SkillNewton

Responsive Design Testing Checklist

Test layouts, images, text, touch targets, forms and navigation across screen sizes, orientations, zoom levels and real devices.

Use this checklist to test whether a website works well on phones, tablets, laptops and large monitors. It suits front-end developers, testers and students building their first responsive projects. Test each page type (home, listing, article, form) at the widths listed, in both orientations, and on at least one real Android phone and one iPhone. Note every problem with the width and device where you saw it, then fix and re-test.

How to use it

1. **Pick your page types.** Choose one example of each template, such as home, blog post, product and contact form, rather than every page.
2. **Test in DevTools first.** Use device mode or responsive mode to step through the widths in the breakpoints section and drag between them.
3. **Confirm on real devices.** Repeat the key checks on real phones and tablets, because touch, keyboards and browser toolbars behave differently.
4. **Log and re-test.** Record each issue with width, device and browser, fix it, then re-test the whole page.

Set-up and viewport

- [ ] Confirm the viewport meta tag is present
  `<meta name="viewport" content="width=device-width, initial-scale=1">`. Without it phones show a zoomed-out desktop page.
- [ ] Make sure the tag does not disable zooming
  Avoid `user-scalable=no` or `maximum-scale=1`; users must be able to pinch-zoom.
- [ ] Check media queries use sensible units
  Breakpoints in `em` or `rem` respond better when users change their default font size.
- [ ] Test in current Chrome, Edge, Firefox and Safari
  Safari on iOS is essential: every iPhone browser uses Apple's WebKit engine in most regions.
- [ ] Clear cache or use a private window between fixes
  Old cached CSS is a common reason a fix appears not to work.

Breakpoints and widths to test

- [ ] 320px wide
  Smallest common phone width and the width WCAG reflow is measured at. Nothing should overflow horizontally.
- [ ] 375px and 414px wide
  Typical modern phone widths. Check headers, cards and buttons fit comfortably.
- [ ] 768px wide (tablet portrait)
  Two-column layouts often start here; check they are not cramped.

- [ ] 1024px wide (tablet landscape, small laptop)
  Navigation frequently switches from a menu button to a full menu around this width.
- [ ] 1280px and 1440px wide
  Common laptop and desktop widths. Check content is not stretched too wide.
- [ ] 1920px and wider
  Set a `max-width` on content so lines do not become too long to read.
- [ ] Drag slowly between breakpoints
  Bugs often appear at in-between widths, such as 600px or 900px, not at the exact breakpoints.

Layout and overflow

- [ ] Check for horizontal scrolling on every page
  In the console, compare `document.documentElement.scrollWidth` with `clientWidth` to find overflow.
- [ ] Make wide tables scroll inside their own container
  Wrap tables in a container with `overflow-x: auto` rather than letting the whole page scroll.
- [ ] Break long words and URLs
  Use `overflow-wrap: anywhere` or `break-word` for user content and long links.
- [ ] Check fixed and sticky elements on small screens
  Sticky headers, cookie banners and chat widgets must not cover content or focused controls.
- [ ] Check embeds and iframes scale
  Use `aspect-ratio` or a responsive wrapper for videos and maps.
- [ ] Check content order makes sense when columns stack
  Visual reordering with CSS should not confuse keyboard or screen reader order.

Images and media

- [ ] Make images flexible
  `max-width: 100%; height: auto;` stops images overflowing their container.
- [ ] Use `srcset` and `sizes` for content images
  The browser can then download a smaller file on phones and a sharper one on high-density screens.
- [ ] Use `<picture>` for art direction or modern formats
  Serve a different crop on small screens, or AVIF and WebP with a fallback.
- [ ] Set `width` and `height` attributes
  This reserves space and prevents layout shift (CLS) as images load.
- [ ] Check important parts of images are not cropped
  Background images with `background-size: cover` may cut off faces or text on narrow screens.
- [ ] Check the network panel for oversized downloads
  Throttle to a mobile network in DevTools and confirm phones are not downloading desktop-sized images.

Typography and readability

- [ ] Use a body text size of around 16px or more
  Smaller text on phones is hard to read and may trigger zoom on form focus in iOS Safari.
- [ ] Keep line length comfortable
  Roughly 45–75 characters per line on desktop; use `max-width` in `ch` units.

- [ ] Check headings scale down on phones
  Use `clamp()` so large headings do not break into one word per line.
- [ ] Check line height and spacing
  Body text with a line height around 1.5 is easier to read; WCAG 1.4.12 requires layouts to survive increased text spacing.
- [ ] Check text contrast in every state
  Text over images, disabled-looking buttons and placeholder text often fail contrast on small screens in sunlight.

Touch targets, forms and navigation

- [ ] Make touch targets at least 24 by 24 CSS pixels
  This is the WCAG 2.2 Target Size (Minimum) requirement at Level AA, with some exceptions such as inline links.
- [ ] Aim for 44 by 44 CSS pixels for main controls
  44 by 44 is the WCAG enhanced (AAA) size and a comfortable target for thumbs.
- [ ] Leave space between adjacent links and buttons
  Small, tightly packed icons cause mis-taps, especially in footers and toolbars.
- [ ] Use the right input types and attributes
  `type="email"`, `type="tel"`, `inputmode` and `autocomplete` bring up the right keyboard and autofill.
- [ ] Check the on-screen keyboard does not hide fields or errors
  Tap each field on a real phone and make sure the active field and its error message stay visible.
- [ ] Test the mobile menu with touch and keyboard
  The menu button needs an accessible name and `aria-expanded`, and must open, close and trap nothing unexpectedly.
- [ ] Check nothing depends on hover
  Touchscreens have no hover, so dropdowns and tooltips must also work on tap and focus.

Orientation, zoom and reflow

- [ ] Test portrait and landscape on phones and tablets
  WCAG 1.3.4 says content should not be locked to one orientation unless essential.
- [ ] Zoom the desktop page to 200%
  Text must remain readable and functional with no loss of content (WCAG 1.4.4 Resize Text).
- [ ] Check reflow at 320 CSS pixels wide
  At 400% zoom on a 1280px window, content should reflow into one column with no two-way scrolling (WCAG 1.4.10).
- [ ] Increase the browser default font size
  Raise it in browser settings and confirm text grows and nothing overlaps.
- [ ] Check focus stays visible when zoomed
  Sticky headers and footers must not cover the focused element (WCAG 2.2 Focus Not Obscured).

Real devices and tools

- [ ] Use DevTools device mode for quick checks
  Chrome, Edge and Firefox all offer a responsive or device mode; it simulates size, not real touch or browser chrome.
- [ ] Test on at least one real Android phone and one iPhone
  Real devices show browser toolbar resizing, notches, keyboards and performance that emulators miss.
- [ ] Use remote debugging for real-device bugs
  Connect Android via USB to Chrome DevTools, or an iPhone to Safari Web Inspector on a Mac.

- [ ] Check safe areas on notched phones

  Use `env(safe-area-inset-*)` if content runs to the screen edges.

- [ ] Run Lighthouse with the mobile profile

  It flags missing viewport tags, small tap targets and slow images.

- [ ] Record each issue with width, device and browser

  Precise notes make bugs reproducible and quick to fix.

Online version, with the matching free guide: skillnewton.com/resources/templates/responsive-testing-checklist/
Learn this hands-on: PHP Web Development: Foundation & Practical Implementation – skillnewton.com/courses/php-foundation-course/