Project: Landing Page

The whole CSS series in one real page: a ChaiPoint landing page with variables, a flex navbar, hero, grid features, footer and a phone check.

Introduction

Time to spend everything you have earned. This project builds **ChaiPoint** — a complete landing page for a tea brand — from an empty folder to a page that survives a phone: CSS **variables** and a base reset first, a **flexbox navbar** (sticky, Lesson 2.8), a **hero** with a hover-lift call-to-action (2.10/2.11), a **grid of features** (2.7), a proper **footer**, and a **responsive** finish (2.9). Every rule in the file comes from a lesson in this series — this page is the series, assembled. *(Roman Urdu: Ye project pichli tamam lessons ka imtihan hai — aik asal page, shuru se aakhir tak.)*

Key points

- **Project order matters:** (1) variables + base, (2) big layout pieces (navbar, hero, sections), (3) details (hover states, shadows), (4) responsive overrides **last**. Styling details before the skeleton exists means re-doing them.
- **Variables from day one:** brand colours and the radius live in `:root` (Lesson 2.14 goes deep) — `--brand`, `--ink`, `--paper`. Change one value, the whole page follows. No hunting hex codes through the file.
- **A sticky navbar** is just the 2.6 flex navbar plus `position: sticky; top: 0` from 2.8 — two old tools shaking hands. It stays reachable on a long page without the content-jump problems of `fixed`.
- **The hero sells in one screen:** big heading, one supporting line, one call-to-action button — with the 60-seconds hover lift. Everything else on the page supports that one button.

  (Roman Urdu: Landing page ka kaam aik hai — visitor ko aik action par raazi karna.)
- **Features in Grid, rows in Flex** — the 2.7 decision rule applied without thinking: the 3-card features area is a layout (grid), the navbar and button rows are lines (flex).
- **The phone check is part of the build, not an afterthought:** after the desktop page looks right, open the device toolbar at 390px and fix what breaks — *then* the project is done.

Code examples

Example 1 — Variables, base, and the sticky flex navbar

```
:root {
  --brand: #0b6e4f;
  --brand-dark: #084f39;
  --ink: #14231d;
  --paper: #faf9f6;
  --accent: #e8b04b;
  --radius: 14px;
}
* { box-sizing: border-box; margin: 0; padding: 0; }
body { font-family: "Inter", Arial, sans-serif; color: var(--ink); background: var(--paper); }

.navbar {
  position: sticky; top: 0;
  display: flex; justify-content: space-between; align-items: center;
  padding: 14px 24px;
  background: var(--paper);
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.08);
}
.navbar .links { display: flex; gap: 22px; }
.navbar a { color: var(--ink); text-decoration: none; font-weight: 600; }
```

What it does: The project's foundation in one block: palette variables (2.14's preview), the border-box reset (2.4), the body font and colours, then the navbar — flex for the row (2.6), sticky for the stay-on-top behaviour (2.8), a whisper of shadow to lift it off the scrolling page.

What you will see: a clean top bar that parks itself at the top of the screen as the page scrolls — logo left, links right — and recolouring the entire site is now a one-line edit in `:root`.

Example 2 — Hero with a call-to-action that earns its hover

```
.hero {
  text-align: center;
  padding: 80px 20px 64px;
}
.hero h1 { font-size: 52px; line-height: 1.15; }
.hero h1 span { color: var(--brand); }
.hero p { font-size: 19px; color: #4a5a53; margin: 16px auto 28px; max-width: 560px; }
.cta {
  display: inline-block;
  background: var(--brand); color: white;
  padding: 14px 32px; border-radius: var(--radius);
  font-weight: 700; text-decoration: none;
  transition: all 0.3s ease;
}
.cta:hover { background: var(--brand-dark); transform: translateY(-3px); box-shadow: 0 10px
20px rgba(0,0,0,0.18); }
```

What it does: The hero follows the one-screen rule: a 52px promise, one grey supporting line (capped at 560px so it never stretches into a hard-to-read ribbon — a quiet typography pro-move), and the single green CTA carrying the full 2.11 hover treatment.

What you will see: a calm, confident first screen: big green-accented headline, centred pitch, and a button that lifts under your mouse like it *wants* to be clicked. That button is the page's entire business model.

Example 3 — Features grid, footer, and the responsive finish

```
.features {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 18px;
  max-width: 1024px; margin: 0 auto; padding: 24px 20px 64px;
}
.feature {
  background: white; border: 1px solid #e6e9e4;
  border-radius: var(--radius); padding: 24px;
}
footer {
  background: var(--ink); color: var(--paper);
  text-align: center; padding: 28px 20px;
}

@media (max-width: 768px) {
  .features { grid-template-columns: 1fr; }
  .hero h1 { font-size: 36px; }
  .navbar .links { gap: 14px; }
}
```

What it does: The features area is a 3-column grid centred in a 1024px column (2.7 + 2.4's `margin: 0 auto`); the footer closes the page in the ink colour. Then the media query does the whole phone job in four lines: one feature per row, a smaller hero headline, tighter nav gaps (2.9).

What you will see: scroll the finished page on desktop — sticky bar, airy hero, three white feature cards, dark footer. Shrink to 390px: the cards stack, the headline steps down, nothing overflows sideways. *That* is a finished landing page — PHASE 2 COMPLETE, and the page is ready for JavaScript to wake it up in Phase 3.

Quick recap

- Build order: **variables + base** → **layout** → **details** → **responsive last**.
- The landing page is old lessons cooperating: sticky (2.8) + flex (2.6) navbar, hover CTA (2.11), grid features (2.7), media-query finish (2.9).
- Variables in `:root` make the whole design re-themeable from one block.

Common mistakes

1. **Styling before structure** — decorating a hero before the navbar and sections exist means re-doing the decoration when the layout moves. Skeleton first, lipstick last.

2. **Hard-coding the brand colour twenty times** — the day the client says "make it blue", twenty edits (and one you miss) await. Variables exist so that day costs one edit.
3. **Declaring victory at desktop width** — skipping the 390px check ships a broken page to the majority of visitors. The phone check is the last step of the build, every time.

Practice

1. Rebuild ChaiPoint as your own brand (name, colours, one-line pitch) using only this lesson's three examples as the starting file — then change the entire colour scheme by editing **only** the `:root` block.
2. Add a fourth section of your choice (a menu list or prices — a table from HTML 1.5 styled in this design system) and a second media-query breakpoint at 1024px. Verify at 390 / 768 / 1280px.

Practice quiz

1. What comes first in the landing page build order?

- Hover states and shadows
- Variables + base styles, then big layout pieces
- Responsive overrides
- The footer

Answer: Variables + base styles, then big layout pieces

2. When do responsive overrides happen in the project order?

- First, before anything else
- In the middle, with the layout
- Last, after details
- Never, the page is desktop-only

Answer: Last, after details

3. At what width is the phone check done at the end of the build?

- 1024px
- 768px
- 390px
- 1440px

Answer: 390px

Watch the video lesson: https://youtu.be/XrMNRqJ8_xE

Read online: https://scopetreasure.com/notes/css-project-landing-page · More free tools and notes at https://scopetreasure.com