

INFS3202/7202 – Web Information Systems

Lecture Week 9: Accessibility & Progressive Web Apps

Dr Aneesha Bakharia (Senior Lecturer, EECS)
Email: a.bakharia1@uq.edu.au

Contents

- 01 RiPPL E Weekly Tasks
- 02 Preparing for your Code Review (in Week 10)
- 03 Accessibility
- 04 Responsive Web Applications
- 05 Progressive Web Applications (PWA's)
- 06 Using Large Language Models – Part 2
 Prompt Engineering for code

Course Updates

Issue/Feedback	Change
Design Document Assessment Item	<ul style="list-style-type: none">• Grades released• Feedback provided on where grades were deducted
Code Review	<ul style="list-style-type: none">• In the Week 9 Lab you can work on your project• During the Week 10 Lab you will need to complete your code review with a demonstrator• If you can't make your scheduled Lab please email a.bakharia1@uq.edu.au to arrange an alternate time. The Code Review must be conducted in Week 10 unless you have an extension.• The Code Review will be recorded via Zoom (to keep a record)
Week 10 Lecture	<ul style="list-style-type: none">• Guest lecture from AWS Solution Architects• AWS technologies for deployment will be covered eg Route 65, EC2, etc

Weekly Activities (RiPPL E Tasks)

Weekly Activities Rubric (Weighting 10%)

- From Week 2 to Week 10 (i.e. 9 weeks), you are required to participate in an online weekly activity where you create, moderate and answer questions.
 - No extensions are permitted.
 - Your grade will be based on the best **7 out of 9** submissions.
 - You get **1.43 marks for each week** you meet the requirement of the create, moderate or answer (practice) weekly assessment on the RiPPL E platform.
 - Due weekly at 3pm on Friday from Week 2 - 10, except in Week 6 where the due date is Thurs 28/3/24 at 3pm.
-
- Many students already have achieved 10 Marks
 - If you have received 10 Marks, you won't be able to improve your marks in Week 9 and 10 but the weekly activity is useful for learning
 - If you have less than 10 Marks, you should complete Week 9 and Week 10 RiPPL E Activities to improve your mark.

Code Review Assessment Item Rubric

INFS3202 & INFS7202

Rubric

Web Project Code Review (10%)

No Marks	Half Marks	Full Marks			
A version of the student's <i>Project</i> was not demonstrated during the Week 10 Lab, or was substantially incomplete . The student did not demonstrate an understanding of Web Technologies during the Code Review. The student did not articulate issues they experienced during the production of their Web Project, with poor oral communication evident.	0.00 % 0.00 % 0.00 %	A version of the student's <i>Project</i> was demonstrated during the Week 10 Lab, but was not near half-complete & partially functional. The student demonstrated some understanding of Web Technologies during the Code Review. The student articulated some issues they experienced during the production of their Web Application.	2.00 % 2.00 % 1.00 %	A near half-complete & partially functional version of the student's <i>Web Project</i> was demonstrated during the Week 10 Lab. The student demonstrated a strong understanding of Web Technologies during the Code Review. The student articulated issues they experienced during the production of their Web Application, reflecting on how they might make improvements.	4.00 % 4.00 % 2.00 %

Preparing for your Code Review

- **Know the structure of your application?**

Know the conventions of CodeIgniter or your framework that you are using
e.g. How routes link to controllers link to views? Which folders store the files?

- **Know how functionality works?**

How is the Admin panel implemented?
e.g. How is data retrieved from a database displayed in a HTML table?
e.g. How have you used models?

- **Be prepared to talk about issues you've encountered:**

What have you found difficult?

- **How will you implement the remaining functionality?**

What is your approach to implement the remaining functionality?

Week 10 Lab – Code Review

- Must attend your allocated Contact
(Mainly because everybody needs to attend and we need to make sure there is enough time for the Demonstrator to chat with all attending students.)
- **Please come on time**
- If you can't make your allocated Lab, please email a.bakharia1@uq.edu.au to make alternate arrangements
- You don't need a Powerpoint
It is a discussion with your Demonstrator and not a presentation
- Your App does not have to be completed, but you must show progress that is equivalent for a half-way mark.

Accessibility

What is Web Accessibility?

Web accessibility means ensuring websites, tools, and technologies are designed and developed so that all people, including those with disabilities, can use them.

Inclusivity: About providing equal access and opportunities to everyone, regardless of their ability. This includes people with impairments like visual, auditory, physical, speech, cognitive, and neurological disabilities.

Key Benefits:

- 1. Usability:** Makes your site usable to the broadest group of people.
- 2. Legal Compliance:** Meets international standards and laws, reducing legal risks.
- 3. Broader Reach:** Accessible sites reach a wider audience, including the elderly and those with temporary disabilities.

Standards:

Follows guidelines such as the Web Content Accessibility Guidelines (WCAG), which provide criteria to improve accessibility.

Why is Accessibility Important?

Ethical Responsibility:

- Inclusivity: Ensures everyone has equal access to information and functionality.
- Equality: Supports the right to information and interaction for all users, regardless of their physical capabilities.

Legal Compliance:

- Regulations: Many countries have laws requiring digital accessibility (e.g., ADA in the USA, EN 301 549 in the EU).
- Avoid Penalties: Non-compliance can lead to legal actions and fines.

Business Benefits:

- Wider Audience: Accessibility increases your potential market as it includes users with disabilities, which represent about 15% of the global population.
- SEO Improvement: Accessible sites tend to have better search engine rankings due to good structure and usability.

Accessibility = Improve User Experience

- **Universal Design:**
Features that aid accessibility (like clear navigation and text-to-speech) often enhance the user experience for all visitors.
- **Flexibility:**
Accommodates a wider range of user preferences and needs, improving overall satisfaction.

Web Content Accessibility Guidelines (WCAG)

- WCAG provides a set of recommendations for making web content more accessible to people with disabilities.
- Developed by the Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C), the main international standards organization for the Internet.
- Widely adopted as the standard for web accessibility by organizations and governments worldwide.
- Forms the basis for many national laws governing digital accessibility.

WCAG – Key Principles

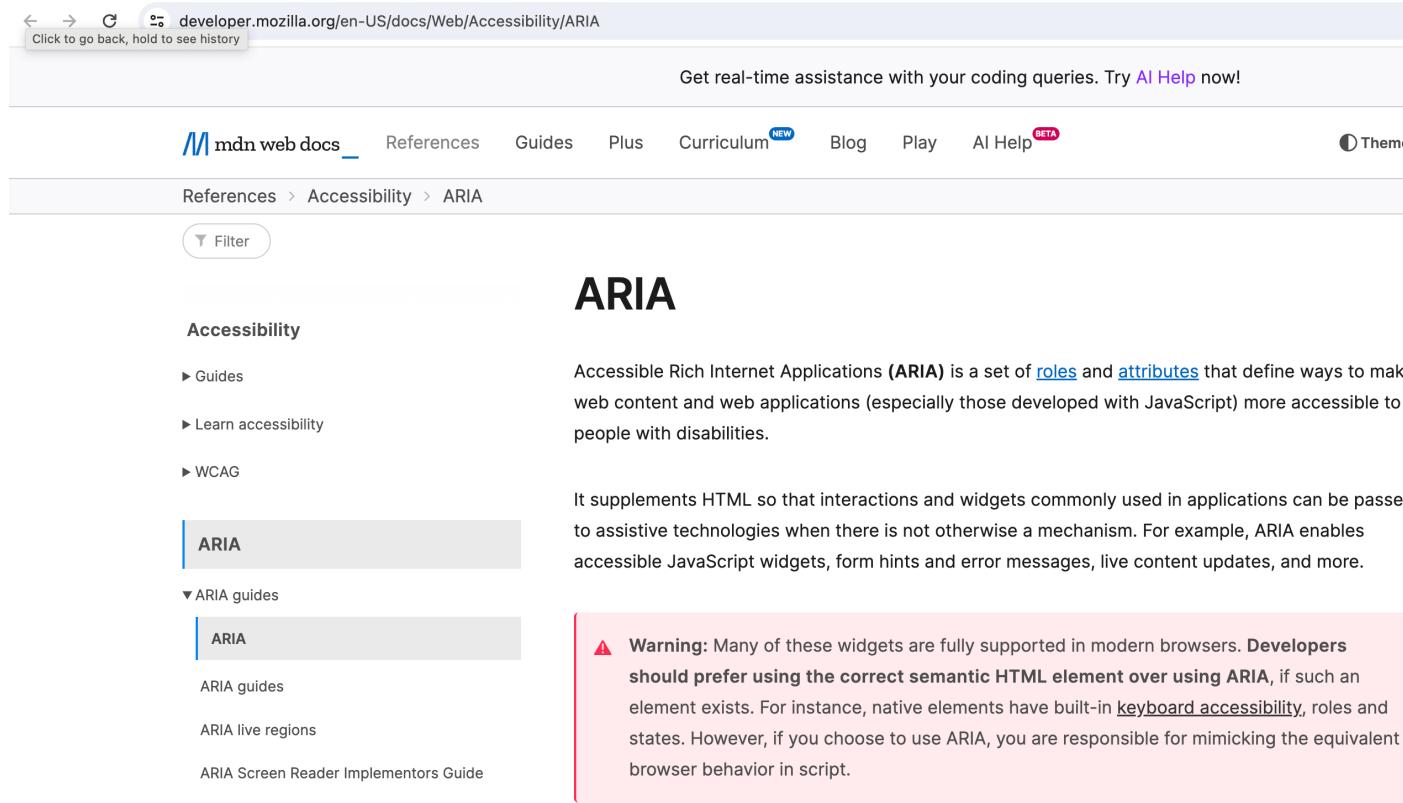
- **Perceivable:**
Information must be presented in ways that users can perceive (e.g., text alternatives for images, captions for videos).
- **Operable:**
User interface components and navigation must be operable (e.g., all functionality available from a keyboard).
- **Understandable:**
Information and operation of the user interface must be understandable (e.g., text is readable and understandable).
- **Robust:**
Content must be robust enough to be reliably interpreted by a wide variety of user agents, including assistive technologies.

WCAG Website

The screenshot shows the WCAG 2.1 Recommendation page. The URL in the address bar is <https://www.w3.org/TR/WCAG21/>. The page features a blue header bar with the W3C logo. On the left, there's a sidebar titled "W3C Recommendation" containing a "TABLE OF CONTENTS" with sections like Abstract, Status of This Document, Introduction, and various success criteria. The main content area has a large title "Web Content Accessibility Guidelines (WCAG) 2.1" and a subtitle "W3C Recommendation 21 September 2023". It includes links for "More details about this document", "This version", "Latest published version", "Latest editor's draft", "History", "Implementation report", "Previous Recommendation", "Editors", "Feedback", and "Errata". At the bottom, there's a copyright notice: "Copyright © 2020-2023 World Wide Web Consortium. W3C® liability, trademark and document use rules apply."

<https://www.w3.org/TR/WCAG21/>

Accessible Rich Internet Applications (ARIA)



The screenshot shows the MDN Web Docs page for ARIA. The URL in the address bar is <https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA>. The page title is "ARIA". The left sidebar has a tree view with "Accessibility" expanded, showing "Guides", "Learn accessibility", and "WCAG". Under "Accessibility", "ARIA" is expanded, showing "ARIA guides", "ARIA", "ARIA guides", "ARIA live regions", "ARIA Screen Reader Implementors Guide", "Using ARIA: Roles, states, and properties", "Multipart labels: Using ARIA for labels with embedded fields inside them", and "How to file ARIA-related bugs". The main content area starts with a definition of ARIA: "Accessible Rich Internet Applications (ARIA) is a set of [roles](#) and [attributes](#) that define ways to make web content and web applications (especially those developed with JavaScript) more accessible to people with disabilities." It then explains that ARIA supplements HTML to enable assistive technologies. A red warning box contains the text: "Warning: Many of these widgets are fully supported in modern browsers. Developers should prefer using the correct semantic HTML element over using ARIA, if such an element exists. For instance, native elements have built-in [keyboard accessibility](#), roles and states. However, if you choose to use ARIA, you are responsible for mimicking the equivalent browser behavior in script." Below this, a note states: "The first rule of ARIA is 'If you can use a native HTML element or attribute with the semantics and behavior you require already built in, instead of re-purposing an element and adding an ARIA role, state or property to make it accessible, then do so.'"

<https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA>

ARIA

ARIA helps make web content and web applications more accessible to people with disabilities, especially when traditional HTML falls short.

Bridge Gaps: It provides additional semantics that can be used by assistive technologies like screen readers.

Core Features of ARIA:

Roles: Define what an element is or does (e.g., role="button", role="navigation"). Roles help assistive technology understand the purpose of an element.

Properties and States: Describe the properties of elements or their current state (e.g., aria-expanded="false", aria-haspopup="true"). These attributes help communicate changes and features of interactive components to users.

When to Use ARIA:

Enhancement, Not Replacement: Use ARIA to enhance elements, not to replace basic HTML functionalities. Start with semantic HTML, then add ARIA for additional details if needed.

Complex Widgets: For interactive controls or complex widgets that HTML cannot define alone, such as sliders, drag-and-drop functionalities, or modal dialogs.

Basic Accessibility Features – Part 1

Text Alternatives:

Purpose: Provide alternatives for non-text content.

Examples: Alt text for images, labels for links, labels for buttons, transcripts for audio.

```
<!-- Accessible Version -->

<!-- Non-Accessible Version -->
  
  

<!-- Accessible Version -->

    Click here

<!-- Non-Accessible Version -->
Click here  
  

<!-- Accessible Version -->
<button aria-label="Search">
    <img alt="Search" data-bbox="242 785 714 825"/>
</button>
<!-- Non-Accessible Version -->
<button>
    
</button>
```

Basic Accessibility Features – Part 2

Semantic HTML:

Purpose: Use HTML elements according to their intended purpose.

Examples: Using `<header>`, `<nav>`, `<main>`, `<footer>` for structuring, `<button>` for buttons, `<h1>` to `<h6>` for headings.

```
<header>
  <nav>
    <!-- Navigation links -->
  </nav>
</header>

<main>
  <article>
    <section>
      <h1>Article Title</h1>
      <p>Article content...</p>
    </section>
  </article>
</main>

<footer>
  <p>Contact information</p>
</footer>
```

Basic Accessibility Features – Part 3

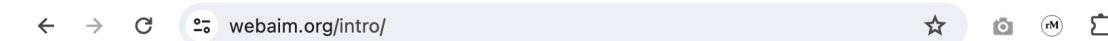
Accessible Forms:

Purpose: Make forms easy to complete and understand.

Examples: Label elements explicitly linked to input fields, clear instructions, error messages.

```
<form>
  <fieldset>
    <legend>Payment Information</legend>
    <label for="card">Credit Card Number:</label>
    <input type="text" id="card" name="card">
    <!-- Additional fields -->
  </fieldset>
</form>
```

Principles of Accessible Design



Principles of Accessible Design

Below you will find a list of some key principles of accessible design. Most can be implemented without compromising the overall look and feel of your website.

Provide equivalent alternative text

Alternative text provides a textual alternative to non-text content in web pages. It is especially helpful for people who are blind and rely on a screen reader to have the content of the website read to them.

Create logical document structure

Headings, lists, and other structural elements provide meaning and structure to web pages. They can also facilitate keyboard navigation within the page.

Provide headers for data tables

Data tables need row and column header cells (`<th scope="row">` and `<th scope="col">`) to programmatically associate them with their corresponding data cells (`<td>`), making it easier for screen reader users to navigate and understand the table.

Ensure users can complete and submit all forms

Every form element (text field, checkbox, dropdown list, etc.) needs a programmatically-associated label. Users must be able to [submit the form and recover from errors](#).

Write links that make sense out of context

Every link should make sense when read by itself. Screen reader users may choose to read only the links on a web page. Certain phrases like "click here" and "more" must be avoided.

Caption and/or provide transcripts for media

Videos and live audio must have captions and a transcript. With archived audio, a transcription may be enough.

<https://webaim.org/intro/>

Basic Accessibility Features – Part 4

Color Contrast:

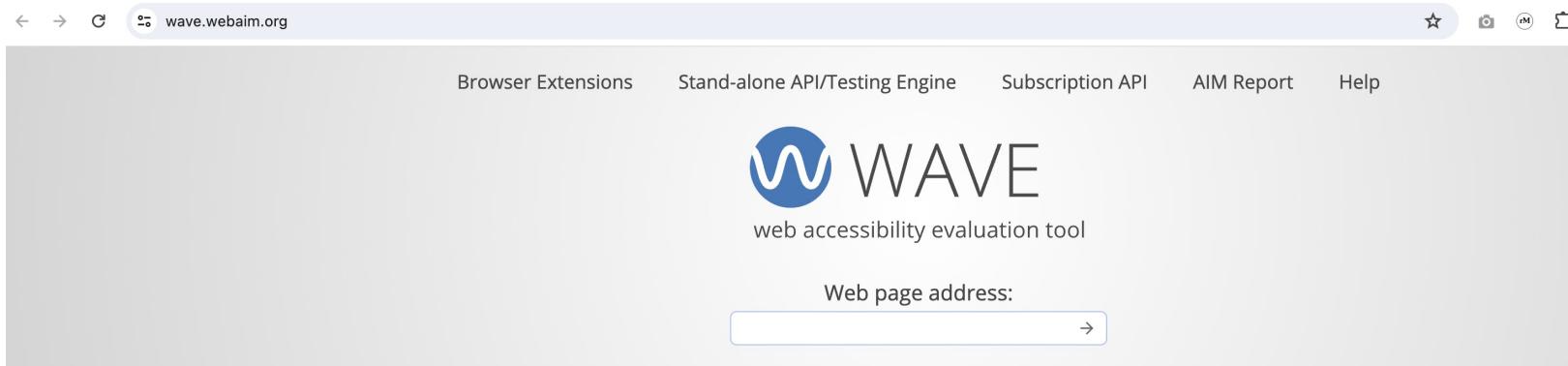
Purpose: Ensure that text stands out against the background.

Examples: High contrast color schemes, avoiding color reliance for conveying information.

The screenshot shows the homepage of the WCAG Color Contrast Checker. The title 'WCAG Color Contrast Checker' is at the top center. Below it is a sub-instruction: 'Enter your color combinations to check whether they pass conformance. To check colors straight from your browser, download our free chrome extension.' A 'Add to Chrome for Free' button is present. The main section is titled 'How it Works' with a sub-instruction: 'Please select a foreground color and a background color. You can enter hex color codes or use the color selector tool (aka eye dropper in the color input element). The "WCAG Compliance Information" chart will tell you if the selected colors pass conformance.' Below this are color selection tools for 'Foreground Color' (#503f3f) and 'Background Color' (#1d1b1b), and a 'Contrast Ratio' of '1.73:1'. The 'WCAG Compliance Results' table shows failing results for 'Small Text', 'Large Text', and 'UI Components' under both AA and AAA columns. To the right, there are visual samples of 'SMALL sample text: 14pt (18.5px)' and 'LARGE sample text: 18pt (24px)'.

ELEMENT TYPE	AA	AAA
Small Text	✗ Fail	✗ Fail
Large Text	✗ Fail	✗ Fail
UI Components	✗ Fail	✗ Fail

Accessibility Testing Tools



WAVE Web Accessibility Evaluation Tools

WAVE® is a suite of evaluation tools that helps authors make their web content more accessible to individuals with disabilities. WAVE can identify many accessibility and Web Content Accessibility Guideline (WCAG) errors, but also facilitates human evaluation of web content. Our philosophy is to focus on issues that we know impact end users, facilitate human evaluation, and to educate about web accessibility.

WAVE Browser Extensions

You can use the online WAVE tool by entering a web page address (URL) in the field above. [WAVE Chrome, Firefox, and Edge browser extensions](#) are available for testing accessibility directly within your web browser - handy for checking password protected, locally stored, or highly dynamic pages.

WAVE API and Testing Engine

The [WAVE subscription API](#) and [Stand-alone WAVE API and Testing Engine](#) are powerful tools for easily collecting accessibility test data on many pages. The stand-alone API and Testing Engine can be integrated into your infrastructure for testing intranet, non-public, and secure pages, including in continuous integration processes.

Accessibility IMpact (AIM) Report

Your [Accessibility IMpact \(AIM\) assessment report](#) provides detailed WAVE test data, your site's AIM score (a measure of end user impact compared to web pages generally and as determined by human testers), and expert manual test results to give you insights into the accessibility of your web site for users with disabilities.

Accessibility Testing Tools

The screenshot shows the W3C Web Accessibility Initiative (WAI) website at <https://www.w3.org/WAI/test-evaluate/tools/list/>. The page title is "Web Accessibility Evaluation Tools List". The main content area displays a list of 20 tools, with the first tool, "Accessible Web RAMP", highlighted. The "Filters" sidebar on the left allows users to search for tools based on purpose (Automated testing, Manual testing, Simulated user experience) and product evaluated (Website, Document, Source code, Other, Mobile application).

Filters

Purpose

- Automated testing (20)
- Manual testing (7)
- Simulated user experience (3)

Product evaluated

- Website (18)
- Document (5)
- Source code (5)
- Other (2)
- Mobile application (1)

Tools list

Showing 20 tools

Accessible Web RAMP by Accessible Web Last updated: Feb 2024

Description	Features
RAMP helps teams discover, understand, and remediate web accessibility issues. Our suite of tools includes a website monitoring platform, free browser extension for	<ul style="list-style-type: none">Website, Source codeBrowser plugin, Online toolFree, Limited free functionality, Time-limited trial,

<https://www.w3.org/WAI/test-evaluate/tools/list/>

Accessibility & Bootstrap 5

The screenshot shows a web browser displaying the Bootstrap 5 documentation at <https://getbootstrap.com/docs/5.0/getting-started/accessibility/>. The page has a purple header with the Bootstrap logo and navigation links for Home, Docs, Examples, Icons, Themes, and Blog. A prominent message "There's a newer version of Bootstrap!" is displayed above the main content. The main title is "Accessibility". Below it is a brief overview: "A brief overview of Bootstrap's features and limitations for the creation of accessible content." To the left is a sidebar with a "Getting started" section containing links to Introduction, Download, Contents, Browsers & devices, JavaScript, Build tools, Webpack, Parcel, Accessibility (which is currently selected), RFS, and RTL. Other sections include Customize, Layout, Content, Forms, Components, Helpers, Utilities, Extend, and About. On the right, there's a "On this page" sidebar with links to Overview and limitations, Structural markup, Interactive components, Color contrast, Visually hidden content, Reduced motion, and Additional resources. A "View on GitHub" button is also present. A sidebar on the right contains a "Design and Development tips in your inbox. Every weekday." section with icons for email, calendar, and social media, and a note "ads via Carbon". At the bottom, there's a "Structural markup" section and a footer note about markup structures.

<https://getbootstrap.com/docs/5.0/getting-started/accessibility/>

Australian Requirements

WCAG Compliance: Websites must adhere to the Web Content Accessibility Guidelines (WCAG) to be considered accessible.

DDA Requirements: The Disability Discrimination Act 1992 (DDA) mandates that web content should be accessible to people with disabilities to prevent discrimination.

National Transition Strategy: Specifies that all Australian government websites must meet at least WCAG 2.0 Level AA standards.

Current Standard: WCAG 2.0 is the current Australian standard, with WCAG 2.1 and the new WCAG 2.2 offering further guidance for mobile and cognitive disabilities.

Responsive Design

Responsive Design

Responsive design is crucial for a variety of reasons, many of which center on delivering a seamless and effective user experience across different devices and screen sizes.

User Experience

- **Adaptability:** Users access websites from a multitude of devices with varying screen sizes. A responsive design ensures a consistent look and feel.
- **Ease of Use:** Navigation and site interaction are straightforward regardless of whether a user is on a desktop computer, tablet, or mobile phone.

Increased Reach

- **Mobile Users:** With the increasing number of people using mobile devices to access the internet, having a mobile-responsive site can significantly widen your audience.
- **Social Sharing:** Content shared on social media is often viewed on mobile devices. Responsive design ensures that your pages look good wherever they're viewed, making users more likely to share.

The Viewport Meta Tag

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

The viewport meta tag is a critical element in designing responsive web pages.

Purpose: The viewport meta tag tells browsers how to adjust the page's dimensions and scaling to suit the display device.

Control Layout on Mobile: It's essential for controlling the layout on mobile browsers, which would otherwise scale websites down to fit the screen.

Default Scaling: Without the viewport tag, mobile devices typically render pages at typical desktop widths, then scale them down, making text small and unreadable.

Viewport Width: The width=device-width setting ensures the width of the page matches the width of the device's screen.

Initial Scale: The initial-scale=1 setting establishes the initial zoom level when the page is first loaded by the browser.

Responsive Design: The viewport tag is a foundational element for creating a responsive design, allowing the content to fluidly adjust to different screen sizes.

Media Queries

- Media queries are CSS techniques used for creating responsive designs
- Adapt the layout to different screens (e.g., desktops, tablets, phones).

Common Media Features

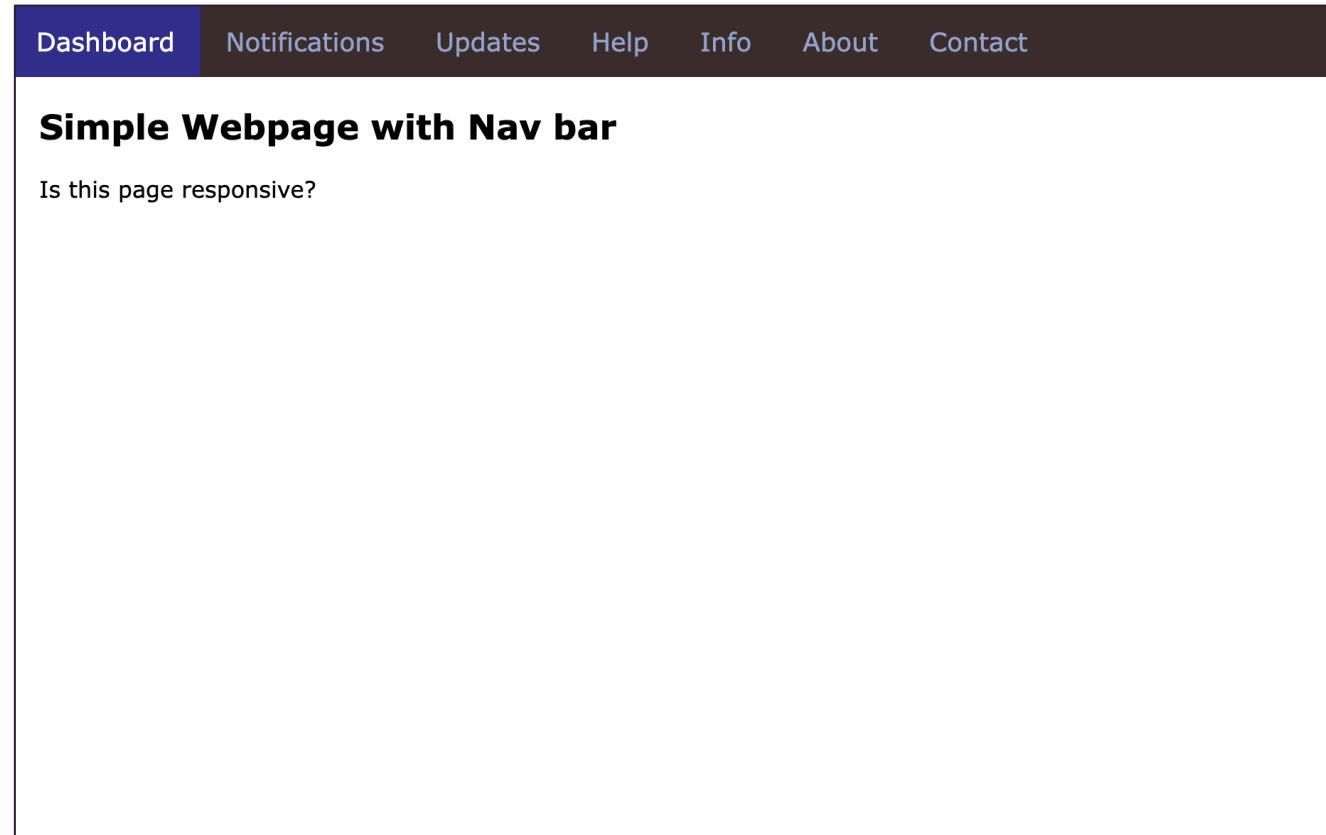
- max-width / min-width
- max-height / min-height
- orientation: landscape/portrait

```
@media media-type and (media-feature-rule) {  
    /* CSS rules go here */  
}
```

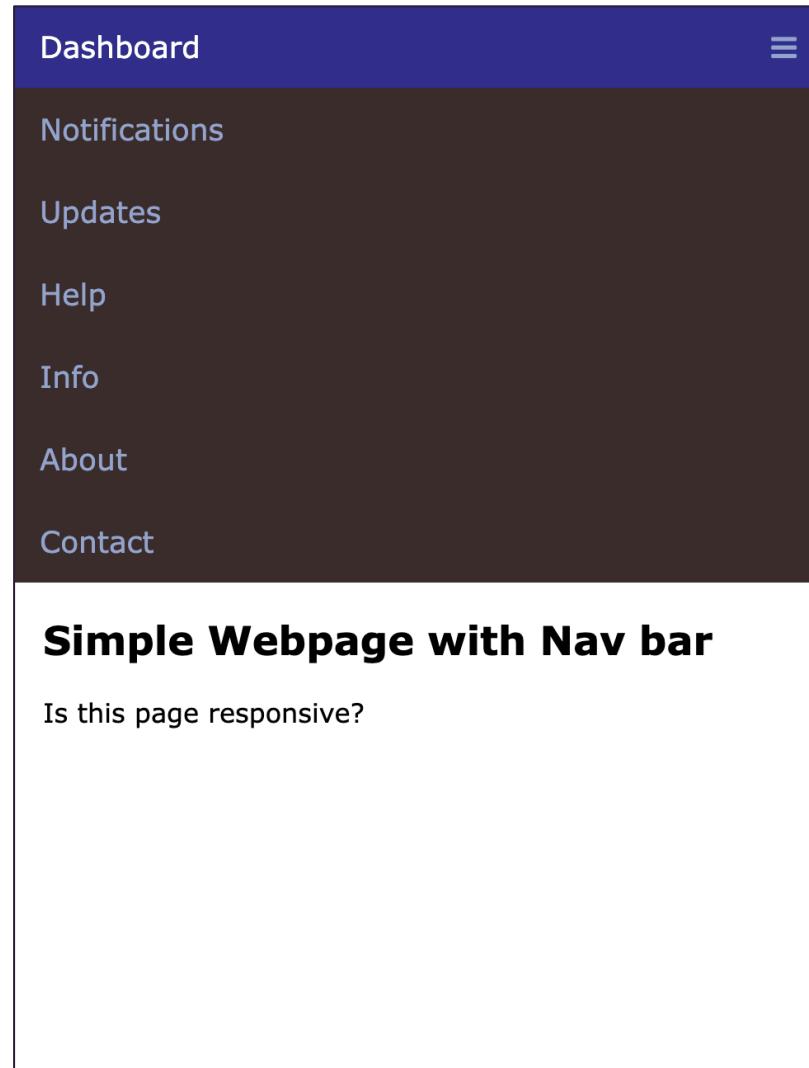
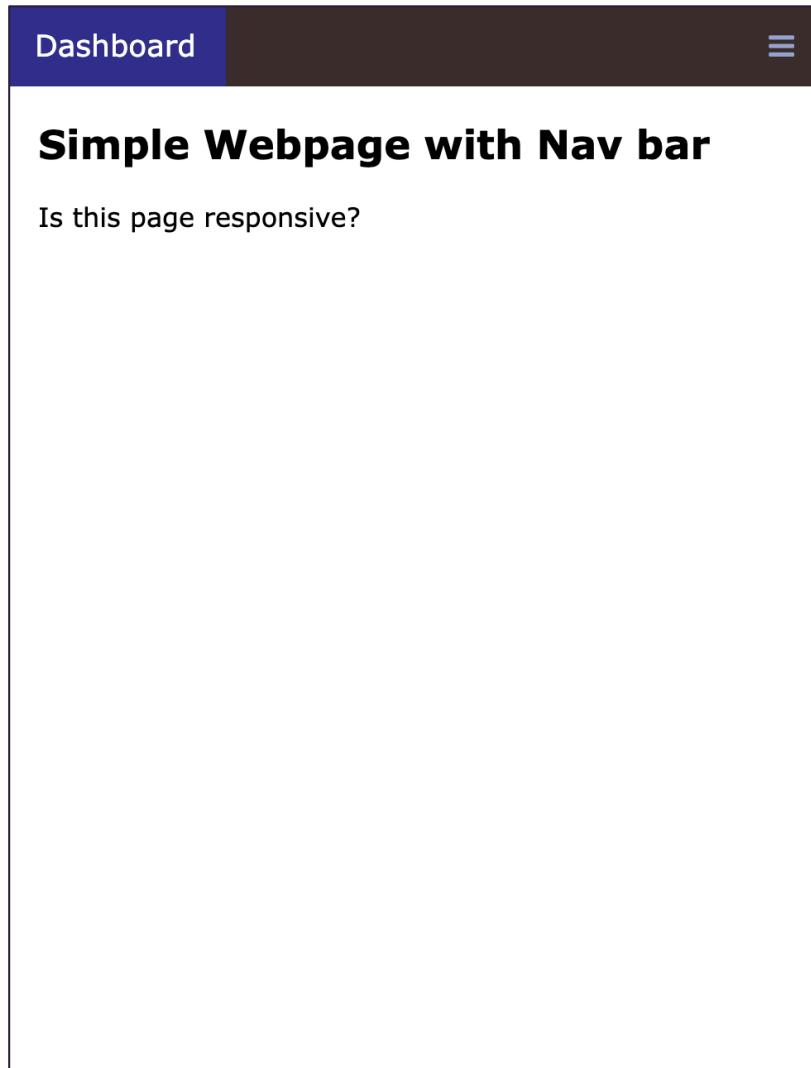
Media Queries – Device Breakpoints

```
1  /* Extra small screens */
2  @media only screen and (max-width: 600px) {
3
4  }
5
6  /* Tablets and Large Phones */
7  @media only screen and (min-width: 600px) {
8
9  }
10
11 /* Tablets in Landscape */
12 @media only screen and (min-width: 768px) {
13
14 }
15
16 /* Large Laptop screens */
17 @media only screen and (min-width: 992px) {
18
19 }
20
21 /* Extra large screens – Laptops and Desktops */
22 @media only screen and (min-width: 1200px) {
23
24 }
```

Creating a Responsive Menu



Creating a Responsive Menu



Responsive Design with Bootstrap

The screenshot shows a web browser displaying the Bootstrap documentation at getbootstrap.com/docs/5.0/layout/breakpoints/. The page has a purple header with the Bootstrap logo and navigation links for Home, Docs, Examples, Icons, Themes, and Blog. A search bar and a 'Download' button are also present. The main content area features a large heading 'Breakpoints' with a subtext explaining they are customizable widths for responsive layouts. To the left is a sidebar with a navigation menu. On the right, there's a sidebar titled 'On this page' listing various Bootstrap concepts.

There's a newer version of Bootstrap!

B Home Docs Examples Icons Themes Blog

Search docs... Ctrl + /

Bootstrap v5.0

> Getting started
> Customize
_LAYOUT
Breakpoints
Containers
Grid
Columns
Gutters
Utilities
Z-index
> Content
> Forms
> Components
> Helpers
> Utilities
> Extend
> About
Migration

Breakpoints

View on GitHub

Breakpoints are customizable widths that determine how your responsive layout behaves across device or viewport sizes in Bootstrap.

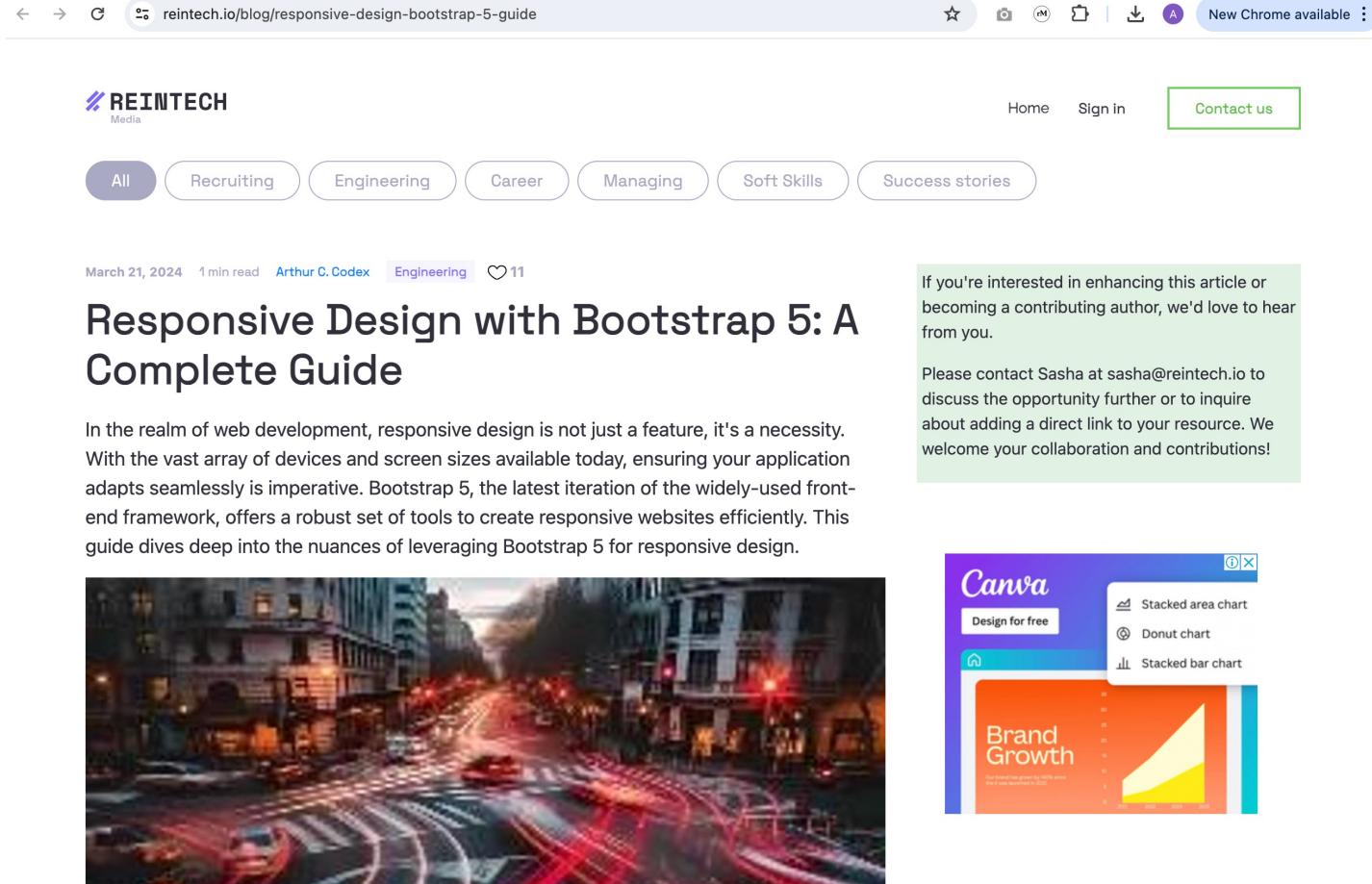
Design and Development tips in your inbox. Every weekday.
ads via Carbon

Core concepts

- **Breakpoints are the building blocks of responsive design.** Use them to control when your layout can be adapted at a particular viewport or device size.
- **Use media queries to architect your CSS by breakpoint.** Media queries are a feature of CSS that allow you to conditionally apply styles based on a set of browser and operating system parameters. We most commonly use `min-width` in our media queries.
- **Mobile first, responsive design is the goal.** Bootstrap's CSS aims to apply the bare minimum of styles to make a layout work at the smallest breakpoint, and then layers on styles to adjust that design for larger devices. This optimizes your CSS, improves rendering time, and provides a great experience for your visitors.

<https://getbootstrap.com/docs/5.0/layout/breakpoints/>

Responsive Design with Bootstrap



The screenshot shows a web browser window with the URL reintech.io/blog/responsive-design-bootstrap-5-guide. The page is from the REINTECH Media blog. The header includes the REINTECH logo, navigation links for Home, Sign in, and Contact us, and category filters for All, Recruiting, Engineering, Career, Managing, Soft Skills, and Success stories. The main content features a post by Arthur C. Codex published on March 21, 2024, with a 1-minute read time. The title of the post is "Responsive Design with Bootstrap 5: A Complete Guide". The post discusses the importance of responsive design in web development and how Bootstrap 5 facilitates it. It includes a large image of a city street at night with blurred lights from moving vehicles. To the right of the post, there is a sidebar with a call to action for enhancing the article or becoming a contributing author, and information about contacting Sasha at sasha@reintech.io.

If you're interested in enhancing this article or becoming a contributing author, we'd love to hear from you.

Please contact Sasha at sasha@reintech.io to discuss the opportunity further or to inquire about adding a direct link to your resource. We welcome your collaboration and contributions!



https://reintech.io/blog/responsive-design-bootstrap-5-guide#google_vignette

Progressive Web Applications (PWA's)

What is a PWA?

A Progressive Web Application (PWA) is a web application that provides a native app-like experience to users, using modern web technologies such as HTML, CSS, and JavaScript. PWAs are designed to work seamlessly across various devices, including desktops, laptops, mobile phones, and tablets.

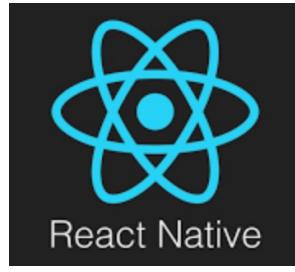
Key Characteristics:

- **Responsive:** PWAs adapt to different screen sizes and devices, providing an optimal user experience.
- **Fast and Reliable:** PWAs use modern web technologies to provide fast and reliable performance, even on low-end devices.
- **Offline Access:** PWAs can function offline or with a slow internet connection, allowing users to access content and functionality even when network connectivity is limited.
- **Installable:** Users can install PWAs on their devices, creating a native app-like experience.
- **Secure:** PWAs use HTTPS, ensuring that all data transmitted between the app and the server is encrypted and secure.
- **Engaging:** PWAs provide a native app-like experience, with features such as push notifications, home screen installation, and more.

PWA is an alternate to building a Native App for each platform



Flutter



React Native

HTML

vs

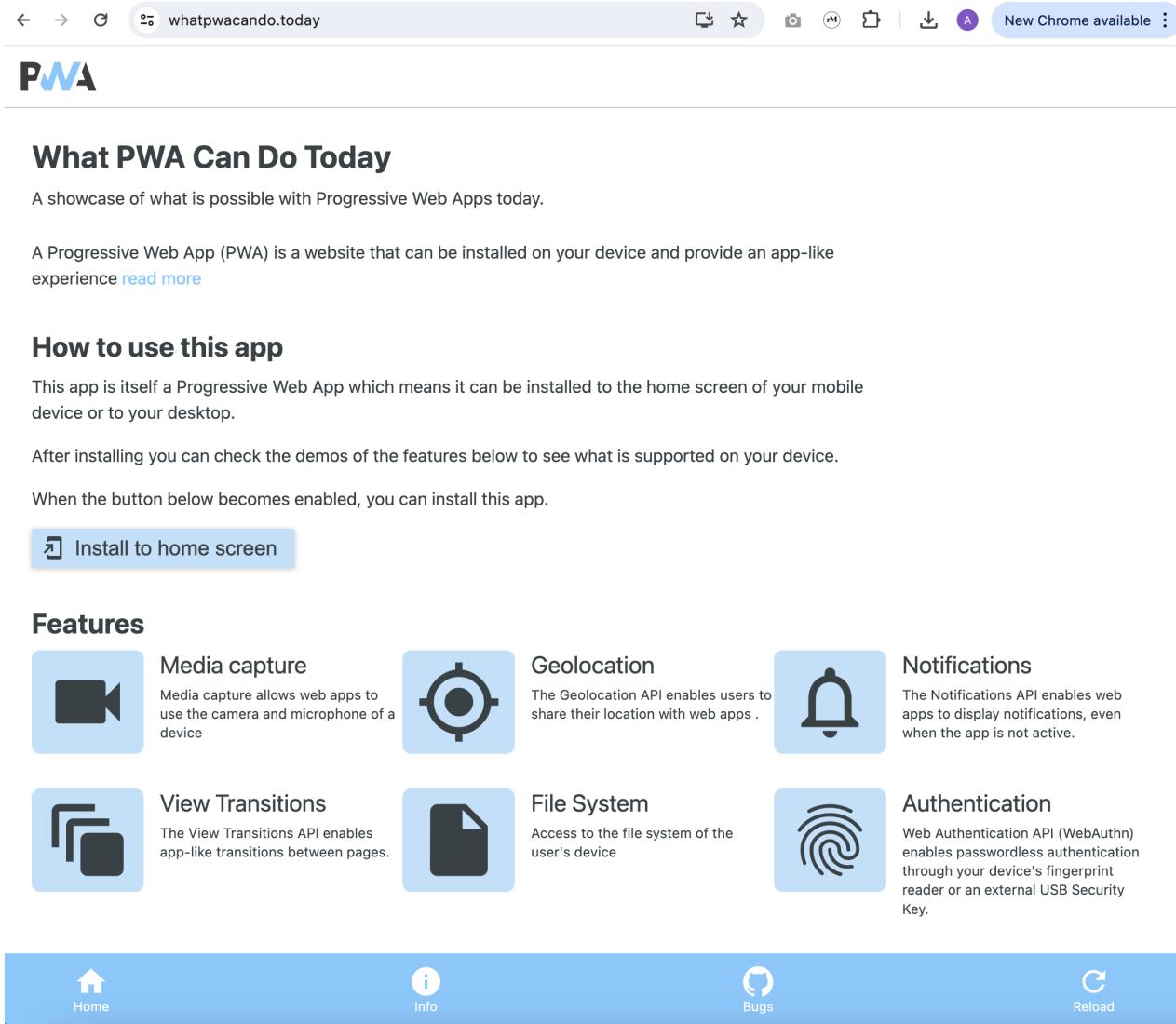
JS

Android The Android logo, which is a green silhouette of a robot head.

CSS



What can PWA do Today?



A screenshot of a Progressive Web App (PWA) website titled "What PWA Can Do Today". The page includes a header with browser controls and a "New Chrome available" notification. Below the header, there's a "PWA" logo and a section titled "What PWA Can Do Today" with a sub-section about showcasing what is possible with PWA today. It mentions that a PWA is a website that can be installed on your device and provide an app-like experience, with a "read more" link. A "How to use this app" section explains that it's itself a PWA and can be installed to the home screen. It also states that after installing, users can check demos of supported features. A prominent blue button labeled "Install to home screen" is visible. The "Features" section lists six technologies with icons: Media capture (camera), Geolocation (location pin), Notifications (bell), View Transitions (document with arrows), File System (file icon), and Authentication (fingerprint). At the bottom, there's a navigation bar with icons for Home, Info, Bugs, and Reload.

- You can install this app on your Mobile Phone's homepage.
- Then test to check which features are available

What can PWA do Today?

The image shows two web pages side-by-side. The left page is titled "What PWA Can Do Today" and lists various Progressive Web App features with icons: Media capture, Geolocation, Notifications, View Transitions, File System, Authentication, Protocol Handling, and File Handling API. It also includes a "read more" link for progressive web apps. The right page is titled "Can I use" and shows a chart of browser support for "Web Notifications". The chart compares support across multiple browsers including Chrome, Edge, Safari, Firefox, Opera, IE, Chrome for Android, Safari on iOS, Samsung Internet, Opera Mini, Opera Mobile, UC Browser for Android, Android Browser, Firefox for Android, QQ Browser, Baidu Browser, and KaiOS Browser. The chart uses a color-coded system where green indicates full support and red indicates partial or no support. Global usage statistics are also provided.

12:12 5G

PWA

What PWA Can Do Today

A showcase of what is possible with Progressive Web Apps today.

A Progressive Web App (PWA) is a website that can be installed on your device and provide an app-like experience [read more](#)

Check the demos of the features below to see what is supported on your device.

Features

Media capture	Geolocation	Notifications	View Transitions
File System	Authentication	Protocol Handling	File Handling API
Home	Info	Bugs	Reload

April 7, 2024 - 8 new features

Compare browsers About

Can I use _____

? Settings

Feature: Web Notifications

Web Notifications

Method of alerting the user outside of a web page by displaying notifications (that do not require interaction by the user).

Usage % of all users

Global 77.86% + 13.45% = 91.3%
unprefixed: 77.86% + 11.42% = 89.27%

Browser	Version	Support
Chrome	4	Partial
Edge	5-21	Partial
Safari	12-13	Partial
Firefox	3.1-5.1	Partial
Opera	2-21	Partial
IE	10-24	Partial
Chrome for Android	3.2-16.3	Partial
Safari on iOS	4	Partial
Samsung Internet	16.4-17.3	Partial
Opera Mini	5-23	Partial
Opera Mobile	12-12.1	Partial
UC Browser for Android	2.1-4.3	Partial
Android Browser	4.4-4.4.4	Partial
Firefox for Android	12.0	Partial
QQ Browser	12.4	Partial
Baidu Browser	14.9	Partial
KaiOS Browser	2.5	Partial
Others	12.5	Partial

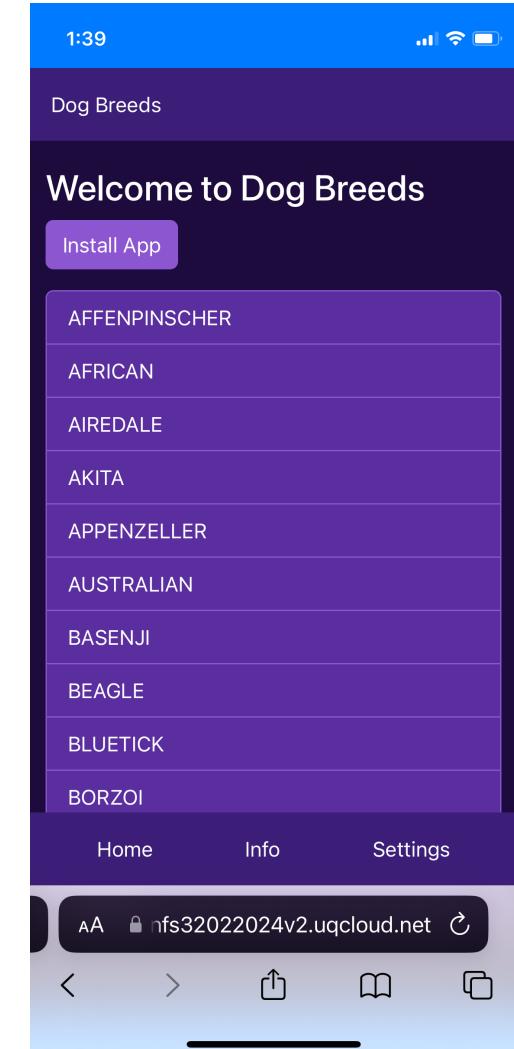
Notes Test on a real browser Known issues (3) Resources (9) Feedback

¹ Supports notifications via the [Push API](#) but not the Web Notifications API.
² Chrome for Android requires the call to be made with a [service worker registration](#)
³ Requires website to first be [added to the Home Screen](#)

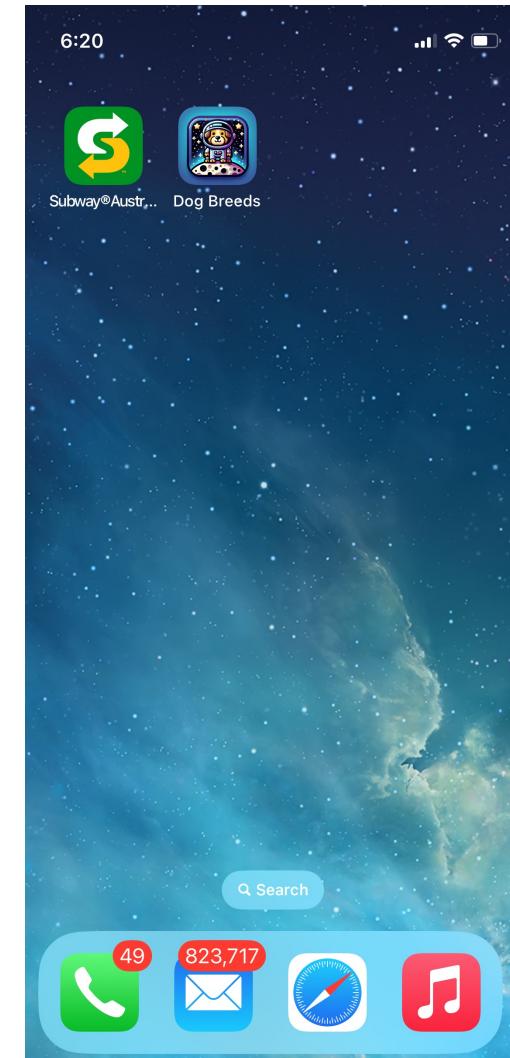
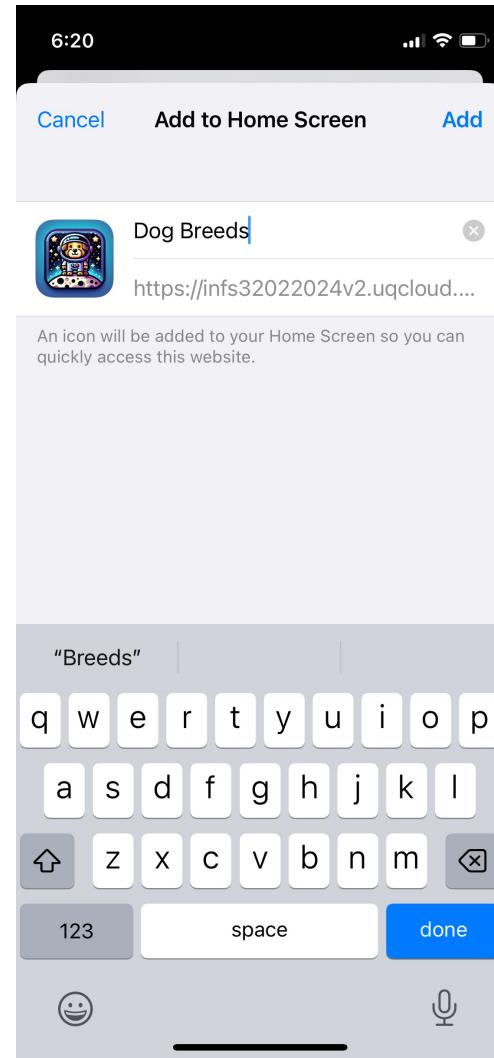
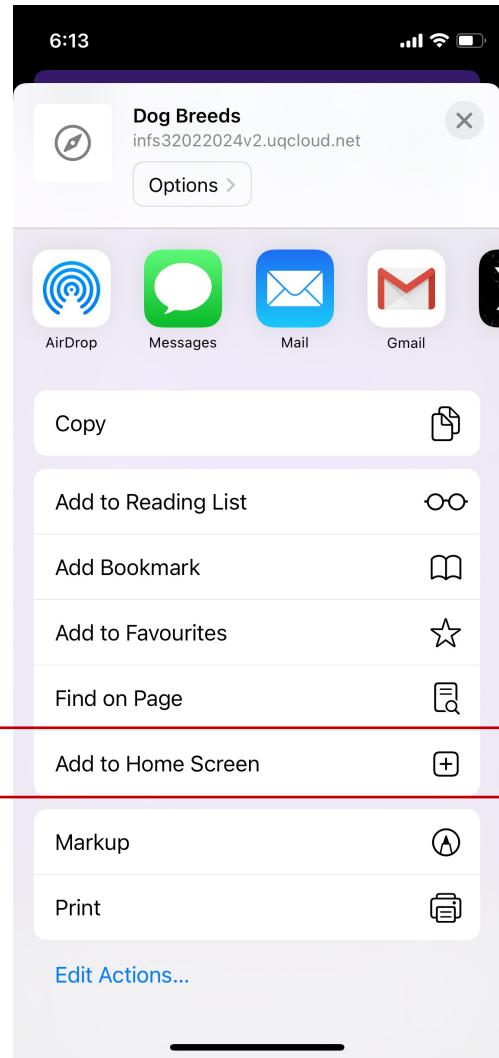
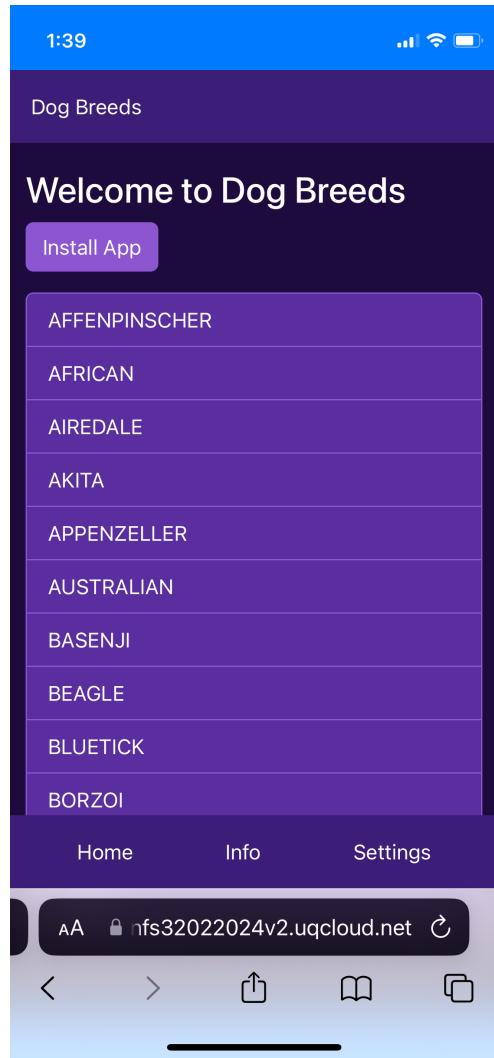
<https://whatpwacando.today/>

<https://caniuse.com/notifications>

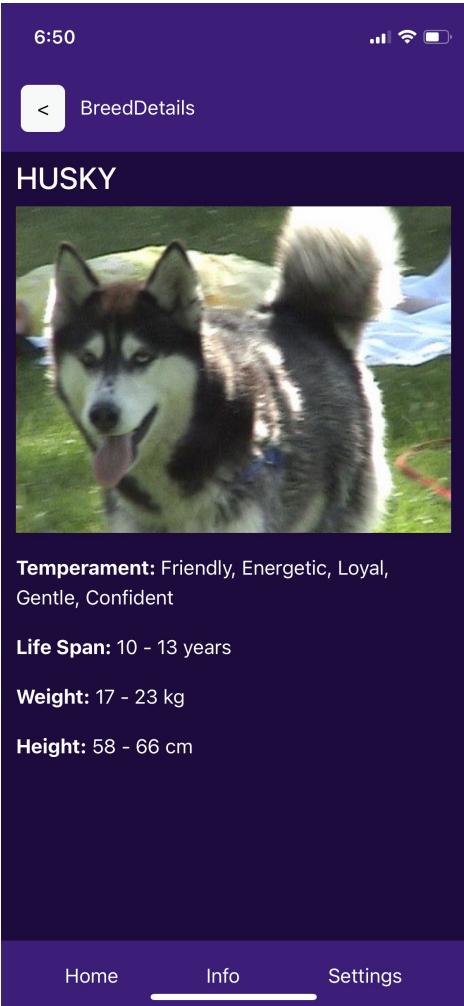
The Dog Breeds PWA



The Dog Breeds PWA



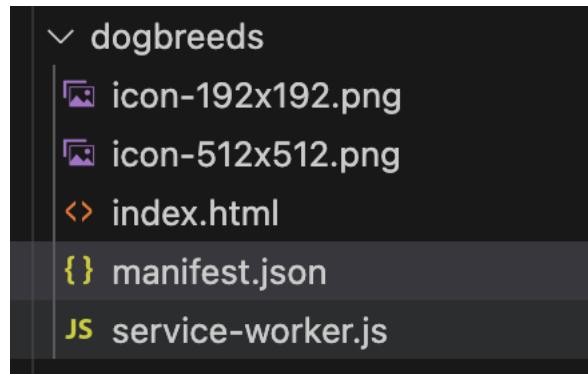
The Dog Breeds PWA



The Dog Breeds PWA

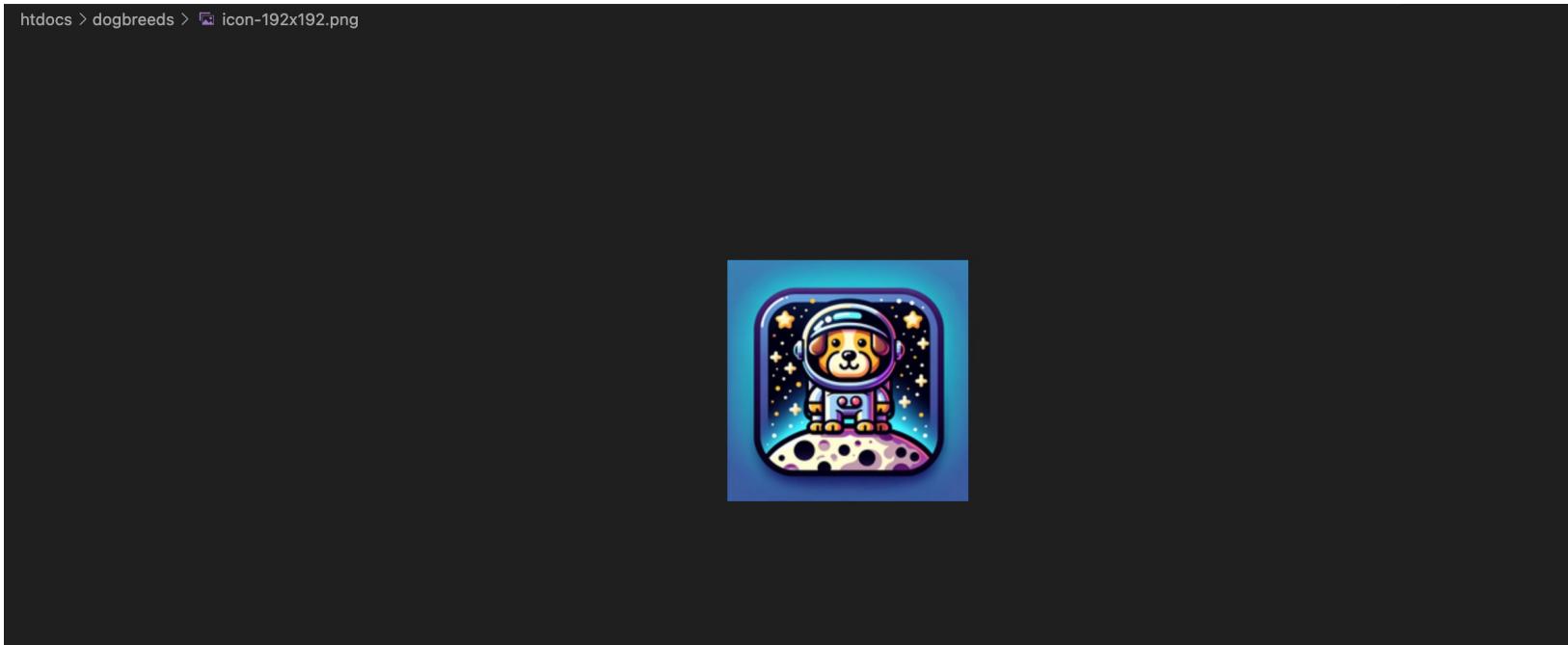
```
htdocs > dogbreeds > {} manifest.json > ...
1  {
2    "name": "Dog Breeds",
3    "short_name": "Dog Breeds",
4    "start_url": "/dogbreeds/index.html",
5    "display": "standalone",
6    "background_color": "#1c0b3c",
7    "theme_color": "#3c1e78",
8    "icons": [
9      {
10        "src": "/dogbreeds/icon-192x192.png",
11        "sizes": "192x192",
12        "type": "image/png"
13      },
14      {
15        "src": "/dogbreeds/icon-512x512.png",
16        "sizes": "512x512",
17        "type": "image/png"
18      }
19    ]
20  }
21
```

The Dog Breeds PWA



```
htdocs > dogbreeds > {} manifest.json > ...
1  {
2    "name": "Dog Breeds",
3    "short_name": "Dog Breeds",
4    "start_url": "/dogbreeds/index.html",
5    "display": "standalone",
6    "background_color": "#1c0b3c",
7    "theme_color": "#3c1e78",
8    "icons": [
9      {
10        "src": "/dogbreeds/icon-192x192.png",
11        "sizes": "192x192",
12        "type": "image/png"
13      },
14      {
15        "src": "/dogbreeds/icon-512x512.png",
16        "sizes": "512x512",
17        "type": "image/png"
18      }
19    ]
20  }
21
```

The Dog Breeds PWA



Created with DALLE 3

The Dog Breeds PWA

```
htdocs > dogbreeds > index.html > html > head
47  <header class="fixed-top text-white p-3 mb-2">
48  |   <button id="backButton" class="btn btn-light d-none me-2">&lt;</button>
49  |   <span id="pageTitle" class="mt-2">Dog Breeds</span>
50  </header>
51
52  <main class="content container mt-3">
53  |   <div id="home">
54  |   |   <h1>Welcome to Dog Breeds</h1>
55  |   |   <button id="installButton" class="btn btn-primary">Install App</button>
56  |   |   <ul id="breedList" class="list-group mt-3"></ul>
57  |   </div>
58  |   <div id="info" class="d-none">
59  |   |   <h2>Info</h2>
60  |   |   <p>This is the info page.</p>
61  |   </div>
62  |   <div id="settings" class="d-none">
63  |   |   <h2>Settings</h2>
64  |   |   <p>This is the settings page.</p>
65  |   </div>
66  |   <div id="breedDetails" class="d-none"></div>
67  </main>
68
69  <footer class="fixed-bottom p-3">
70  |   <div class="d-flex justify-content-around">
71  |   |   <a href="#" class="text-decoration-none text-center text-white" data-page="home">
72  |   |   |   <i class="bi bi-house-door d-block fs-4"></i>
73  |   |   |   Home
74  |   |   </a>
75  |   |   <a href="#" class="text-decoration-none text-center text-white" data-page="info">
76  |   |   |   <i class="bi bi-info-circle d-block fs-4"></i>
77  |   |   |   Info
78  |   |   </a>
79  |   |   <a href="#" class="text-decoration-none text-center text-white" data-page="settings">
80  |   |   |   <i class="bi bi-gear d-block fs-4"></i>
81  |   |   |   Settings
82  |   |   </a>
83  |   </div>
84  </footer>
```

The Dog Breeds PWA

```
htdocs > dogbreeds > index.html > head  
..  
87  <script>  
88  const backButton = document.getElementById('backButton');  
89  const pageTitle = document.getElementById('pageTitle');  
90  const installButton = document.getElementById('installButton');  
91  const breedList = document.getElementById('breedList');  
92  const breedDetails = document.getElementById('breedDetails');  
93  const pages = document.querySelectorAll('main > div');  
94  
95  let deferredPrompt;  
96  
97  window.addEventListener('beforeinstallprompt', (e) => {  
98    e.preventDefault();  
99    deferredPrompt = e;  
100   installButton.classList.remove('d-none');  
101 });  
102  
103 installButton.addEventListener('click', async () => {  
104   if (deferredPrompt !== null) {  
105     deferredPrompt.prompt();  
106     const { outcome } = await deferredPrompt.userChoice;  
107     if (outcome === 'accepted') {  
108       installButton.classList.add('d-none');  
109     }  
110     deferredPrompt = null;  
111   }  
112 });  
113  
114 function navigateTo(pageId) {  
115   pages.forEach(page => {  
116     if (page.id === pageId) {  
117       page.classList.remove('d-none');  
118       pageTitle.textContent = pageId.charAt(0).toUpperCase() + pageId.slice(1);  
119       backButton.classList.remove('d-none');  
120     } else {  
121       page.classList.add('d-none');  
122     }  
123   });
124 }
125
```

```
126  backButton.addEventListener('click', () => {  
127    navigateTo('home');  
128    backButton.classList.add('d-none');  
129    pageTitle.textContent = 'Dog Breeds';  
130  });
131  
132  document.querySelectorAll('footer a').forEach(link => {  
133    link.addEventListener('click', (e) => {  
134      e.preventDefault();  
135      const pageId = link.getAttribute('data-page');  
136      navigateTo(pageId);  
137    });
138  });
139
```

The Dog Breeds PWA

```
htdocs > dogbreeds > index.html > head
141  async function fetchBreeds() {
142    const response = await fetch('https://dog.ceo/api/breeds/list/all');
143    const data = await response.json();
144    const breeds = Object.keys(data.message);
145    breeds.forEach(breed => {
146      const li = document.createElement('li');
147      li.classList.add('list-group-item');
148      li.textContent = breed.toUpperCase();
149      li.addEventListener('click', () => fetchBreedDetails(breed));
150      breedList.appendChild(li);
151    });
152  }
153
154  async function fetchBreedDetails(breed) {
155    const response = await fetch(`https://dog.ceo/api/breed/${breed}/images/random`);
156    const data = await response.json();
157    const image = data.message;
158
159    const breedResponse = await fetch(`https://api.thedogapi.com/v1/breeds/search?q=${breed}`);
160    const breedData = await breedResponse.json();
161    const breedInfo = breedData[0];
162
163    let temperament = '';
164    let lifeSpan = '';
165    let weight = '';
166    let height = '';
167
168    if (breedInfo) {
169      temperament = breedInfo.temperament || '';
170      lifeSpan = breedInfo.life_span || '';
171      weight = breedInfo.weight.metric || '';
172      height = breedInfo.height.metric || '';
173    }
174
175    breedDetails.innerHTML =
176      `

## ${breed.toUpperCase()}


177      
178      <p><strong>Temperament:</strong> ${temperament}</p>
179      <p><strong>Life Span:</strong> ${lifeSpan}</p>
180
181    let temperament = '';
182    let lifeSpan = '';
183    let weight = '';
184    let height = '';
185
186    if (breedInfo) {
187      temperament = breedInfo.temperament || '';
188      lifeSpan = breedInfo.life_span || '';
189      weight = breedInfo.weight.metric || '';
190      height = breedInfo.height.metric || '';
191
192      breedDetails.innerHTML =
193        `

## ${breed.toUpperCase()}


194        
195        <p><strong>Temperament:</strong> ${temperament}</p>
196        <p><strong>Life Span:</strong> ${lifeSpan}</p>
197
198    </body>
199  </html>
```

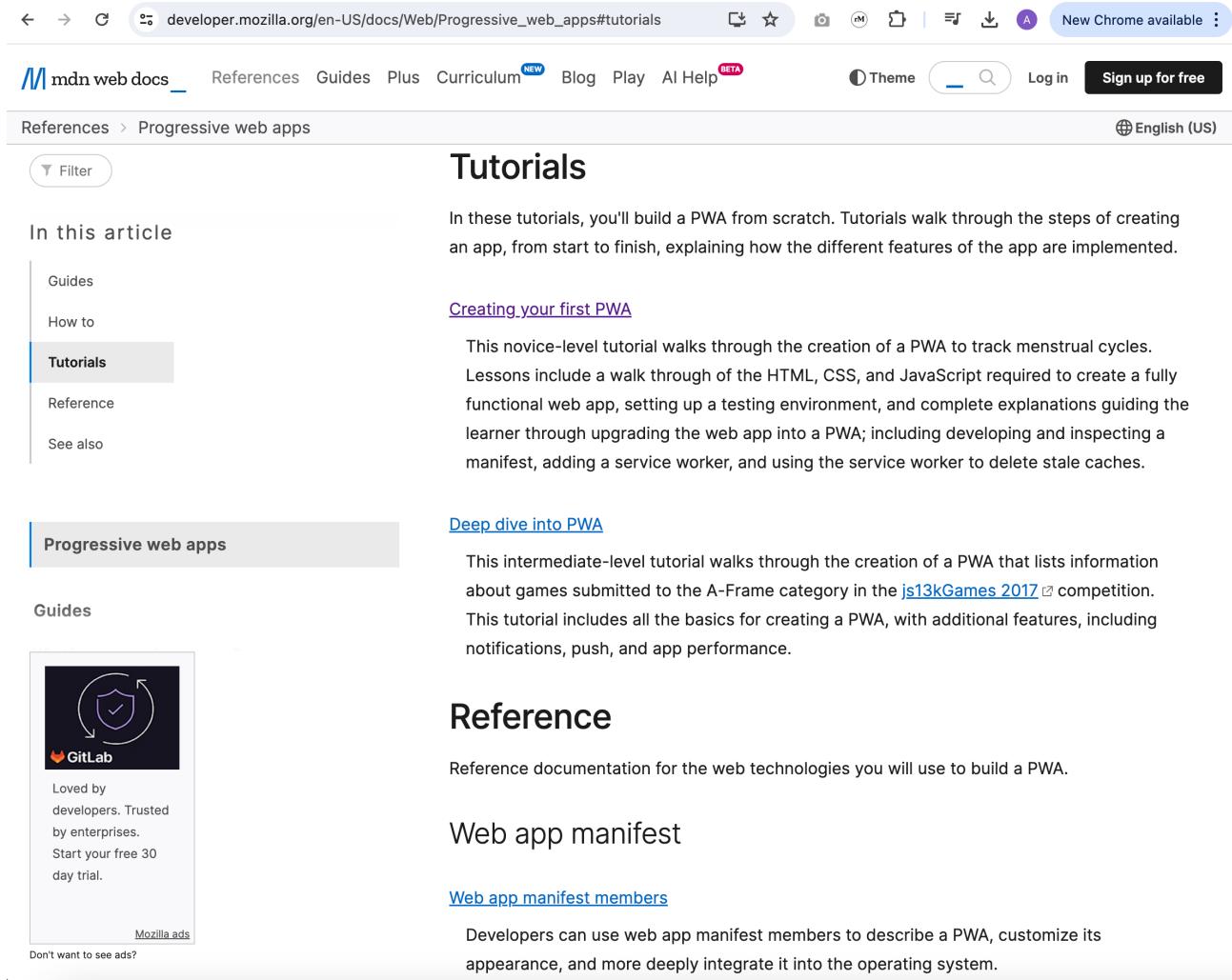
What is a Service Worker?

A Service Worker is a script that runs in the background of a web browser, separate from the web page, and acts as a proxy between the web application, the browser, and the network.

It is a key component of Progressive Web Apps (PWAs) and enables features such as

- offline functionality
- background syncing
- push notifications

Progressive Web Apps – MDN Tutorials



The screenshot shows the MDN Progressive Web Apps Tutorials page. At the top, there's a navigation bar with links for developer.mozilla.org, mdn web docs, References, Guides, Plus, Curriculum (NEW), Blog, Play, AI Help (BETA), Theme, Log in, and Sign up for free. Below the navigation is a breadcrumb trail: References > Progressive web apps, with English (US) selected. A sidebar on the left titled "In this article" has a "Tutorials" section highlighted. The main content area is titled "Tutorials" and describes building a PWA from scratch. It includes sections for "Creating your first PWA" (a novice-level tutorial for tracking menstrual cycles) and "Deep dive into PWA" (an intermediate-level tutorial for creating a PWA for a game competition). A "Reference" section provides documentation for web technologies. At the bottom, there's a Mozilla ads banner for GitLab.

developer.mozilla.org/en-US/docs/Web/Progressive_web_apps#tutorials

mdn web docs References Guides Plus Curriculum NEW Blog Play AI Help BETA Theme Log in Sign up for free

References > Progressive web apps English (US)

In this article

Guides How to Tutorials Reference See also

Progressive web apps

Guides

Loved by developers. Trusted by enterprises. Start your free 30 day trial.

GitLab Mozilla ads

Don't want to see ads?

Tutorials

In these tutorials, you'll build a PWA from scratch. Tutorials walk through the steps of creating an app, from start to finish, explaining how the different features of the app are implemented.

[Creating your first PWA](#)

This novice-level tutorial walks through the creation of a PWA to track menstrual cycles. Lessons include a walk through of the HTML, CSS, and JavaScript required to create a fully functional web app, setting up a testing environment, and complete explanations guiding the learner through upgrading the web app into a PWA; including developing and inspecting a manifest, adding a service worker, and using the service worker to delete stale caches.

[Deep dive into PWA](#)

This intermediate-level tutorial walks through the creation of a PWA that lists information about games submitted to the A-Frame category in the [js13kGames 2017](#) competition. This tutorial includes all the basics for creating a PWA, with additional features, including notifications, push, and app performance.

Reference

Reference documentation for the web technologies you will use to build a PWA.

Web app manifest

[Web app manifest members](#)

Developers can use web app manifest members to describe a PWA, customize its appearance, and more deeply integrate it into the operating system.

Cross Device Support

[mashable.com/article/apple-reverses-decision-home-screen-web-apps-pwa-eu](#)

[Yating Wu on X: "L...](#) [Sterling Crispin on...](#) [Daniel Johnson on...](#) [Michel on X: "... In...](#) [Post | Feed | Link...](#) [phidata/cookbook/...](#) [Tereza Tizkova on...](#) [»](#)

 [Search](#)

Home > Tech

Apple reverses decision to kill home screen web apps in the EU

Progressive web apps will remain on iOS after backlash from users over Apple's DMA-inspired decision.

By [Matt Binder](#) on March 1, 2024 [f](#) [X](#) [d](#)



PWA – An Advanced Feature in the Final Project Rubric

4	Advanced Features (Select 1 advanced feature to equal to 4 grade points. Multiple advanced feature items will not be graded.)	
4.1	MenuScanOrder: Allow staff to view and mark an order as completed	4
4.2	ChartTale & EvalForm: Support additional chart types or additional survey question types and/or visualisations	4
4.3	EvalForm: Allow survey responses to be exported	4
4.4	Implement social sharing, likes and commenting	4
4.5	Implement a recommendation algorithm	4
4.6	Include Machine Learning or GenAI functionality (e.g. auto-create content, write analysis, you can be creative....)	4
4.7	Implement a Progressive Web Application (with Notifications and Mobile Device Specific features)	4

This is why a notifications example is not covered.

... but Notifications are hard on iOS.

Prompt Engineering for Code Generation (Part 2)

The way we teach programming and what can be built is changing

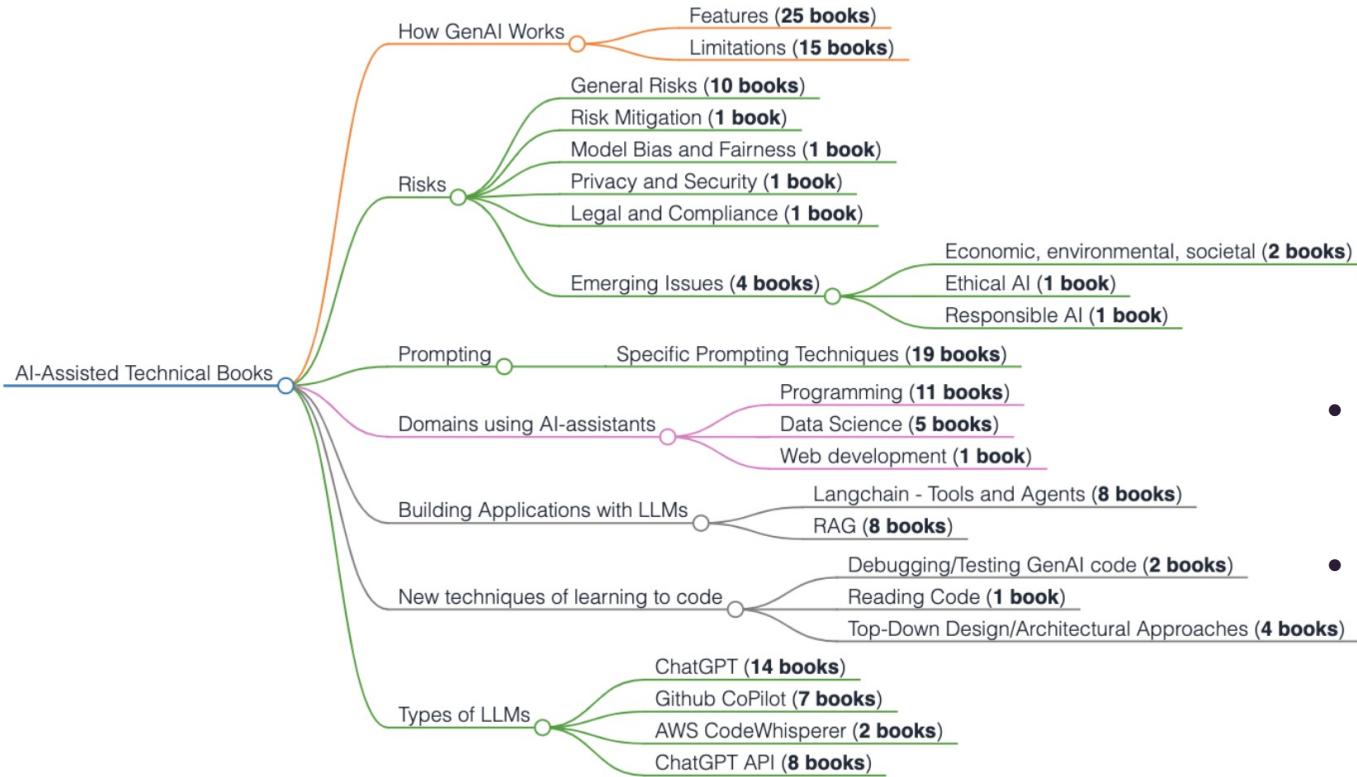
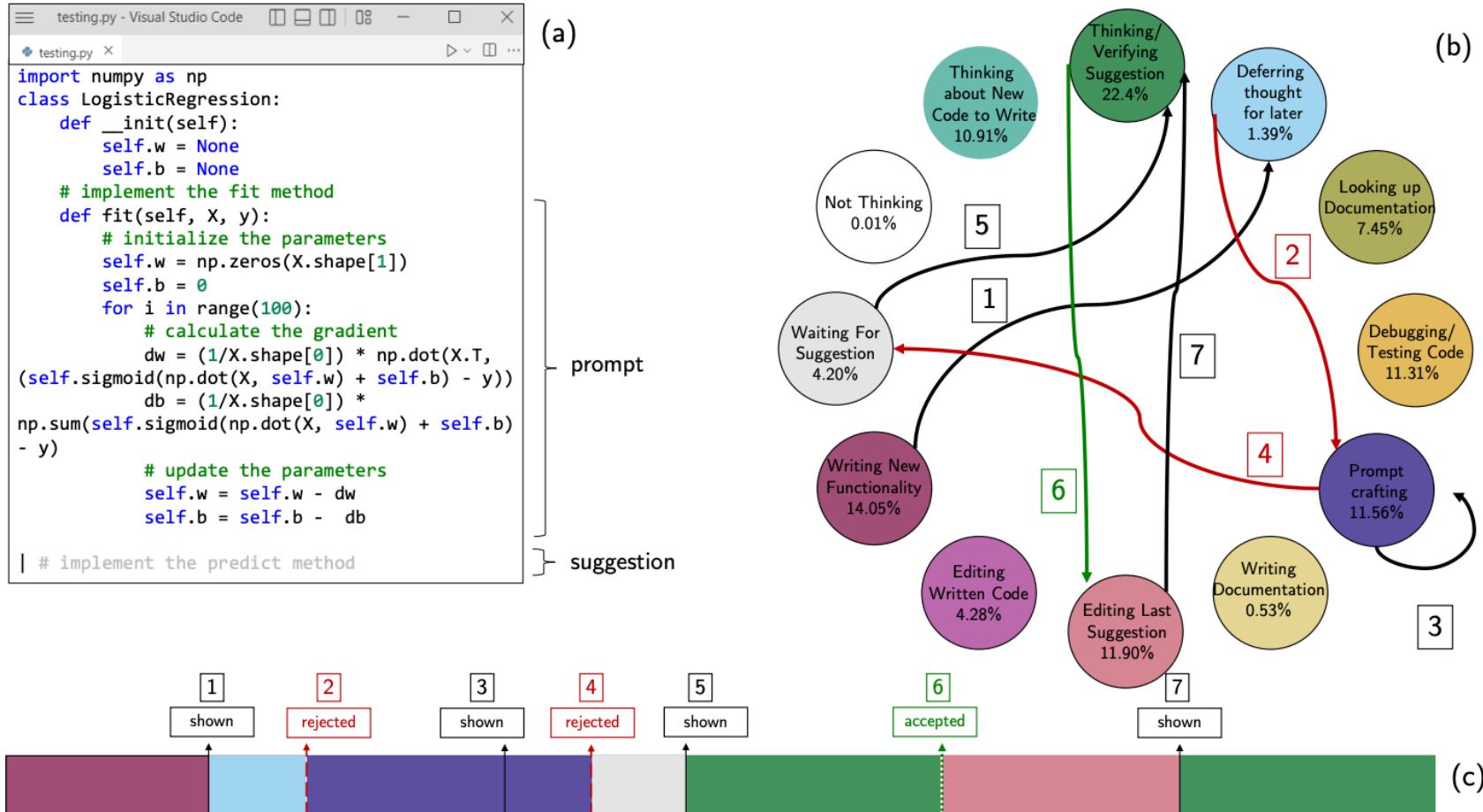


Fig. 1. Mindmap summary of the major topics covered in AI-Assisted programming books.

- **New techniques to teach programming but research is really needed in this area**

- **Integration of AI-enhanced functionality by programmers**

Mental Models while using AI Assistants



Mozannar, H., Bansal, G., Fourny, A., & Horvitz, E. (2024). Reading between the lines: Modeling user behavior and costs in AI-assisted programming. In *CHI 2024*. Retrieved from <https://www.microsoft.com/en-us/research/publication/reading-between-the-lines-modeling-user-behavior-and-costs-in-ai-assisted-programming/>

The way we teach programming and what can be built is changing

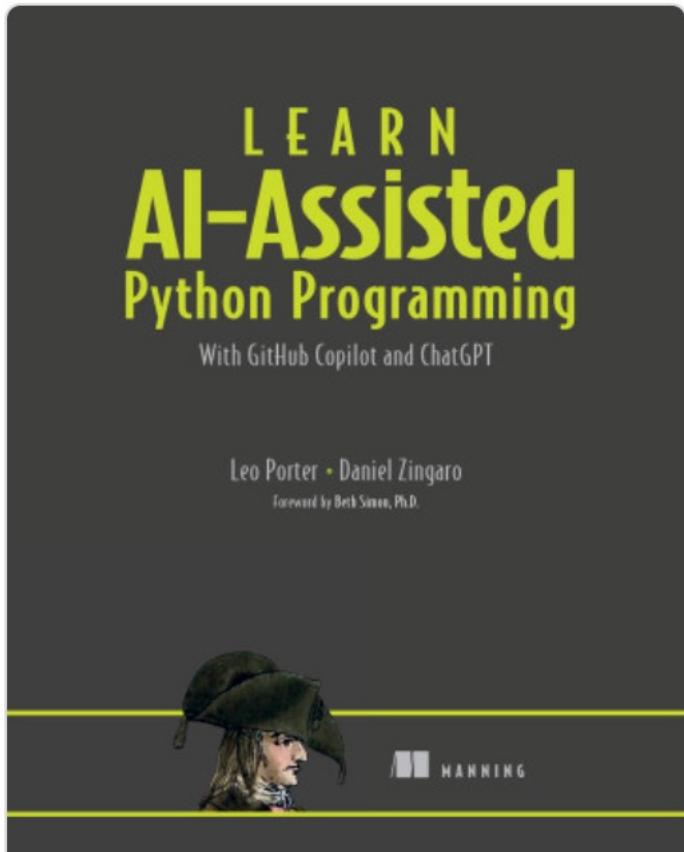


table of contents

You can see this entire book for free. Click anywhere in the table of contents to start reading

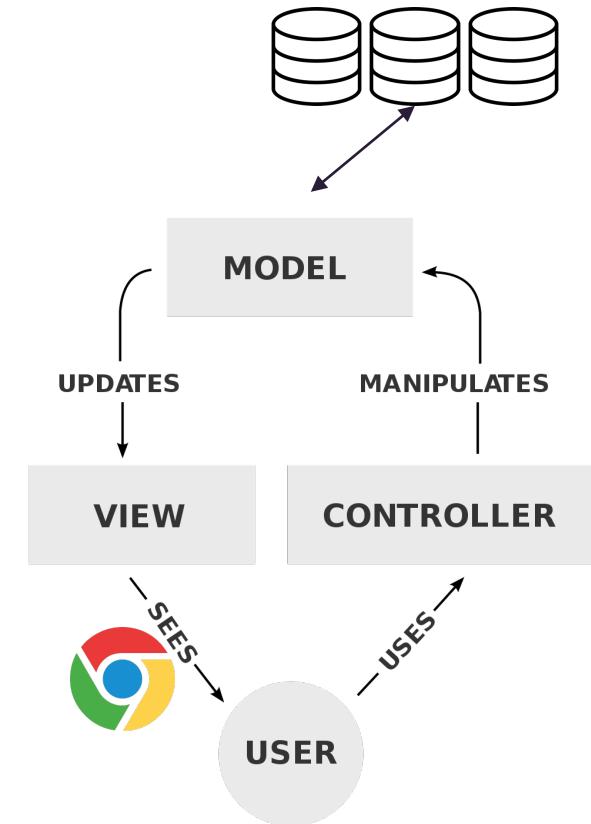
⊕ detailed TOC

- [1 INTRODUCING AI-ASSISTED PROGRAMMING WITH COPILOT ▶](#)
- [2 GETTING STARTED WITH COPILOT ▶](#)
- [3 DESIGNING FUNCTIONS ▶](#)
- [4 READING PYTHON CODE: PART 1 ▶](#)
- [5 READING PYTHON CODE: PART 2 ▶](#)
- [6 TESTING AND PROMPT ENGINEERING ▶](#)
- [7 PROBLEM DECOMPOSITION ▶](#)
- [8 DEBUGGING AND BETTER UNDERSTANDING YOUR CODE ▶](#)
- [9 AUTOMATING TEDIOUS TASKS ▶](#)
- [10 MAKING SOME GAMES ▶](#)
- [11 FUTURE DIRECTIONS ▶](#)
- [REFERENCES](#)

<https://www.manning.com/books/learn-ai-assisted-python-programming>

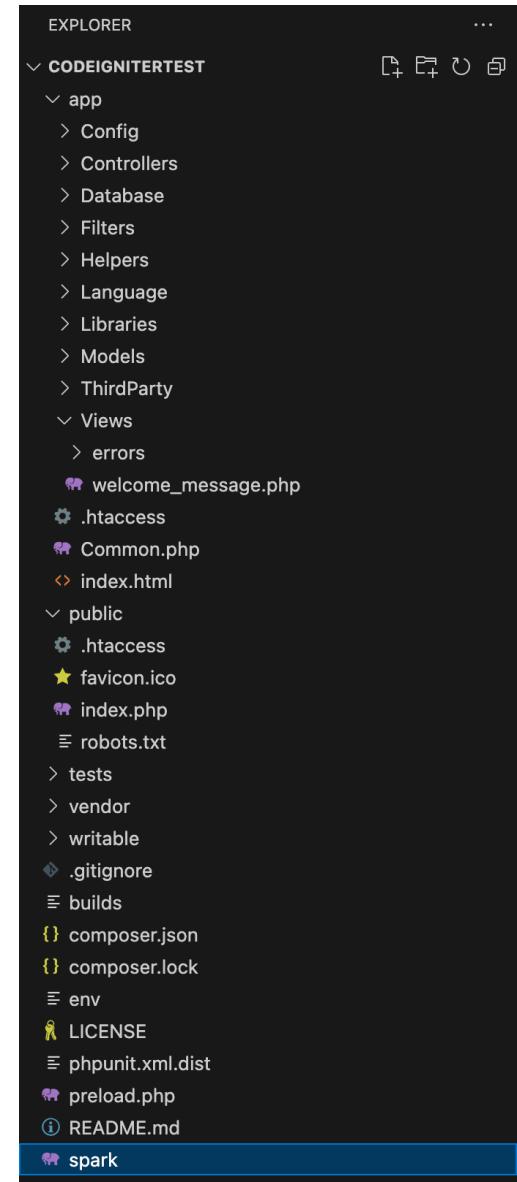
Benefits of MVC

- This will help us to use GenAI to code larger applications
- **Simplifies Complex Applications:** By separating the application into three components, developers can focus on one aspect without affecting others.
- **Facilitates Parallel Development:** Different developers can work on the Model, View, and Controller simultaneously.
- **Easy to Modify or Scale:** Changes to the business logic or user interface can be made with minimal impact on the other components.
- **Supports Reusability:** Components can be reused across different parts of the application or in different applications.



A CodeIgniter Project – Main Folders

- **app/:**
The heart of your CodeIgniter application, containing the models, views, controllers, and configurations.
It's where the majority of your development work will take place.
 - **Controllers/:**
Contains your application's controllers that handle requests, execute business logic, and call views.
 - **Models/:**
Houses the models that represent and work with your application's data.
 - **Views/:**
Where your HTML files go. These files are responsible for rendering the user interface.
 - **Config/:**
Contains configuration files, where you can customize the way CodeIgniter works and behaves.
- **public/:**
The document root of your application. It contains the index.php file, which is the entry point for all requests entering your application, and assets like stylesheets, JavaScript files, and images.



You know very specific terminology,
have practise with CodeIgniter and
have a CI project you can test with!

Use this to build with AI.

Prompt Example 1: Re-imagine a HTML Design

- **Prompt**

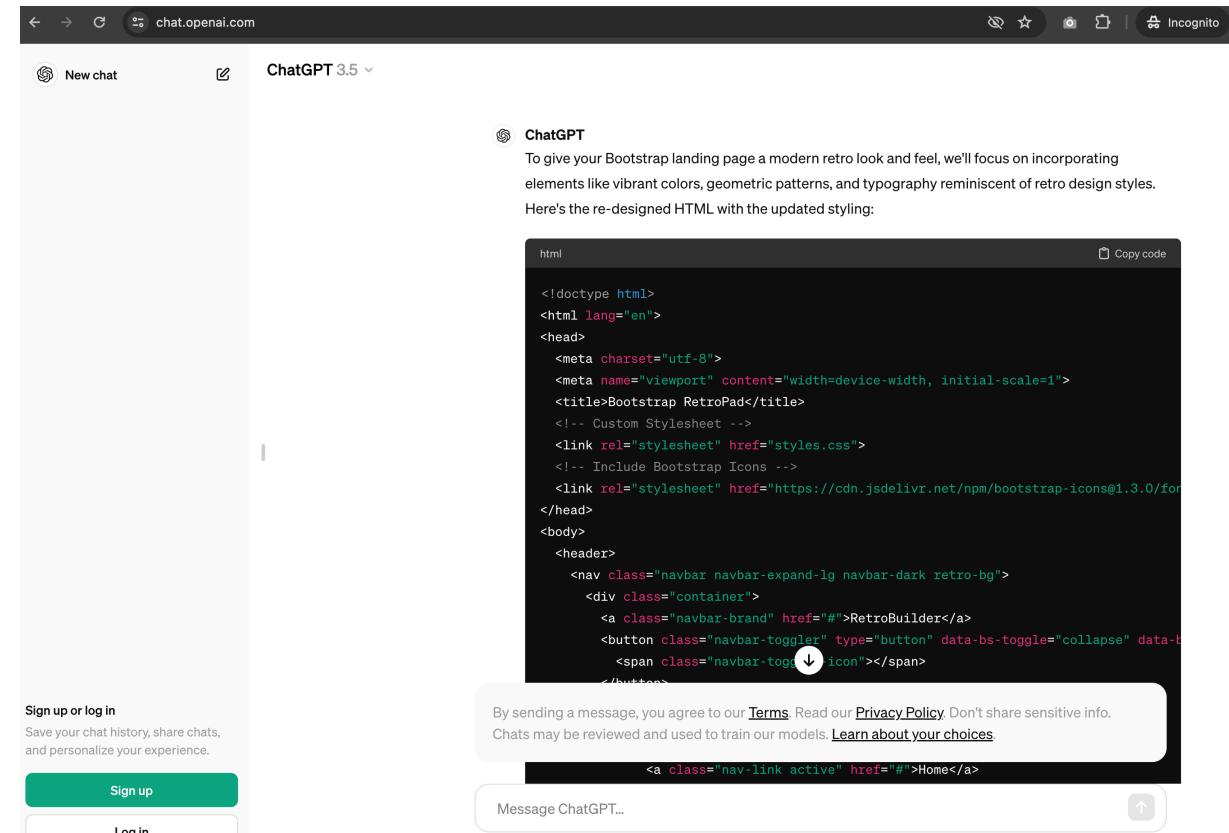
Please act as a creative web application designer and re-design this Bootstrap Landing page to have a modern retro look and feel. Keep all the content but give me a new fresh design.

```

Paste in the HTML

```

- Using the free version of ChatGPT that does not need a Login



To give your Bootstrap landing page a modern retro look and feel, we'll focus on incorporating elements like vibrant colors, geometric patterns, and typography reminiscent of retro design styles. Here's the re-designed HTML with the updated styling:

```
html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Bootstrap RetroPad</title>
  <!-- Custom Stylesheet -->
  <link rel="stylesheet" href="styles.css">
  <!-- Include Bootstrap Icons -->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.3.0/font/bootstrap-icons.css" type="font-style">
</head>
<body>
  <header>
    <nav class="navbar navbar-expand-lg navbar-dark retro-bg">
      <div class="container">
        <a class="navbar-brand" href="#">RetroBuilder</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">
          <span class="navbar-toggler-icon"></span>
        </button>
      </div>
    </nav>
  </header>
  <div class="container py-5" style="text-align: center;">
    <h1>Bootstrap RetroPad</h1>
    <p>A modern, retro-themed Bootstrap landing page template.</p>
    
    <div>
      <h2>Features</h2>
      <ul style="list-style-type: none; padding-left: 0; margin: 0; column-count: 2; column-gap: 20px;">
        <li><span class="icon icon-1"></span> Grid-based layout</li>
        <li><span class="icon icon-2"></span> Responsive design</li>
        <li><span class="icon icon-3"></span> Customizable colors</li>
        <li><span class="icon icon-4"></span> Built-in typography</li>
        <li><span class="icon icon-5"></span> Modular components</li>
        <li><span class="icon icon-6"></span> SEO friendly</li>
      </ul>
    </div>
  </div>
  <div class="text-center" style="margin-top: 20px;">
    <button class="btn btn-primary" href="#">Get Started</button>
  </div>
</body>
```

By sending a message, you agree to our [Terms](#). Read our [Privacy Policy](#). Don't share sensitive info. Chats may be reviewed and used to train our models. [Learn about your choices](#).

Home

Message ChatGPT...

ChatGPT can make mistakes. Consider checking important information. Read our [Terms](#) and [Privacy Policy](#).

Prompt Example 1: Re-imagine a HTML Design

- ## Prompt

Please act as a creative web application designer and re-design this Bootstrap 5 Landing page to have a modern retro look and feel. Keep all the content but give me a new fresh design. You can transform the whole layout, change the colour and the navigation bar. Be extremely creative! Keep the stylesheet links the same. Surprise me! The resulting design should not look like it was created using Bootstrap 5.

...

Paste in the HTML from Lab 4

...

ChatGPT 3.5 ▾

ChatGPT

Sure, let's infuse some retro vibes into this Bootstrap landing page while keeping its functionality intact. Here's the revamped design:

```
html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Retro ResumeBuilder</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.3.0/font/bootstrap-icons.css" rel="stylesheet">
<style>
  /* Custom Styles */
  body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #F6F6F6;
    color: #333333;
  }

  header {
    background-color: #4A4A4A;
    padding-top: 20px;
    padding-bottom: 20px;
  }
</style>
```

By sending a message, you agree to our [Terms](#). Read our [Privacy Policy](#). Don't share sensitive info. Chats may be reviewed and used to train our models. [Learn about your choices](#).

font-size: 2rem;

Message ChatGPT... ↑

Prompt Example 1: Re-imagine a HTML Design

- **Prompt**

Please act as a creative web application designer and re-design this Bootstrap 5 Landing page to have a modern retro look and feel. Keep all the content but give me a new fresh design. You can transform the whole layout, change the colour and the navigation bar. Be extremely creative! Keep the stylesheet links the same. Surprise me! The resulting design should not look like it was created using Bootstrap 5.

...

Paste in the HTML from Lab 4

...

Retro ResumeBuilder

Home Login

Create Professional Resumes Effortlessly

Retro ResumeBuilder simplifies the process of creating stunning resumes that stand out.

[Get Started](#)



Key Features

Customizable Templates
Choose from a variety of professionally designed resume templates.

Easy-to-Use Interface
Our intuitive interface makes it simple to create and edit your resume.

PDF & Word Downloads
Easily download your resume in PDF or Word format for sharing.

© 2024 Retro ResumeBuilder. All rights reserved.
[Privacy Policy](#) [Terms of Service](#)

Prompt Example 2: Creating a Base Template

- ## Prompt

Please act as an expert CodeIgniter 4 developer. Convert the HTML into a base template with the header and footer. Then make a Route for /home, a controller for home that calls a view called landingpage that extends the base template and displays the content from the provided HTML. Make sure you name the files you create.

```

Paste in the HTML

```



ChatGPT

To convert the provided HTML into a base template with header and footer, and then integrate it into a CodeIgniter 4 application, follow these steps:

1. Create Base Template:

- Create a new file named `base_template.php` in the `app/Views` directory of your CodeIgniter project.
- Copy the content of the HTML header (from `<!doctype html>` to `<style>`) into the `base_template.php` file.
- Replace the hard-coded links for CSS and JavaScript files with CodeIgniter's `base_url()` function.
- Keep the header and footer content as placeholders for now.

2. Create Route:

- Open the `app/Config/Routes.php` file.
- Add a new route for the home page:

```
php
```

```
$routes->get('/', 'HomeController::index');
```

Copy code

3. Create Controller:

- Create a new file named `HomeController.php` in the `app/Controllers` directory.
- Define the `HomeController` class with an `index()` method.
- Inside the `index()` method, load the view `landingpage` and pass any necessary data.

Prompt Example 2: Creating a Base Template

• Prompt

Please act as an expert CodeIgniter 4 developer. Convert the HTML into a base template with the header and footer. Then make a Route for /home, a controller for home that calls a view called landingpage that extends the base template and displays the content from the provided HTML. Make sure you name the files you create.

```

Paste in the HTML

```

4. Create Landing Page View:

- Create a new file named `landingpage.php` in the `app/Views` directory.
- Copy the content of the landing page HTML (from `<main>` to `</main>`) into the `landingpage.php` file.
- Extend the `base_template.php` by using CodeIgniter's view inheritance.

Here's how your files should look:

app/Views/base_template.php:

```
php
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Retro ResumeBuilder</title>
    <link href=<?= base_url('public/css/bootstrap.min.css') ?>" rel="stylesheet">
    <link rel="stylesheet" href=<?= base_url('public/css/bootstrap-icons.css') ?>">
    <style>
        /* Custom Styles */
        /* Paste your CSS styles here */
    </style>
</head>
<body>
```

Prompt Example 2: Creating a Base Template

- **Prompt**

Please act as an expert CodeIgniter 4 developer. Convert the HTML into a base template with the header and footer. Then make a Route for /home, a controller for home that calls a view called landingpage that extends the base template and displays the content from the provided HTML. Make sure you name the files you create.

```

Paste in the HTML

```

- Note: A GPT4 Pro Subscription would produce all the code (unless it get lazy)

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Retro ResumeBuilder</title>
<link href="= base_url('public/css/bootstrap.min.css') ?&gt;" rel="stylesheet"&gt;
&lt;link rel="stylesheet" href="<?= base_url('public/css/bootstrap-icons.css') ?&gt;"&gt;
&lt;style&gt;
    /* Custom Styles */
    /* Paste your CSS styles here */
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;header&gt;
    &lt;!-- Paste your header content here --&gt;
&lt;/header&gt;

&lt;main&gt;
    &lt;?= $this-&gt;renderSection('content') ?&gt;
&lt;/main&gt;

&lt;footer&gt;
    &lt;!-- Paste your footer content here --&gt;
&lt;/footer&gt;

&lt;script src="<?= base_url('public/js/bootstrap.bundle.min.js') ?&gt;"&gt;&lt;/script&gt;
&lt;/body&gt;
&lt;/html&gt;</pre
```

Prompt Example 2: Creating a Base Template

- **Prompt**

Please act as an expert CodeIgniter 4 developer. Convert the HTML into a base template with the header and footer. Then make a Route for /home, a controller for home that calls a view called landingpage that extends the base template and displays the content from the provided HTML. Make sure you name the files you create.

```

Paste in the HTML

```

- Note: A GPT4 Pro Subscription would produce all the code (unless it get lazy)

app/Controllers/HomeController.php:

```
php

<?php

namespace App\Controllers;

class HomeController extends BaseController
{
    public function index()
    {
        return view('landingpage');
    }
}
```

app/Views/landingpage.php:

```
php

<?= $this->extend('base_template') ?>

<?= $this->section('content') ?>
<section class="py-5 bg-light">
    <!-- Paste the content of the landing page HTML from &lt;main&gt; to &lt;/main&gt; here --&gt;
&lt;/section&gt;
&lt;?= $this-&gt;endSection() ?&gt;</pre>
```

Make sure you have configured your CodeIgniter project correctly, including setting up the base URL and autoloading. Also, ensure that Bootstrap CSS and JavaScript files are placed in the correct directory (`public/css` and `public/js` respectively).

Prompt Example 3: Make a Glossary App

- ## Prompt

Please act as an expert CodeIgniter 4 developer. Make me a simple glossary tool that displays terms in alphabetical order and allows terms to be added, edited and deleted. The views should use the base template you already made.

You will need to create a model file for a glossary table with id, term and definition fields. Provide the routes, controllers and view. Make sure you name all the files you create.



ChatGPT

To create a simple glossary tool in CodeIgniter 4 that allows users to view terms in alphabetical order and perform CRUD operations (Create, Read, Update, Delete), follow these steps:

1. Create Database Table:

- Create a table named `glossary` in your database with the following fields:
 - `id` (Primary Key)
 - `term` (VARCHAR)
 - `definition` (TEXT)

2. Create Model:

- Create a new file named `GlossaryModel.php` in the `app/Models` directory.
- Define the `GlossaryModel` class extending `CodeIgniter\Model`.
- Configure the model to interact with the `glossary` table.

3. Create Controller:

- Create a new file named `GlossaryController.php` in the `app/Controllers` directory.
- Define the `GlossaryController` class.
- Implement methods for displaying the glossary, adding a term, editing a term, and deleting a term.

4. Create Views:

- Create view files (`index.php`, `add.php`, `edit.php`) in the `app/Views/glossary` directory.
- Utilize the base template for these views.
- Implement forms for adding, editing, and deleting terms.

5. Define Routes:

- Define routes in the `app/Config/Routes.php` file to map URLs to controller methods.

Prompt Example 3: Make a Glossary App

- ## Prompt

Please act as an expert CodeIgniter 4 developer. Make me a simple glossary tools that displays terms in alphabetical order and allows terms to be added, edited and deleted. The views should use the base template you already made.

You will need to create a model file for a glossary table with id, term and definition fields. Provide the routes, controllers and view. Make sure you name the all the files you create.

Model: `app/Models/GlossaryModel.php`:

```
php
<?php

namespace App\Models;

use CodeIgniter\Model;

class GlossaryModel extends Model
{
    protected $table      = 'glossary';
    protected $primaryKey = 'id';
    protected $allowedFields = ['term', 'definition'];
}
```

Prompt Example 3: Make a Glossary App

- ## Prompt

Please act as an expert CodeIgniter 4 developer. Make me a simple glossary tools that displays terms in alphabetical order and allows terms to be added, edited and deleted. The views should use the base template you already made.

You will need to create a model file for a glossary table with id, term and definition fields. Provide the routes, controllers and view. Make sure you name the all the files you create.

Controller: `app\Controllers\GlossaryController.php`:

```
php
<?php

namespace App\Controllers;

use App\Models\GlossaryModel;

class GlossaryController extends BaseController
{
    public function index()
    {
        $model = new GlossaryModel();
        $data['terms'] = $model->orderBy('term', 'ASC')->findAll();
        return view('glossary/index', $data);
    }

    // Implement other CRUD methods here
}
```

Prompt Example 3: Make a Glossary App

- ## Prompt

Please act as an expert CodeIgniter 4 developer. Make me a simple glossary tool that displays terms in alphabetical order and allows terms to be added, edited and deleted. The views should use the base template you already made.

You will need to create a model file for a glossary table with id, term and definition fields. Provide the routes, controllers and view. Make sure you name all the files you create.

- Can ask follow up questions to get:
 - SQL to create the table or a Migration
 - The views

Views:

- `app/Views/glossary/index.php`: Display all terms in alphabetical order.
- `app/Views/glossary/add.php`: Form for adding a new term.
- `app/Views/glossary/edit.php`: Form for editing a term.

Routes:

Define routes in `app/Config/Routes.php` to map URLs to controller methods. For example:

```
php Copy code  
  
$routes->get('/glossary', 'GlossaryController::index');  
$routes->get('/glossary/add', 'GlossaryController::add');  
$routes->post('/glossary/add', 'GlossaryController::save');  
$routes->get('/glossary/edit/(:num)', 'GlossaryController::edit/$1');  
$routes->post('/glossary/update/(:num)', 'GlossaryController::update/$1');  
$routes->get('/glossary/delete/(:num)', 'GlossaryController::delete/$1');
```

Make sure to replace `(:num)` with appropriate route parameters as per your application requirements.

This structure provides a foundation for your glossary tool. You can then expand upon it by adding validation, authentication, and additional features as needed.

Prompt Example 3: Make a Glossary App

- ## Prompt

Please give me the code for all the views.

 You
Please give me the code for all the views.

 ChatGPT
Certainly! Here are the view files for the glossary tool:

1. Index View (`app/Views/glossary/index.php`):

```
php
<?= $this->extend('base_template') ?>

<?= $this->section('content') ?>
<section class="py-5">
    <div class="container">
        <h2>Glossary</h2>
        <a href="= site_url('glossary/add') ?" class="btn btn-primary mb-3">Add Term</a>
        <table class="table">
            <thead>
                <tr>
                    <th>Term</th>
                    <th>Definition</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <?php foreach ($terms as $term) : ?>
                <tr>
                    <td><?= $term['term'] ?></td>
                    <td><?= $term['definition'] ?></td>
                    <td>
```

Week 9: Todo

- Weekly RiPPL E task is due in Week 9 on Friday at 3pm
- Week 9 Lab 8
 - No prepared lab
 - Time to work on your project and ask questions
- Continue to work on your Project
 - Code Review in Week 10
- Week 10 Lecture – Guest Lecture by AWS Solution Architects



CREATE CHANGE

Thank you