goto;

MAXIMILIANO FIRTMAN
@firt firt.dev



VANILLA WEB

You don't need that library

About me

Maximiliano Firtman



MOBILE+WEB DEVELOPER

HTML since 1996

JavaScript since 1998, +150 web apps

AUTHOR

Authored 14 books, +70 courses



Learn PWA

A course that breaks down every aspect of modern progressive web app development.

25 activities

0% completed

Save your progress
Sign in

1 Welcome to Learn Progressive Web Apps!

Article

A course that breaks down every aspect of modern progressive web app development.

Read article

2 Progressive Web Apps

Article

3 Getting started

Article

4 Foundations

Article

5 App design

Article

6 Assets and data

Article

Frontend Masters

PLURALSIGHT

LinkedIn Learning

O'REILLY®

Some questions





[Click here for a Blast from TV's past!]



Our communities are home to the most popular collection of FREE HOME PAGES & E-MAIL on the web. Please join or visit one of our 29 neighborhoods today.

* Free Home Pages & Free Member Email Advertiser Information

Audio DIALweb Update

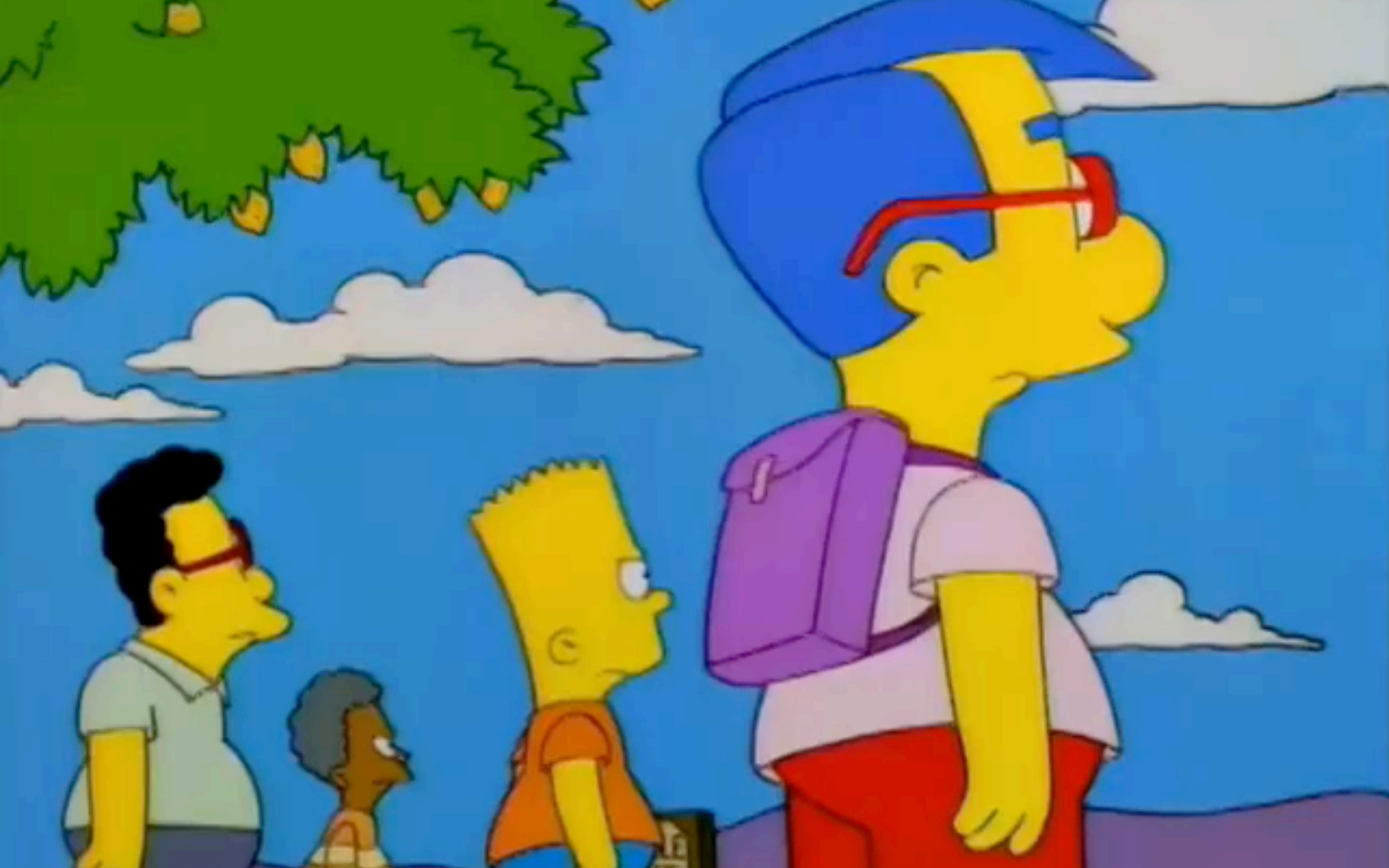
GeoCities Daily Audio Update -- ***Sponsored by IBM VoiceType Simply Speaking***

HOT GEOSTORE

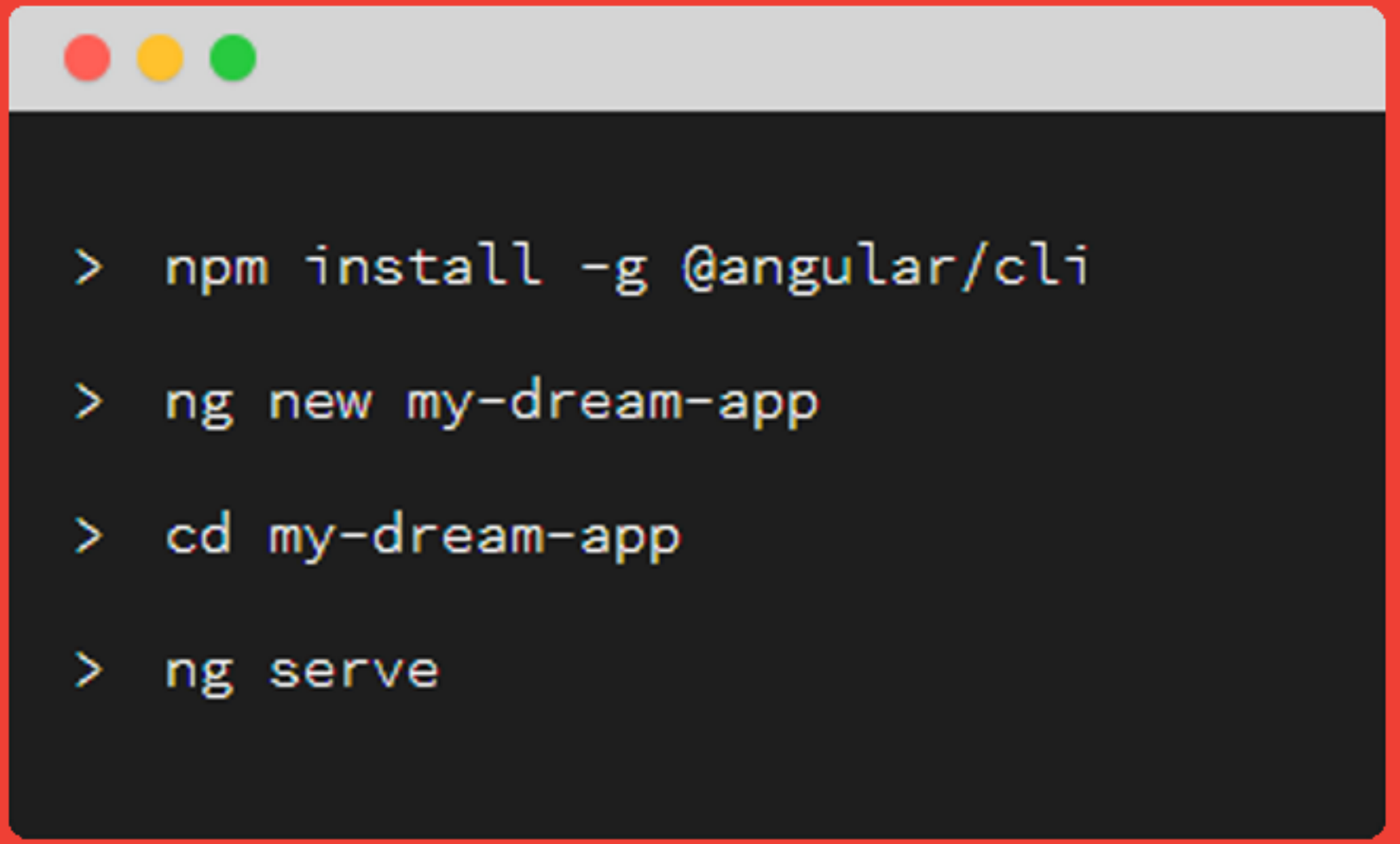
Today's Cool Homestead
Yosemite4273
Sunsets, coastal seagulls and flowers are part of the photographic fare at inetd's homepage.

Geotoon of the week

GeoCities News of the Day - 12/20/96



```
C:\WEB>_
```


> npm install -g @angular/cli
> ng new my-dream-app
> cd my-dream-app
> ng serve





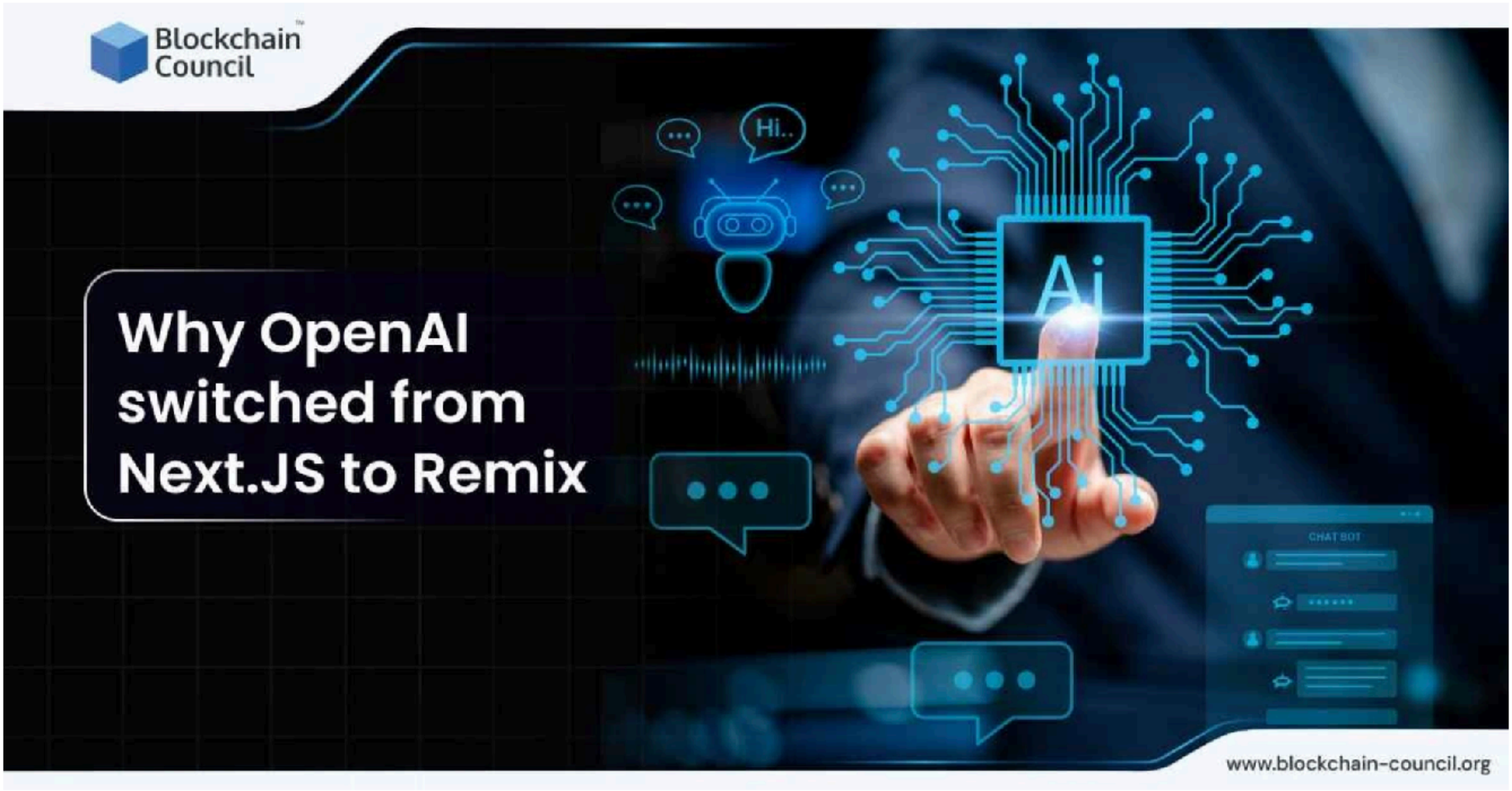




node_modules

How did we get here?

Why Did OpenAI Switch From Next.JS to Remix?



Blockchain Council September 15, 2024



Vanilla Web



But I love

React



But I love

Angular



But I love

Vue







I want you to also love

Vanilla JS





DEFINITION

Vanilla JavaScript

The usage of the core language and browser APIs to create web apps without any additional libraries or frameworks added on top



IMPORTANT

Vanilla JavaScript has been as a concept for a decade now





Ben Holmes
@BHolmesDev

Genuinely think every frontend dev should build their own framework at some point



Ben Holmes @BHolmesDev · 11h

Building @slinkitydotdev taught me more JS and web APIs in a month than years of React ever did. ES modules, SPA routing from scratch, bundling strategies... this is the good stuff 🏆

Why do we need to care about VanillaJS?



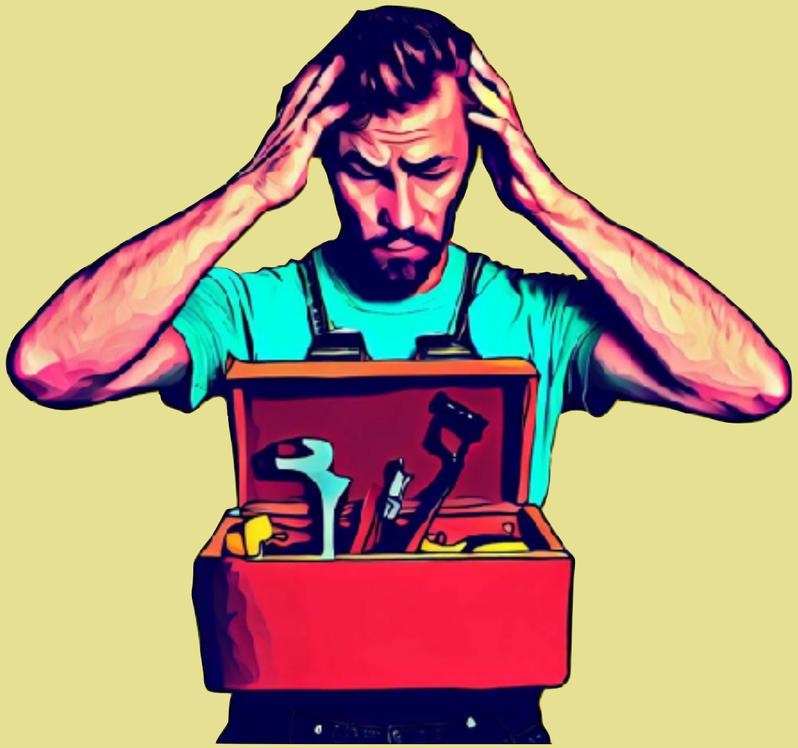
- Add one more tool
- Understand your library
- Extend your library
- Be a better web developer
- To mix it with libraries
- **To use it!**

Main Advantages of Vanilla JS



- Lightweight
- High Performance!
- Control and Power
- Simplicity
- Flexibility
- No node-modules 🤣

Main Fears of Vanilla JS



- Client-side Routing
- Too Verbose
- State Management
- Templating
- Reusable Components
- Learning Curve
- Reinventing the Wheel



IMPORTANT

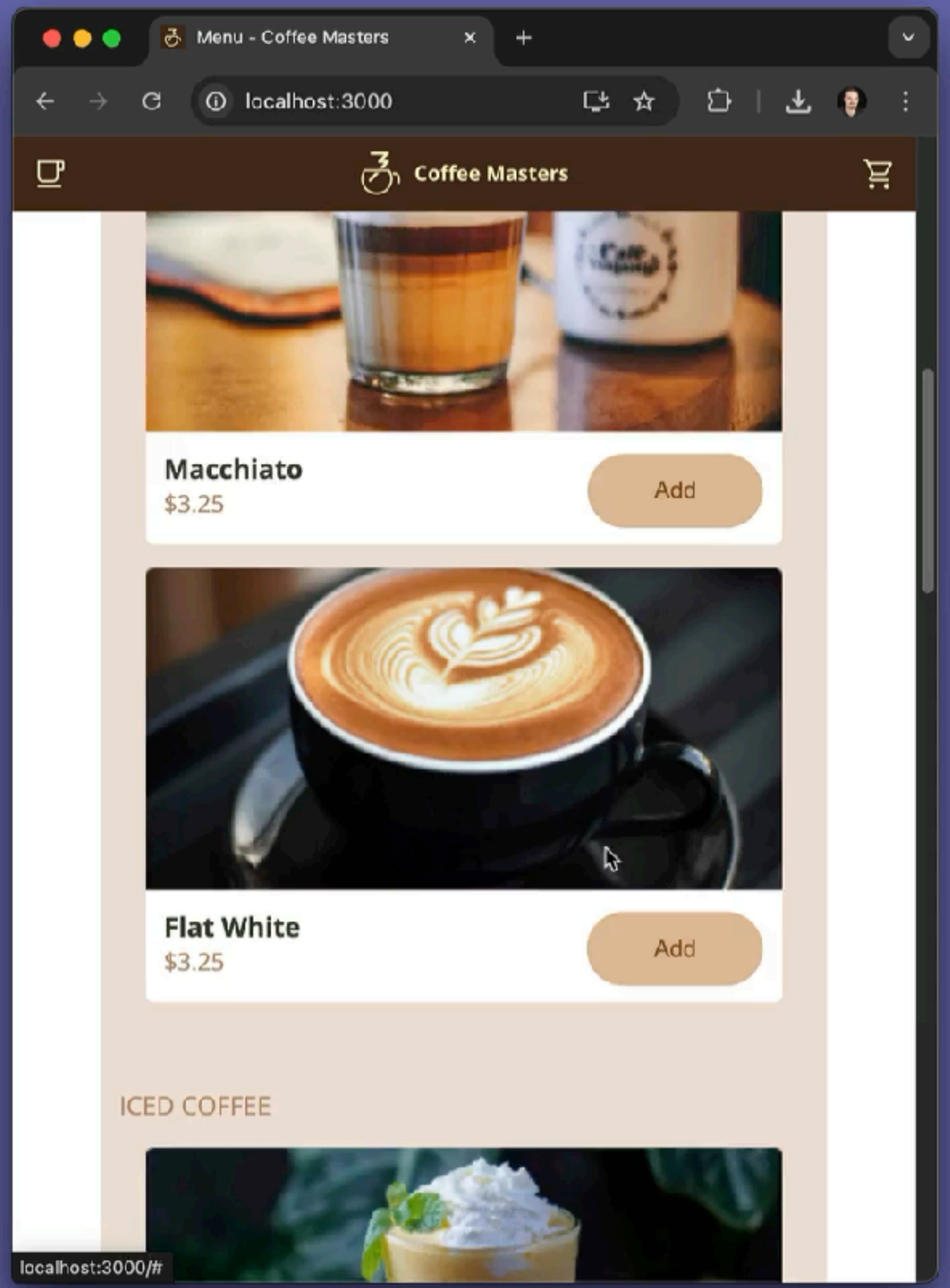
Learn the tool and use it when it's the best option.

We won't advocate for using Vanilla JS on every project.



IDEAS

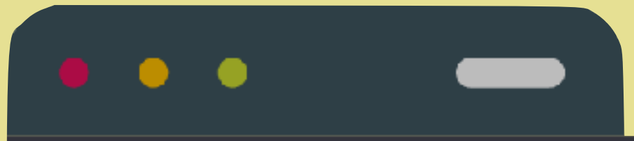
We can still use micro-libraries for specific purposes.


Menu - Coffee Masters
localhost:3000
Coffee Masters
Macchiato
$3.25
Add
Flat White
$3.25
Add
ICED COFFEE
localhost:3000/#



Semantic HTML and Vanilla CSS

Semantic HTML



```
<header>
        <h1><img src="images/logo.svg" width="140" alt="Coffee Masters"></h1>
        <nav>
            <a href="/">cafe</a>
            <span id="badge" hidden></span>
        </nav>
</header>

<main>
        <section>
            <ul id="menu"></ul>
        </section>
</main>
```

Vanilla CSS

```
:root {
    --background-surface: #EFEFEF;
    --primaryColor: #43281C;
    --secondaryColor: #333D29;
    --color3: #7F4F24;
    --color4: #B08968;
    --color5: #DDB892;
    --color6: #EDE0D4;
    --highlight: #FBF2C6;
}
```

Vanilla CSS



```
@container (min-width: 700px) {
  header {
      h2 {
        font-size: 2em;
        color: var(--primaryColor);
      }
  }
}
```



Static Generators

Maximiliano Firtman's content for Mobile and Web Developers

#PWA #Web Performance #Android #JS #iOS – iPadOS #AI #Travel #Native #Retro



PWAs – You Might Not Need That App Store

5-hour online video workshop for web professionals

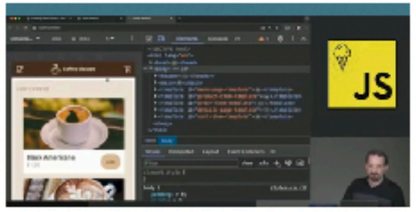
READ MORE →



PHP Basics

4.5-hour online video course for web professionals

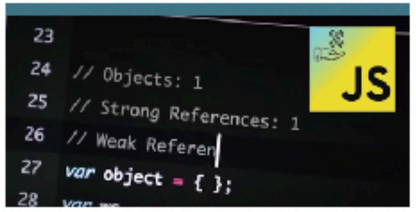
READ MORE →



JavaScript Design Patterns for Web Apps

4.5-hour online video course for web professionals on JavaScript

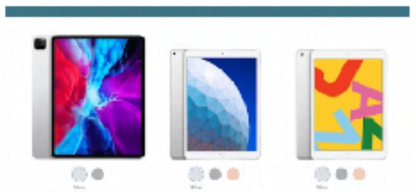
READ MORE →



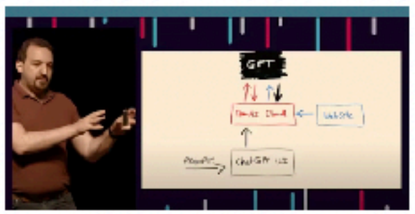
Professional JS – Features You Need to Know

4.5-hour online video course for web professionals on JavaScript

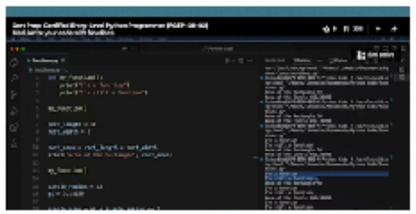
READ MORE →



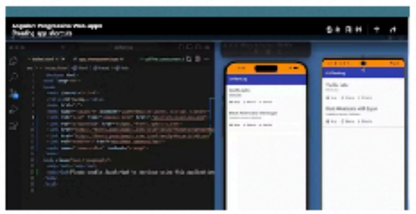
iPad Naming



ChatGPT and AI for

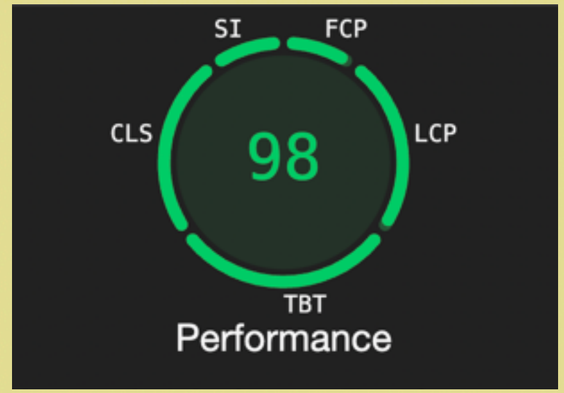


Certified Entry-Level



Angular Progressive

Maximiliano Firtman's content for Mobile and Web Developers



#PWA #Web Performance #Android #JS #iOS – iPadOS #AI #Travel #Native #Retro



PWAs – You Might Not Need That App Store

5-hour online video workshop for web professionals

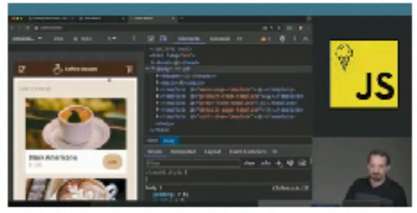
READ MORE →



PHP Basics

4.5-hour online video course for web professionals

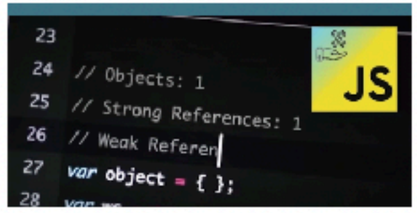
READ MORE →



JavaScript Design Patterns for Web Apps

4.5-hour online video course for web professionals on JavaScript

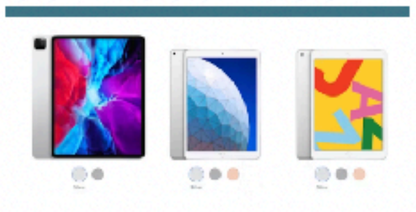
READ MORE →



Professional JS – Features You Need to Know

4.5-hour online video course for web professionals on JavaScript

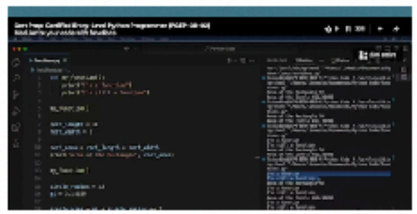
READ MORE →



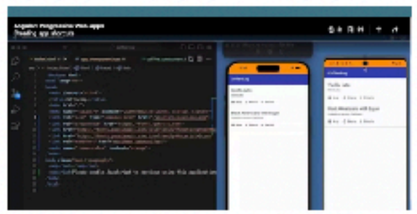
iPad Naming



ChatGPT and AI for



Certified Entry-Level



Angular Progressive

11ty

Eleventy is a simpler static site generator

Blog

11ty is joining Font Awesome

Versions

Stable 2.0.1

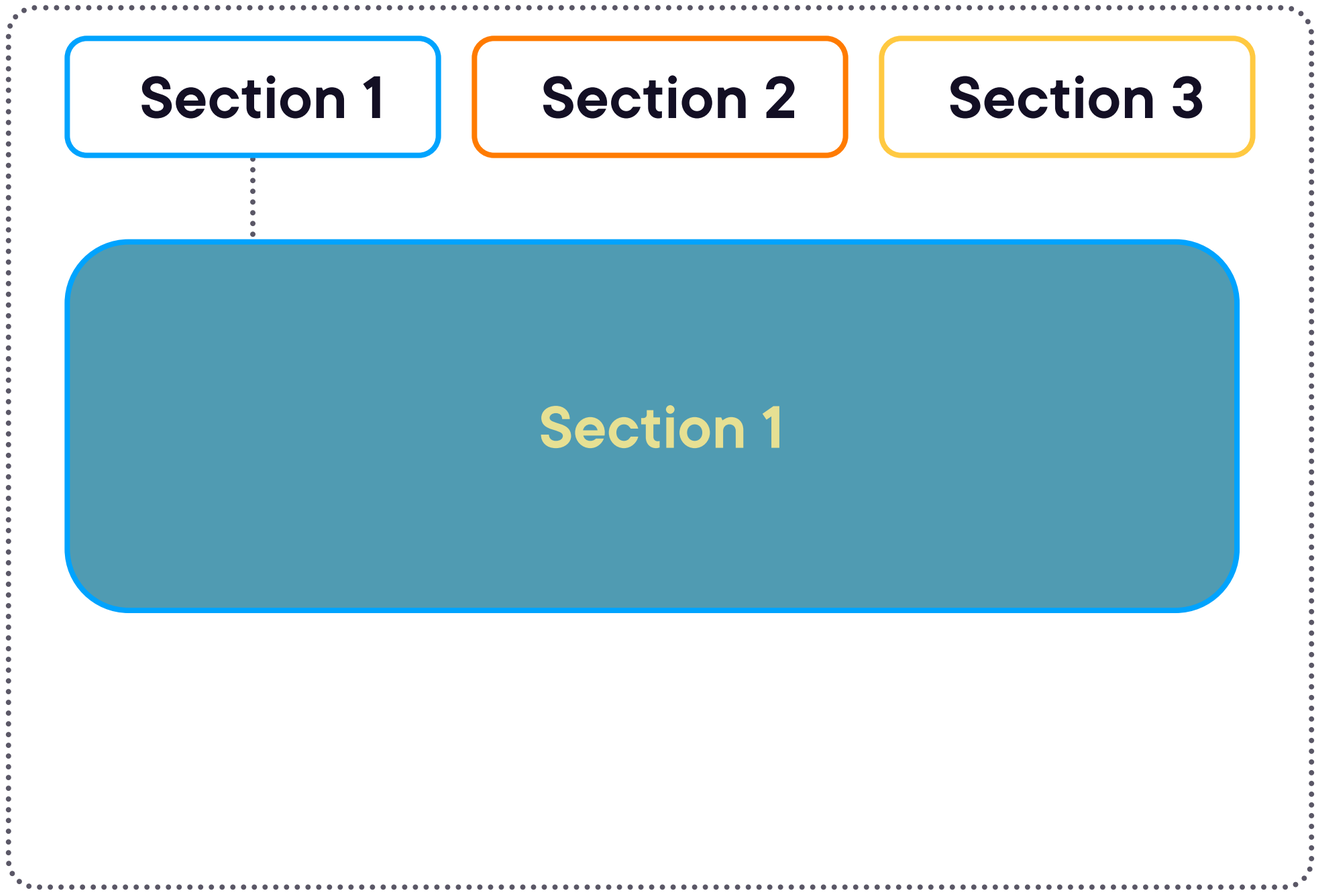
Quick Start

Eleventy requires Node.js 14. You can check whether or not you have Node installed by running `node --version` in a Terminal. (*Well, wait—what is a Terminal?*) If `node` is not found or it reports a version number below 14, you will need to install Node.js before moving on.

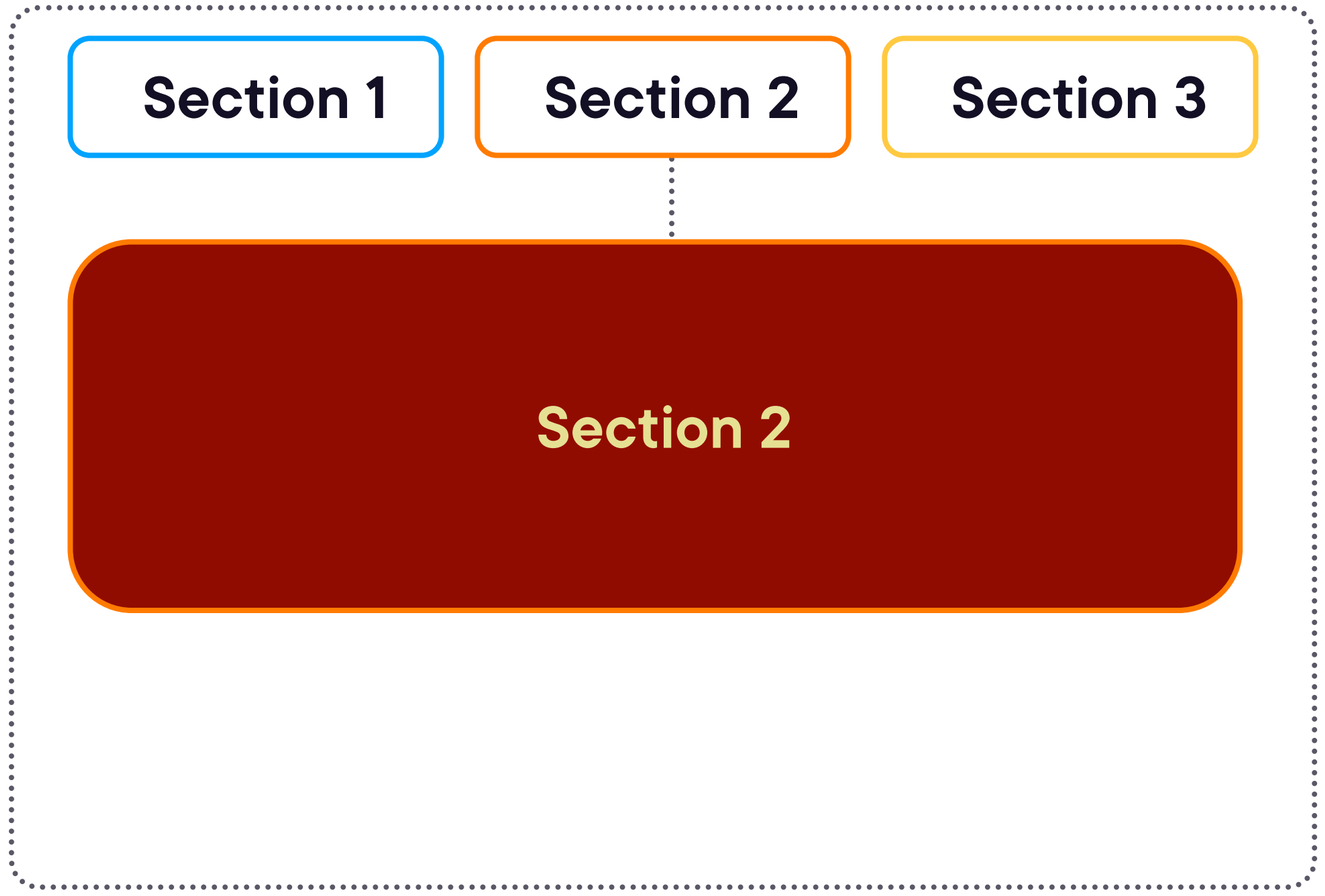


Client-side Routing

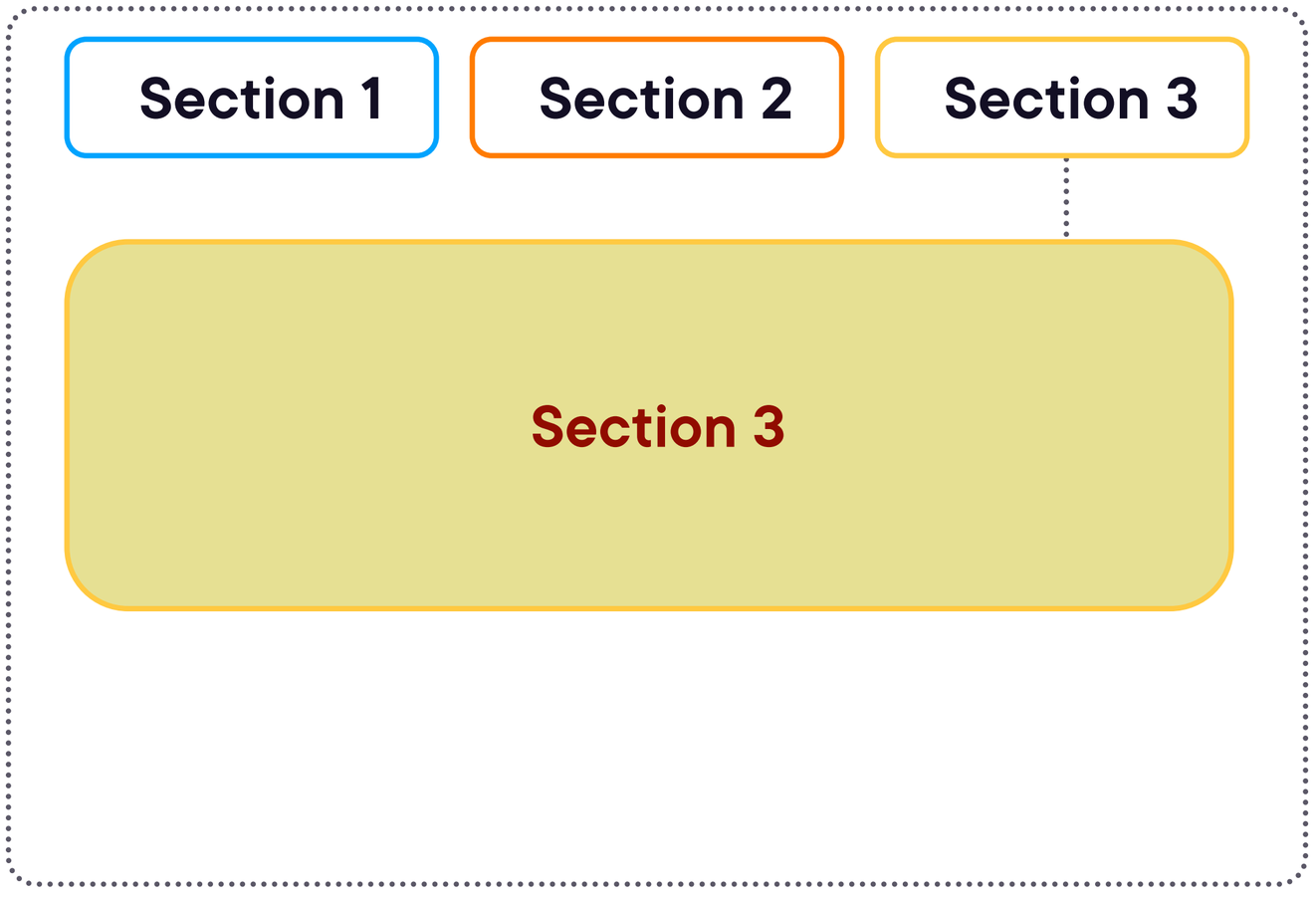
Single Page Application



Single Page Application



Single Page Application



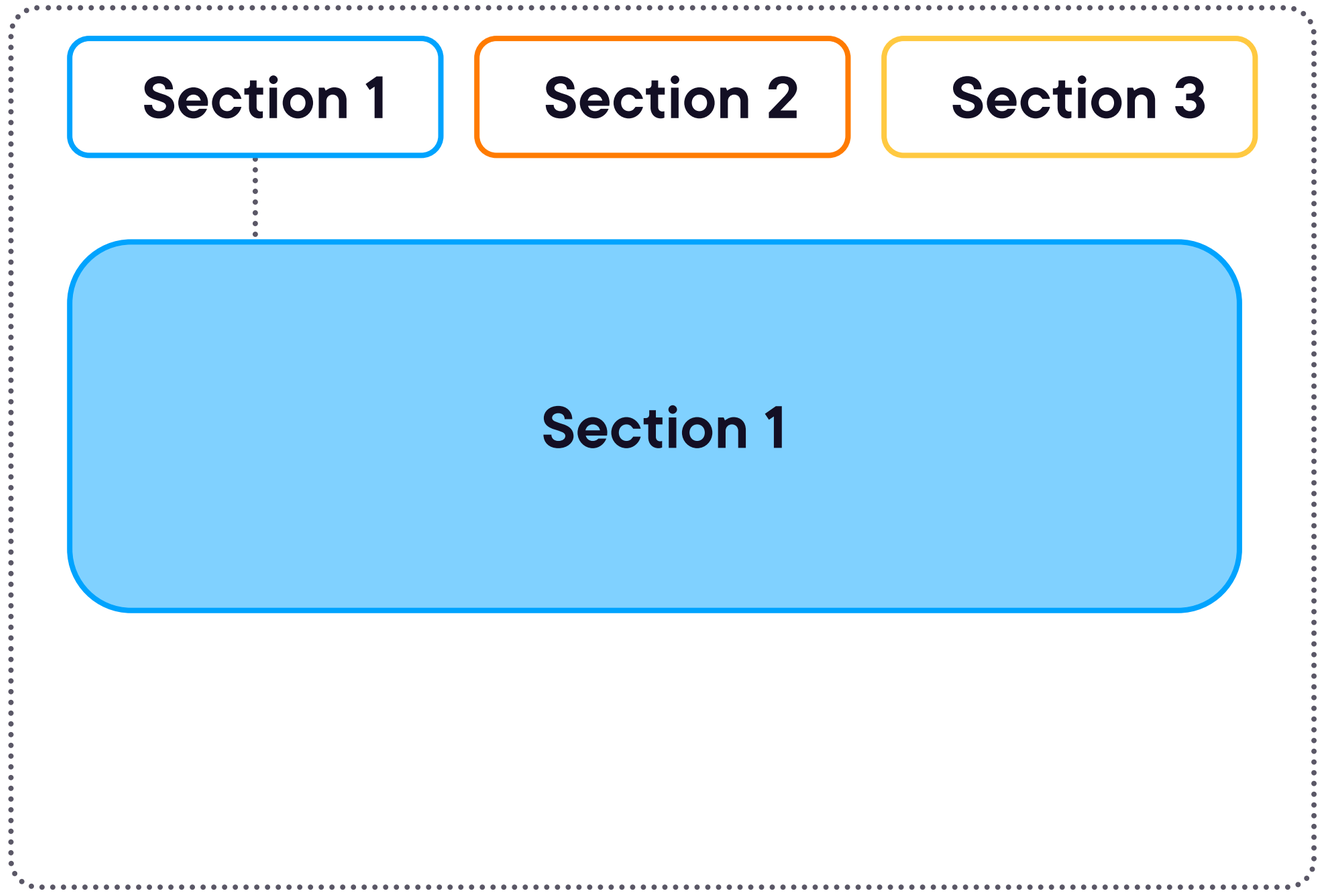
Single Page Applications: how to change content

Remove Previous Page and Inject New Page into the DOM

Hide Previous Page and Show Current Page

Single Page Application

Remove Previous Page and Inject New Page into the DOM

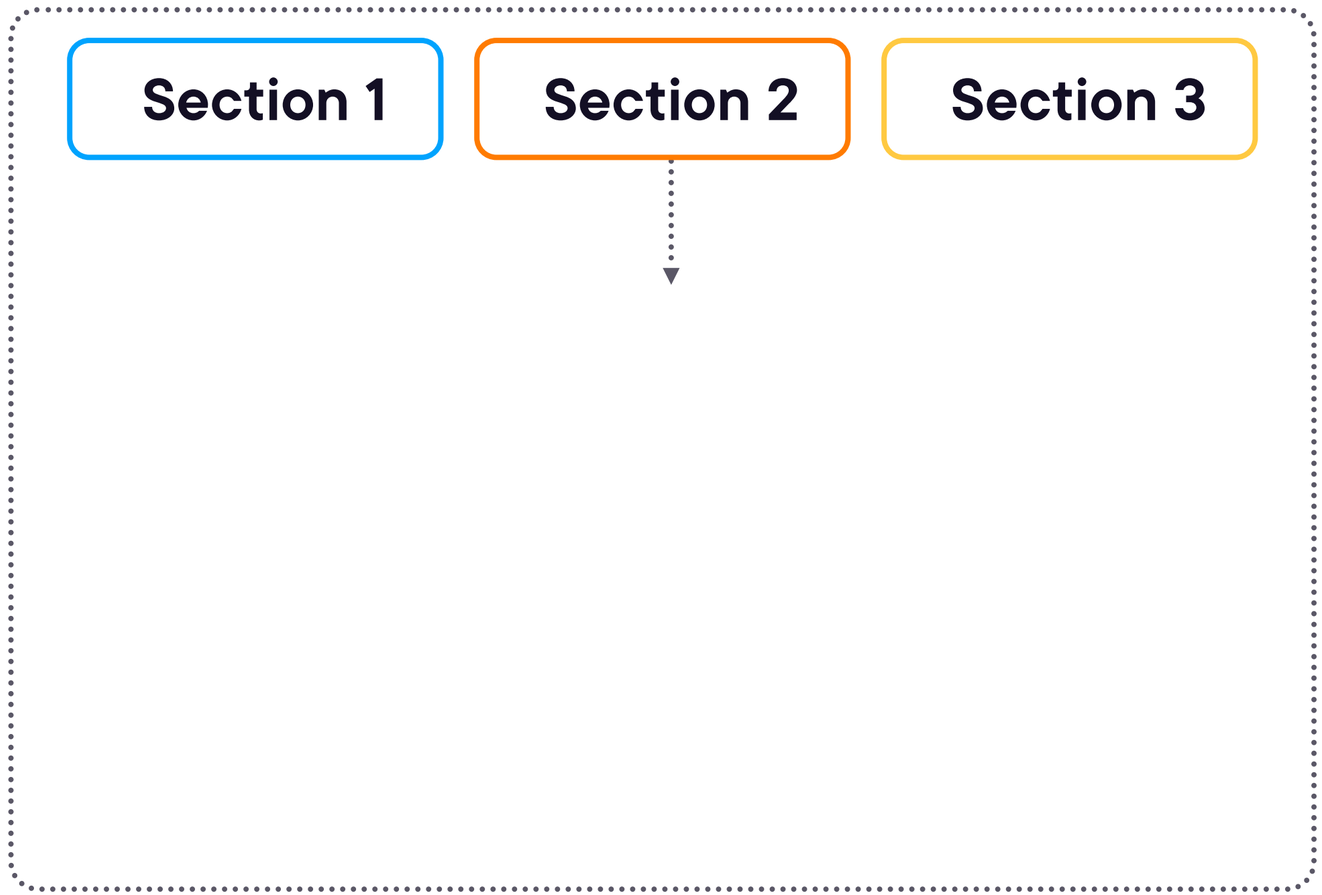



```
<nav>
  <a href="">Section 1</a>
  <a href="">Section 2</a>
  <a href="">Section 3</a>
</nav>


<section id="section1">
</section>
```

Single Page Application

Remove Previous Page and Inject New Page into the DOM

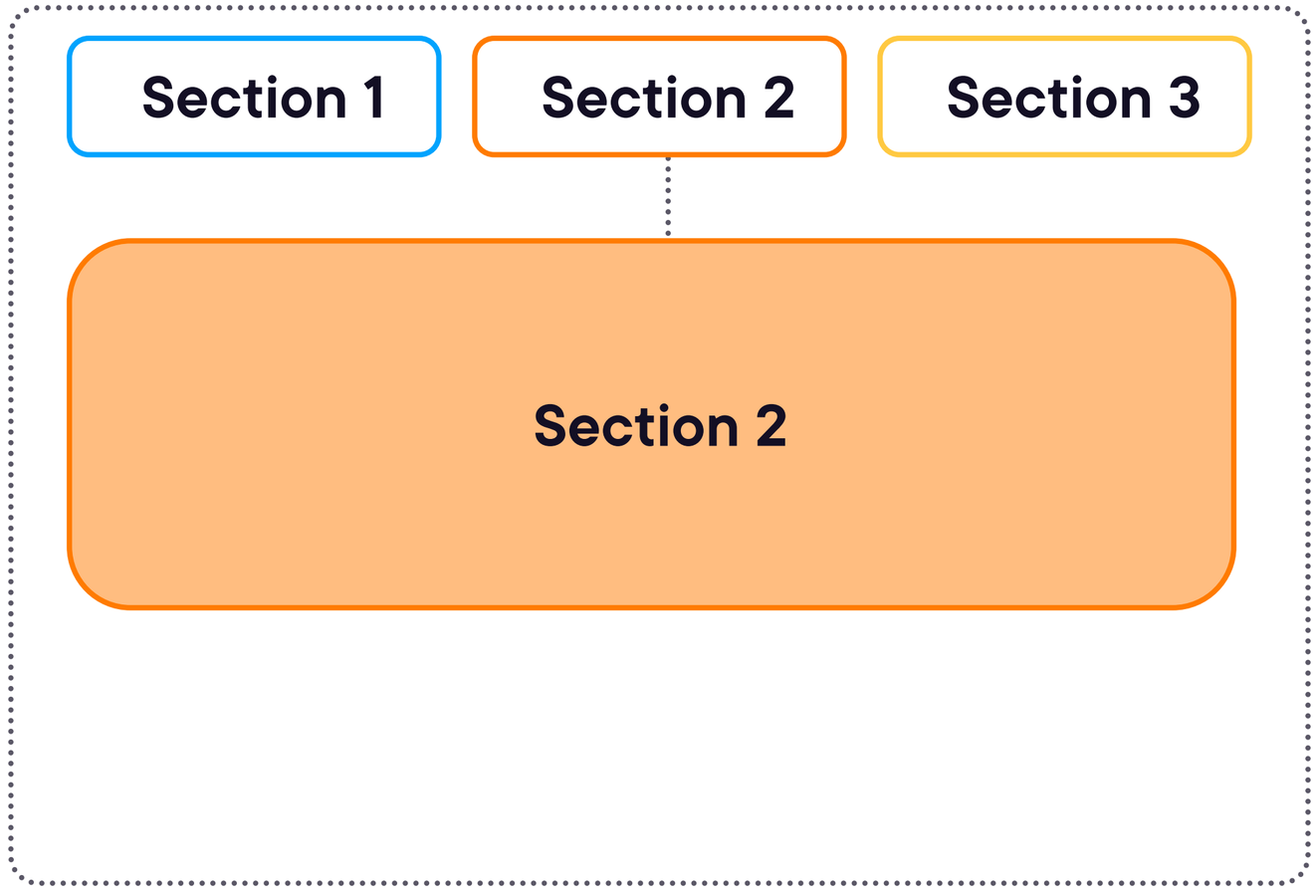



```
<nav>
  <a href="">Section 1</a>
  <a href="">Section 2</a>
  <a href="">Section 3</a>
</nav>


<section id="section1">
</section>
```

Single Page Application

Remove Previous Page and Inject New Page into the DOM

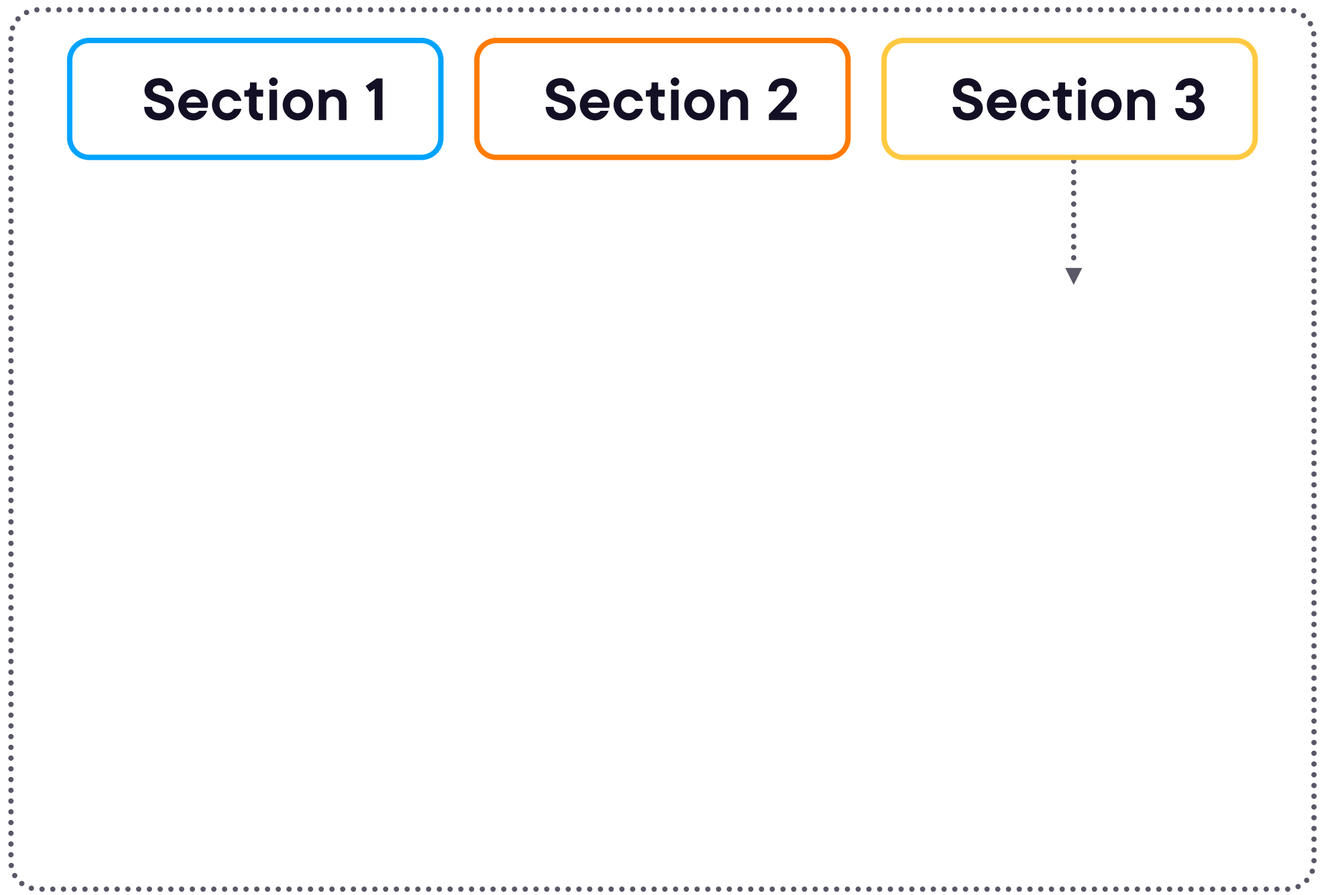



```
<nav>
  <a href="">Section 1</a>
  <a href="">Section 2</a>
  <a href="">Section 3</a>
</nav>

<section id="section2">
</section>
```

Single Page Application

Remove Previous Page and Inject New Page into the DOM

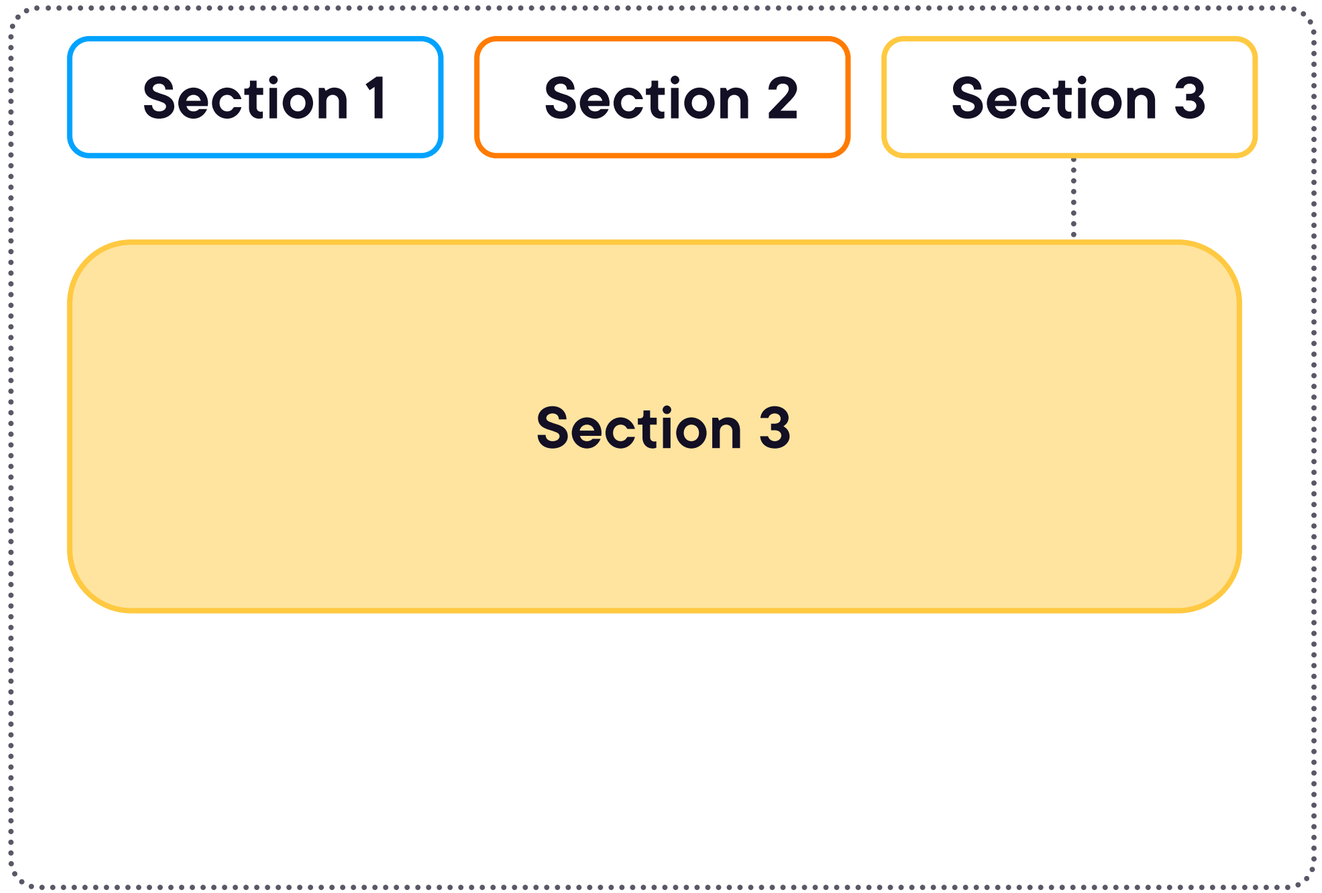



```
<nav>
  <a href="">Section 1</a>
  <a href="">Section 2</a>
  <a href="">Section 3</a>
</nav>


<section id="section2">
</section>
```

Single Page Application

Remove Previous Page and Inject New Page into the DOM

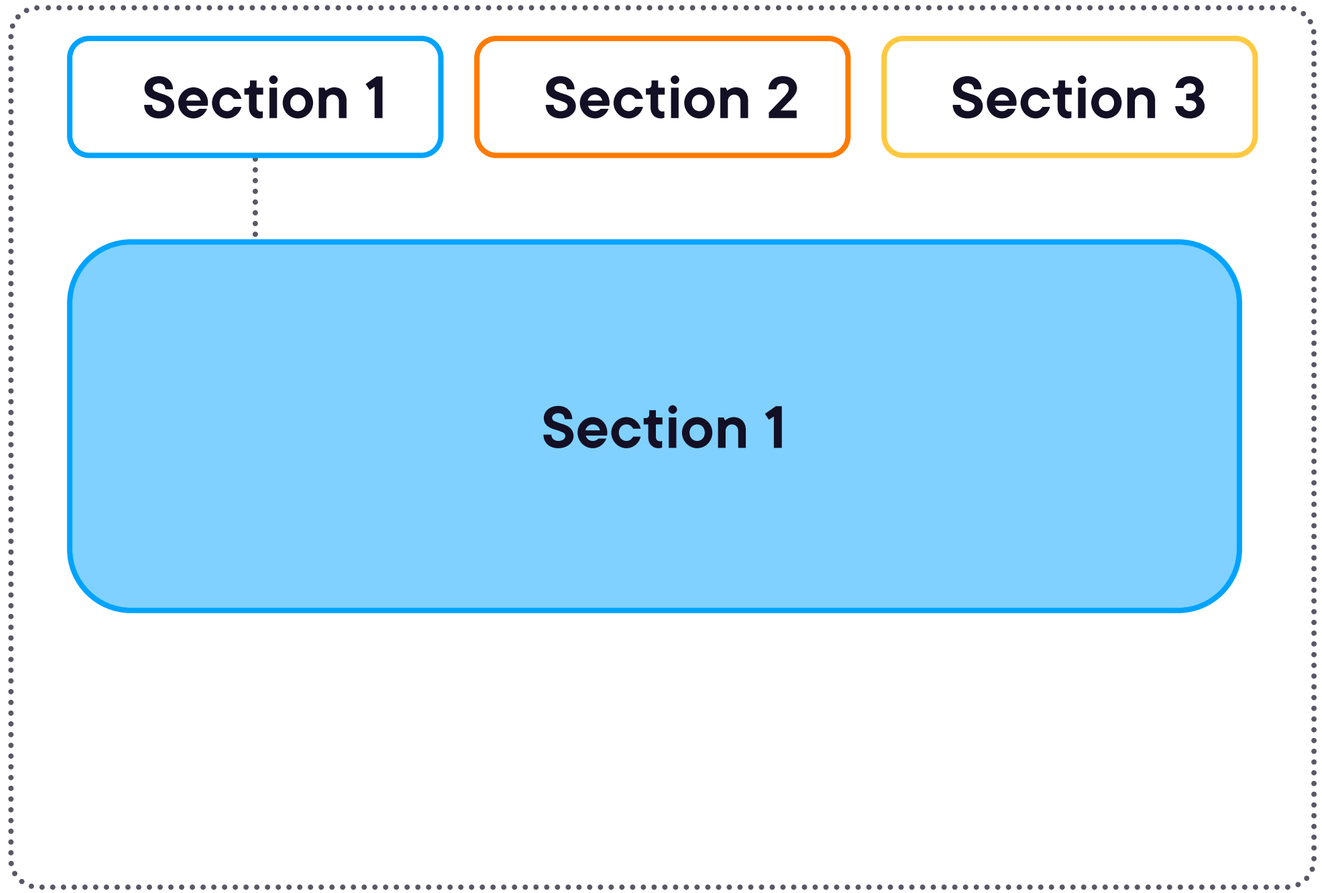



```
<nav>
  <a href="">Section 1</a>
  <a href="">Section 2</a>
  <a href="">Section 3</a>
</nav>

<section id="section3">
</section>
```

Single Page Application

Hide Previous Page and Show Current Page




```
<nav>
  <a href="">Section 1</a>
  <a href="">Section 2</a>
  <a href="">Section 3</a>
</nav>

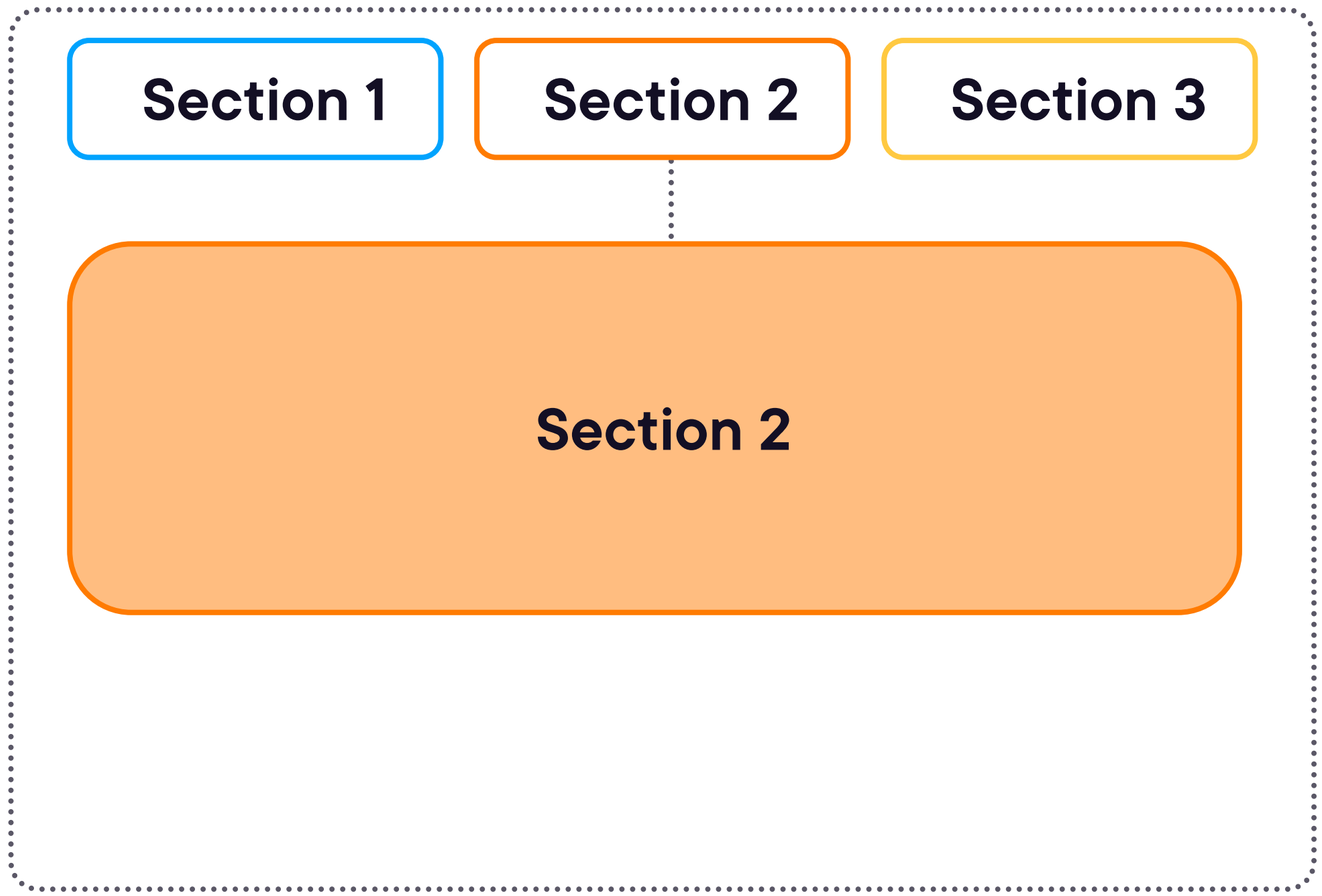

<section id="section1">
</section>

<section id="section2" hidden>
</section>

<section id="section3" hidden>
</section>
```

Single Page Application

Hide Previous Page and Show Current Page




```
<nav>
  <a href="">Section 1</a>
  <a href="">Section 2</a>
  <a href="">Section 3</a>
</nav>

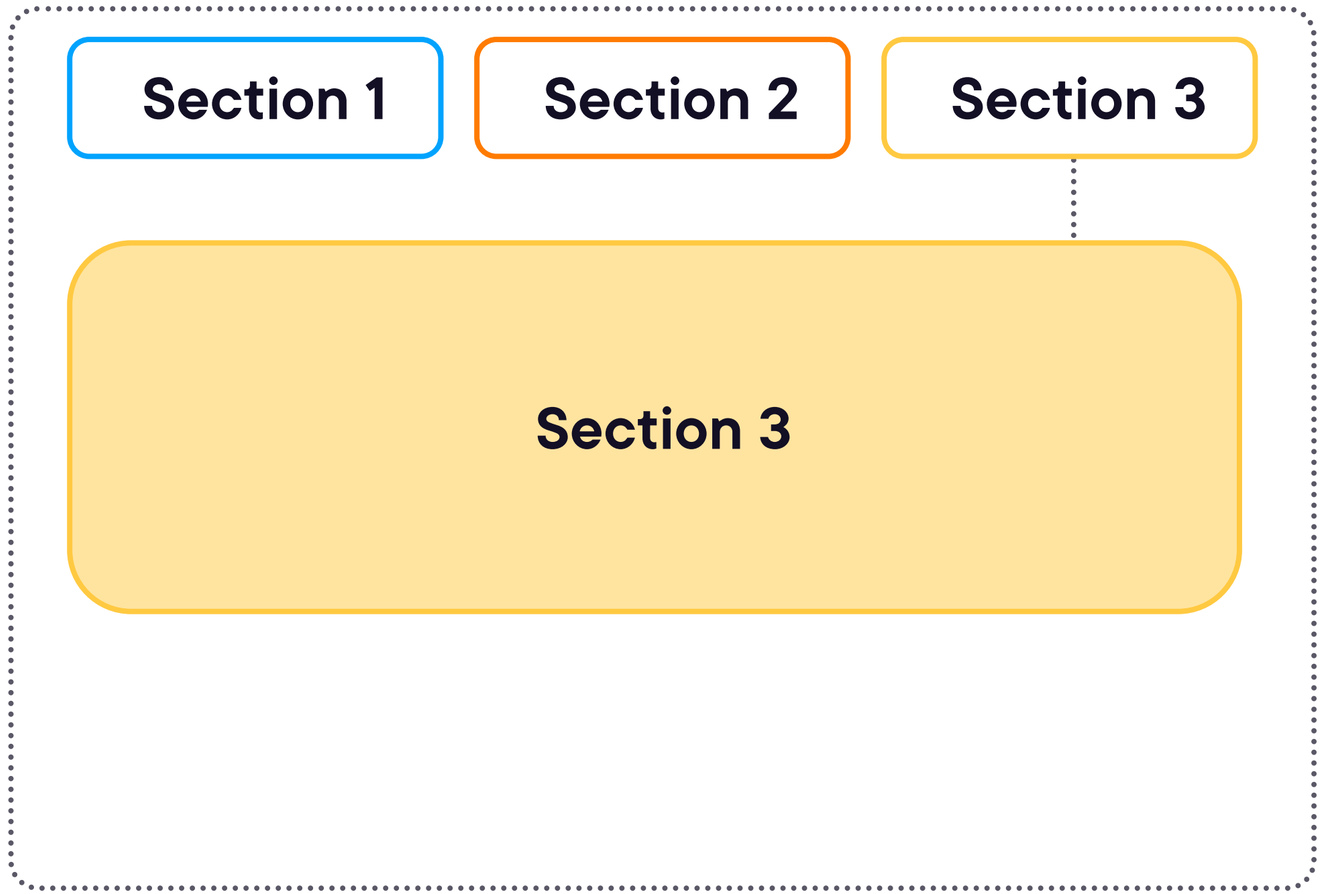
<section id="section1" hidden>
</section>

<section id="section2">
</section>

<section id="section3" hidden>
</section>
```

Single Page Application

Hide Previous Page and Show Current Page




```
<nav>
  <a href="">Section 1</a>
  <a href="">Section 2</a>
  <a href="">Section 3</a>
</nav>

<section id="section1" hidden>
</section>

<section id="section2" hidden>
</section>

<section id="section3">
</section>
```

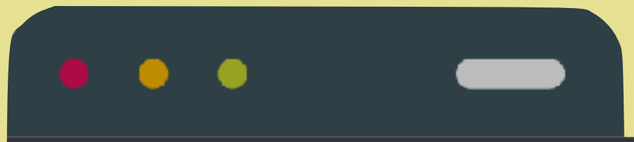
Single Page Application and Router

- We want to change the content of the page based on what the user select:
  - Home Page: menu
  - Details of one particular product
  - Order: cart details and order form

We won't have multiple HTML files, we will use the DOM APIs and Web Components

We will use the History API to push new entries to the navigation history

History API



We can push new "fake" navigation URLs and listen for changes

```
// pushing a new URL; the second argument is unused
history.pushState(optional_state, null, "/new-url");

// to listen for changes in URL within the same page navigation
window.addEventListener("popstate", event ⇒ {
   let url = location.href;
});
```



Web Components



DEFINITION

Web Component

A design pattern to create a modular, reusable building block for web development that encapsulates a set of related functionality and user interface elements.



DEFINITION

Web Component

In short, your own custom HTML tag element.

Web Components

- Compatible with every browser
- It's actually a set of standards:
  - **Custom Elements**
  - **HTML Templates**
  - **Shadow DOM**
- It's similar to the idea of components on most of the libraries for JavaScript
- We have freedom of choice on how to define them and use them



DEFINITION

Custom Element

A way to define new, reusable HTML elements with custom behavior and functionality using JavaScript.

Custom Elements

We can define our own HTML tags using the Custom Elements API

```
class MyElement extends HTMLElement {
   constructor() {
      super();
   }
}

customElements.define("my-element",
                      MyElement);
```

```
<body>
  <my-element></my-element>
</body>

<script>
document.createElement("my-element");
</script>
```



WARNING

The HTML tag we define must contain a hyphen (-) to assure future compatibility

Custom Elements with Attributes

We can define our own custom attributes using the data-* spec.

```
class MyElement extends HTMLElement {
   constructor() {
      super();
      this.dataset.language
   }
}

customElements.define("my-element",
                       MyElement);
```

```
<body>
  <my-element data-language="en">
  </my-element>
</body>
```

Custom Elements Lifecycle

We can override some methods of the super class

```
class MyElement extends HTMLElement {

  constructor() {              // Set up initial state, event listeners, etc.
     super();
  }
  connectedCallback() { }     // The element is added to the document
  disconnectedCallback() { } // The element is removed to the document
  adoptedCallback() { }.      // The element has been moved to a new document

  attributeChangedCallback(name, oldValue, new Value() { }

}

customElements.define("my-element", MyElement);
```

Custom Elements with Slots

The slot is the content that we can define as the component's children.
With templates we can have more than one slot.

```
class MyElement extends HTMLElement {
   constructor() {
      super();
   }
}

customElements.define("my-element",
                       MyElement);
```

```
<body>
  <my-element>
     <div>
        <h2>Slot of My Element</h2>
     </div>
  </my-element>
</body>
```



DEFINITION

Template Element

Fragments of markup that can be cloned and inserted into the document at runtime, with reusable HTML content that can be rendered and modified dynamically.

Template Element

It allow us to define HTML content that is not going to be parsed by the browser and it's going to be available for usage only if we clone it.

```
<template id="template1">
   <header>
      <h1>This is a template</h1>
      <p>This content is not rendered initially</p>
   </header>
</template>
```

Template Element Cloning

We clone the template and we append it as a child; typically in **`connectedCallback`** method of the Custom Element.

```
connectedCallback() {
  const template = document.getElementById("template1");
  const content = template.content.cloneNode(true);
  this.appendChild(content);
}
```

Where to define HTML for a Custom Element

- Use DOM APIs
- Use a <template> in the main HTML
- Use an external HTML file loaded with **fetch** (it can be prefetched)
  - Using innerHTML
  - Using DOMParser



IDEAS

Make template engines using tagged literal strings, such as:

```
utahjs`<h1>${title}</h1>`
```

Lit-html uses this technique

Where to define CSS for a Custom Element

- There are several alternatives
  - Use CSSOM APIs
  - Add a <script> to a <template>
  - Add a <link> in the <template>
  - Use an external CSS file loaded with **fetch** (it can be prefetched) **and injected in the Shadow DOM as a <style>**



WARNING

By default, the nodes of our custom element are part of the same page's DOM, so CSS style declaration applies to all the document.

Template Element

It allow us to define HTML content that is not going to be parsed by the browser and it's going to be available for usage only if we clone it.

```
<template id="template1">
  <style>
    /* this declaration also changes h1s outside of the custom element */
    h1 { color: red }
  </style>
  <header>
      <h1>This is a template</h1>
      <p>This content is not rendered initially</p>
  </header>
</template>
```



DEFINITION

Shadow DOM

A private, isolated DOM tree within a web component that is separate from the main document's DOM tree

Shadow DOM

- Allows more control over styling and encapsulation of functionality of a Custom Element.
- By default, CSS declared in the main DOM won't be applied to the Shadow DOM.
- CSS declared in the Shadow DOM applies only on there.
- There are new pseudo-classes and pseudo-element to allow communication between DOMs in stylesheets.
- It can be opened or closed defining visibility from the outer DOM

Shadow DOM

We have to create a Shadow DOM manually in our Custom Element, typically in the constructor and we save it

```
class MyElement extends HTMLElement {
   constructor() {
      super();
      this.root = this.attachShadow({ mode: "open" });
   }

   connectedCallback() {
      this.root.appendChild( ... )
   }
}
```

Template Element on a Shadow DOM

```
<template id="template1">
  <style>
    /* this declaration ONLY changes h1s inside this custom element */
    h1 { color: red }
  </style>
  <header>
      <h1>This is a template</h1>
      <p>This content is not rendered initially</p>
  </header>
</template>
```



Reactive Programming



DEFINITION

Reactivity

It let us keep data synced using a single source of truth pattern

Proxy

We wrap an object with a Proxy that can handle different operations, such as when a consumer is setting or getting a property

```
const original = {
  name: 'John Doe',
  age: 30
};

const s = new Proxy(original, handler);

console.log(s.age);   // 30 years old
```

```
const handler = {
  get: function(target, prop) {
    if (property === 'age') {
      return target[prop] + ' years old';
    } else {
      return target[prop];
    }
  }
};
```

Proxy

We can use a proxy for validation, data binding and reactive programming

```
const original = {
  name: 'John Doe',
  age: 30
};

const s = new Proxy(original, handler);
s.age = 40;
```

```
const handler = {
  set: function(target, prop, value) {
     target[property] = value;
     if (property === 'age') {
        // UPDATE UI
     }
};
```



IDEAS

This is just a basic sample, you can decouple the update even more using other techniques, such as with Signals.



Web Apps Power

Learn PWA

A course that breaks down every aspect of modern progressive web app development.

25 activities

0% completed

Save your progress
Sign in

1 Welcome to Learn Progressive Web Apps!

Article

A course that breaks down every aspect of modern progressive web app development.

Read article

2 Progressive Web Apps

Article

3 Getting started

Article

4 Foundations

Article

5 App design

Article

6 Assets and data

Article

Learn PWA

A course that breaks down every aspect of modern progressive web app development.

25 activities

0% completed

Save your progress
Sign in

Installation

Once the user installs your PWA, it will:

- Have an icon in the launcher, home screen, start menu, or launchpad.
- Appear as a result when a user searches for the app on their device.
- Have a separate window within the operating system.
- Have support for specific capabilities.

★ **Note:** While installation differs per browser and operating system, you don't need to create any kind of package or executable for your PWA to be installed from a browser.

Installation criteria

Every browser has a criterion that marks when a website or web app is a Progressive Web App and can be installed for a standalone experience. The metadata for your PWA is set by a JSON-based file known as the Web App Manifest, which we will cover in detail in the next module.

Learn PWA

A course that breaks down every aspect of modern progressive web app development.

25 activities

0% completed

Save your progress
Sign in

Offline data

To build a solid offline experience, your PWA needs storage management. In the caching chapter you learned that cache storage is one option to save data on a device. In this chapter, we'll show you how to manage offline data, including data persistence, limits, and the available tools.

Storage

Storage is not just about files and assets, but can include other types of data. On all browsers that support PWAs, the following APIs are available for on-device storage:

- IndexedDB: A NoSQL object storage option for structured data and blobs (binary data).
- WebStorage: A way to store key/value string pairs, using local storage or session storage. It's not available within a service worker context. This API is synchronous so it's not recommended for complex data storage.
- Cache Storage: As covered in the Caching module.

★ **Note:** On supported platforms, you can also use the FileSystem Access API to read and write to and from the user's local filesystem, with permission. Since you are writing files directly to the file system the browser won't assign the data to your origin's quota, and any other application can interact with those files, if the user allows it.

Learn PWA

A course that breaks down every aspect of modern progressive web app development.

25 activities

0% completed

Save your progress
Sign in

Capabilities

Progressive Web Apps can do more than just rendering content on the screen or connecting to web services. PWAs can deal with files from the file system, they can interact with the system's clipboard, they can access hardware that is connected to the device, and much more. Every Web API is available for your PWA, and there are some extra APIs available when your app is installed.

You can use What Web Can Do Today to know what's possible on each platform. For individual APIs or capabilities, you can use Can I Use or the browser compatibility tables on MDN.

Always check for feature support

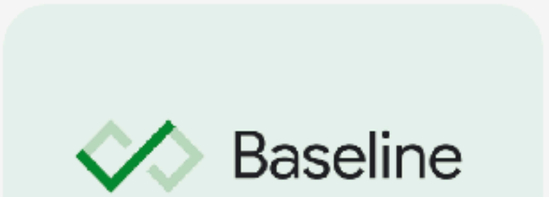
The first letter in PWA stands for progressive, and it comes from the idea of progressive enhancement and feature detection. You should not expect your app to work the same way on every device. The diversity of contexts and abilities on billions of devices in different countries makes PWAs an excellent platform, thanks to their progressiveness.

This means that you need to develop your app in layers that may not be available on every device and to check availability before usage.

Baseline

Web Platform Baseline brings clarity to information about browser support for web platform features.

Baseline gives you clear information about which web platform features are ready to use in your projects today. When reading an article, or choosing a library for your project, if the features used are all part of Baseline, you can trust the level of browser compatibility. By aligning with Baseline, there should be no surprises when testing your site. More about Baseline.



How do things become Baseline?

Baseline has two stages:

- **Newly available:** The feature becomes supported by all of the core browsers, and is therefore interoperable.
- **Widely available:** 30 months has passed since the newly interoperable date. The feature can be used by most sites without worrying about support.

The core browser set

- Chrome (desktop and Android)
- Edge
- Firefox (desktop and Android)
- Safari (macOS and iOS)

Baseline 2024

Learn about the features that became part of Baseline during 2024.



What is Baseline?

Web Platform Baseline gives you clear information about which web platform features are safe to use in your projects today.

Read more

Baseline Newly Available

Follow along as new features become part of Baseline 2024.

See the features


WWDC07

Main Fears of Vanilla JS



- Client-side Routing ✅
- Too Verbose 🧠
- State Management ✅
- Templating ✅
- Reusable Components ✅
- Learning Curve ✅
- Reinventing the Wheel 🧠

goto;



Lessons



Not everything

is a nail

Take good ideas

Developer Experience (DX) is not #1 priority

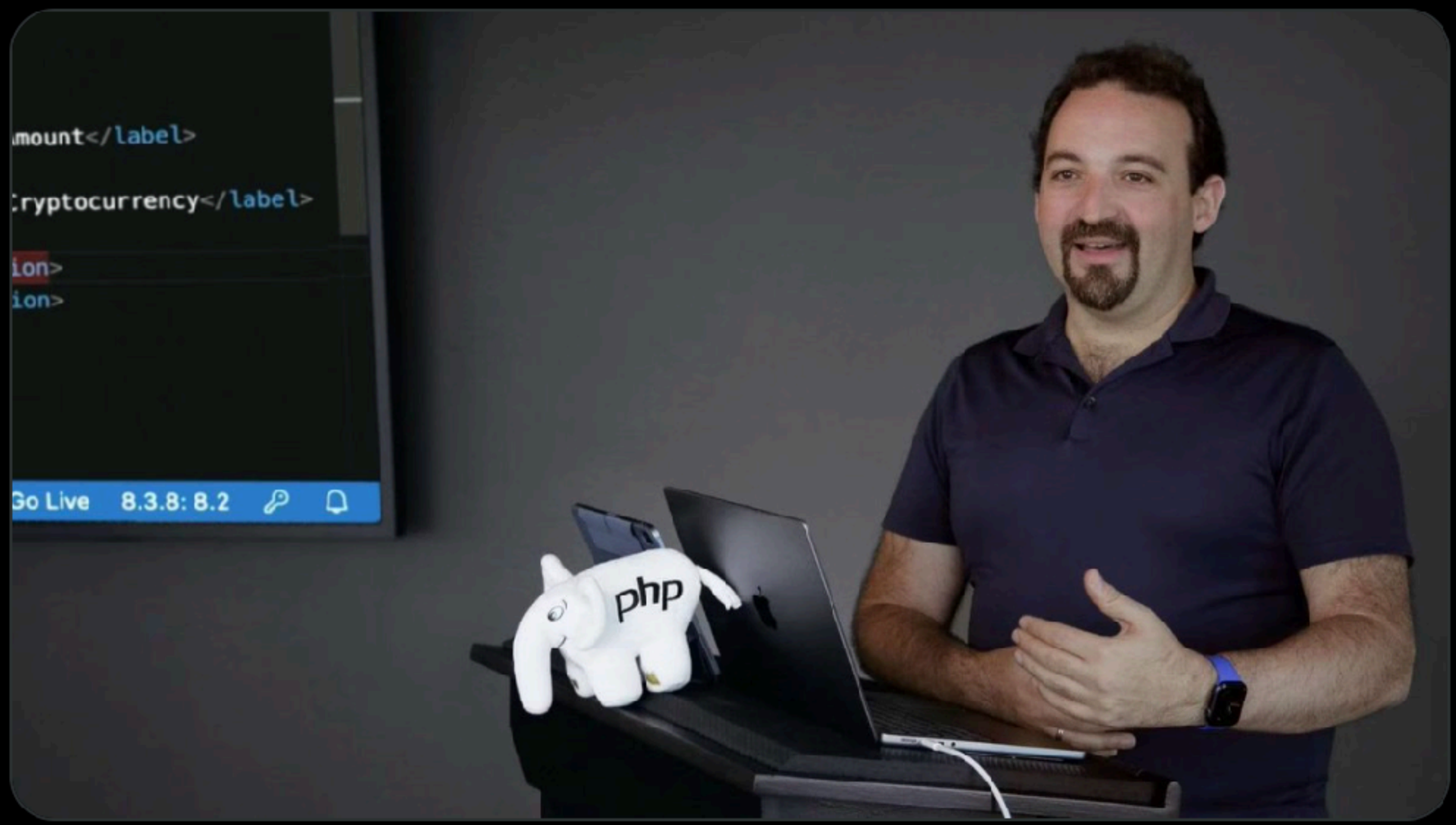
Avoid tech fandom

Maximiliano Firtman @firt · Aug 27

🐘**PHP** powers 70% of the web, yet it's increasingly difficult to find developers with the basic knowledge to maintain **PHP** projects.

That's why I've published a workshop with @FrontendMasters on **PHP** Basics, updated for 2024!

firt.dev/php-course/



Using latest libraries and trends are **NOT** goals

goto;



Goals

Good User Experience

Accessibility

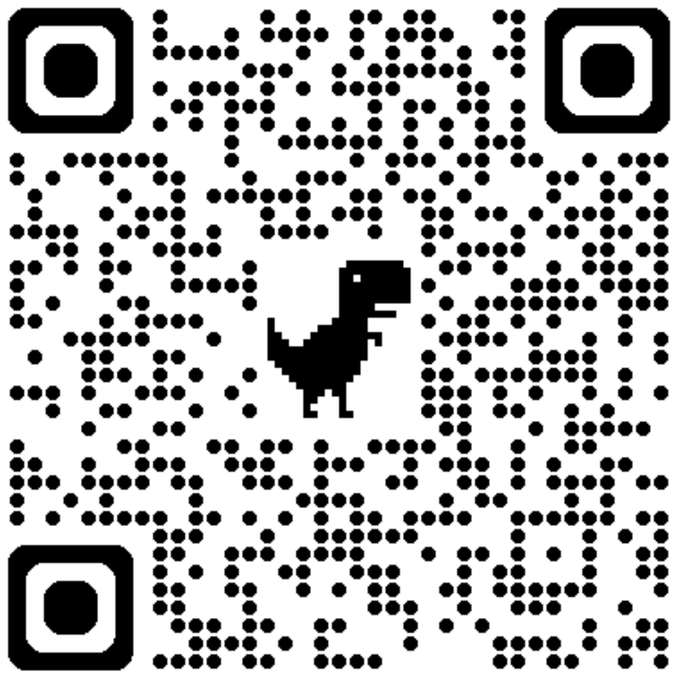
High Performance

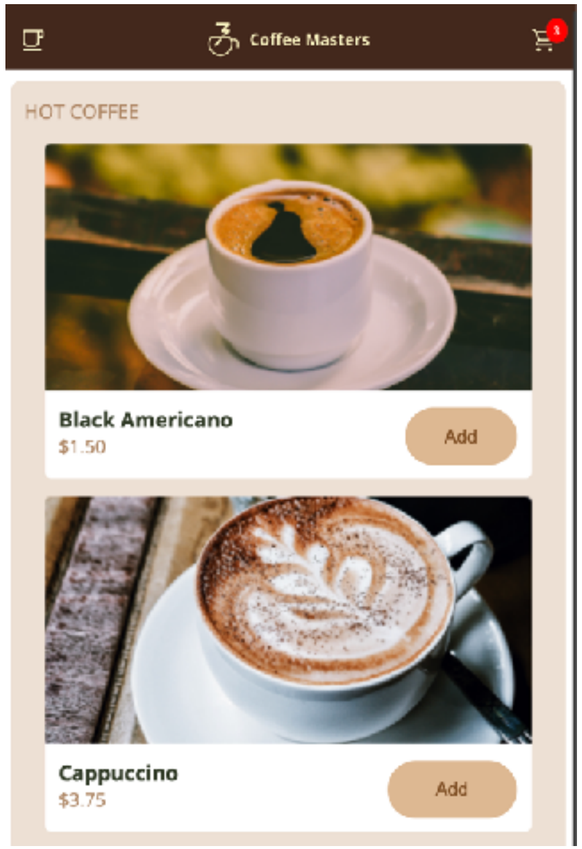
Good software architecture

Your Users

Source code and Tutorial

firtman.github.io/vanilla





Thanks!

MAXIMILIANO FIRTMAN

@firt firt.dev

goto;

Rate this session

in the GOTO Guide app

Help us make GOTO even better