4. **Frontend Frameworks**
5. Frontend App (React)

Frontend

When we build our frontend we need a page for each component:

- index.html
- newsletters.html
- podcasts.html
- chat.html

Frontend

When we build our frontend we need a page for each component:

- index.html
- newsletters.html
- podcasts.html
- chat.html

Problems:

- Each of these had its own HTML, Javascript, CSS
- How do we share/reuse code across pages?
- Each page is loaded separately in browser (Slow)

Frontend

Problems:

- Each of these had its own HTML, Javascript, CSS
- How do we share/reuse code across pages
- Each page is loaded separately in browser (Slow)

Solution:

- Create a single page app that manages HTML, Javascript, CSS as components
- Use frontend App **Frameworks**

Frontend Frameworks

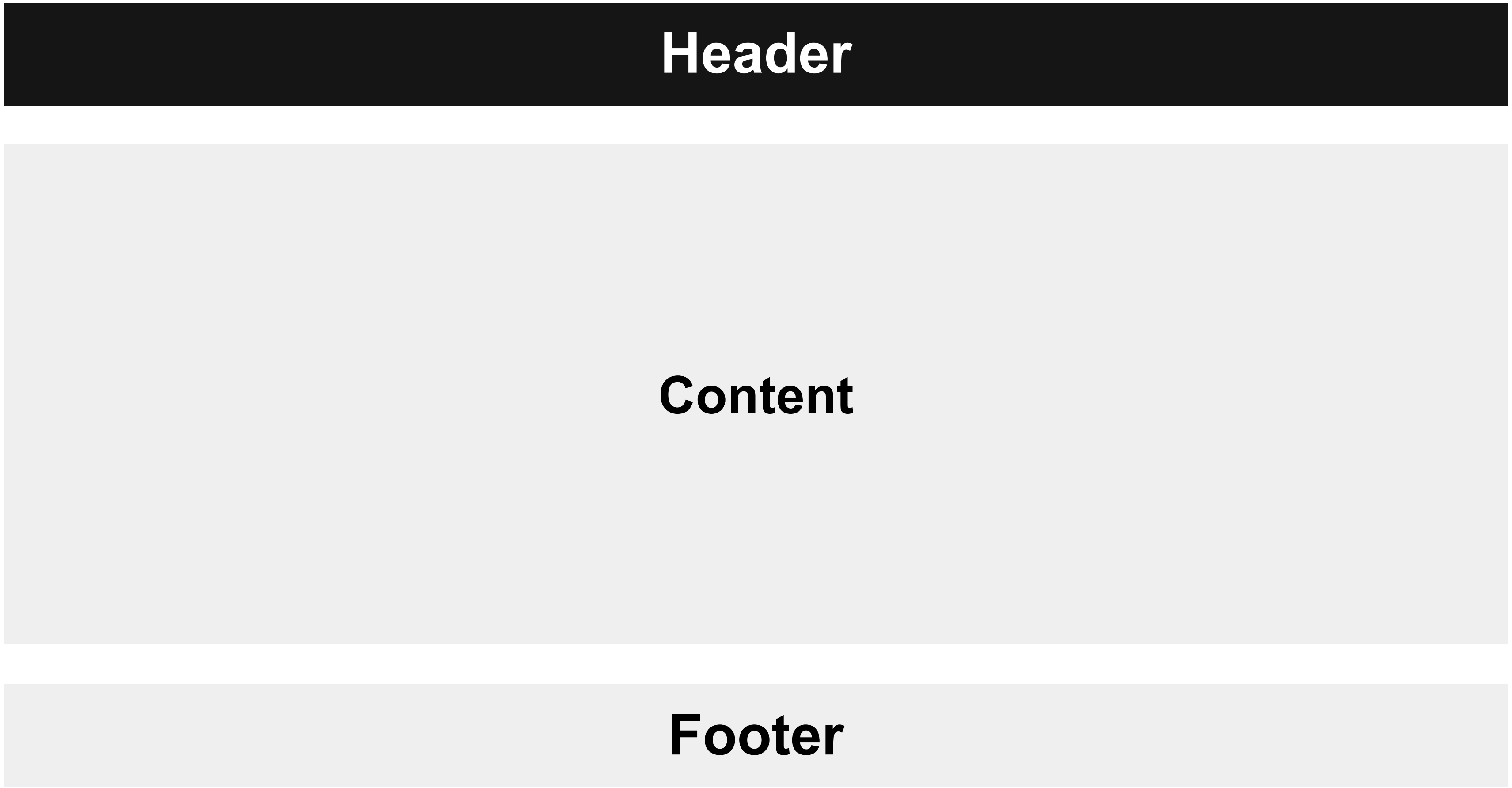
The common frontend app frameworks are:

- Angular (Google)
- **React (Facebook)**
- Vue
- Svelte

React

- Everything is a **Component**
- Uses **JSX** instead of Javascript
- JSX is an extension to JavaScript
- JSX is like a template language, but it comes with the full power of JavaScript

React App



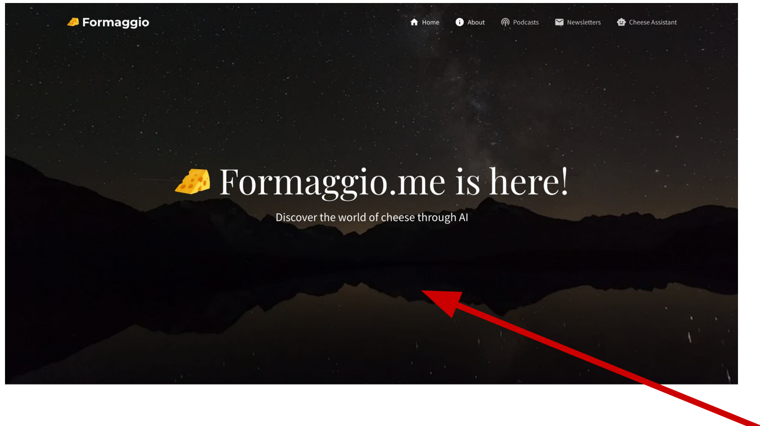
Header

Content

Footer

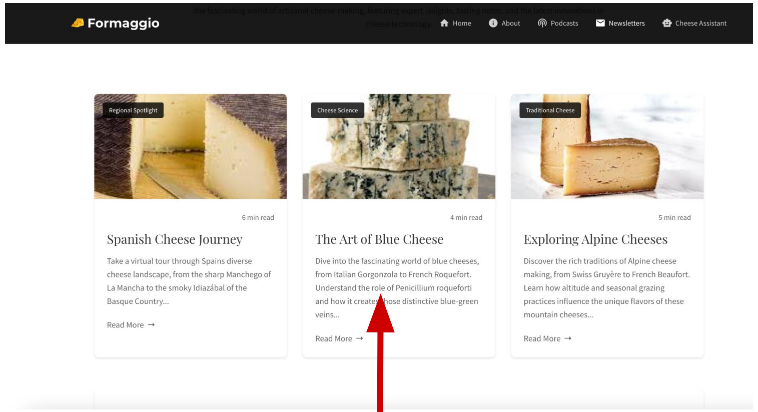
React App

Header defined only once



Formaggio.me is here!

Discover the world of cheese through AI



6 min read

Spanish Cheese Journey

Take a virtual tour through Spains diverse cheese landscape, from the sharp Manchego of La Mancha to the smoky Idiazábal of the Basque Country...

Read More →

4 min read

The Art of Blue Cheese

Dive into the fascinating world of blue cheeses, from Italian Gorgonzola to French Roquefort. Understand the role of Penicillium roqueforti and how it creates those distinctive blue-green veins...

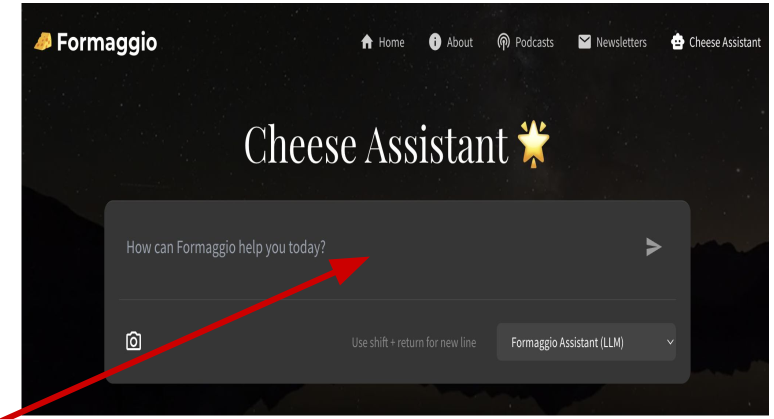
Read More →

5 min read

Exploring Alpine Cheeses

Discover the rich traditions of Alpine cheese making, from Swiss Gruyère to French Beaufort. Learn how altitude and seasonal grazing practices influence the unique flavors of these mountain cheeses...

Read More →



Cheese Assistant

How can Formaggio help you today?

Use shift + return for new line

Formaggio Assistant (LLM)

Content block switched for each page

Tutorial: Frontend React

Steps to run Cheese App **React Frontend**:

- https://github.com/dlops-io/cheese-app-v2#frontend-app-react

THANK YOU