

INTERNATIONAL ADVANCED LEVEL

INFORMATION TECHNOLOGY

UNIT 2

Topic 10: Designing Web Pages



Academic Year: 2024-2025

✍ Prepared by: Irikefe Hope Oniovosa

1. PRINCIPLES OF DESIGN

A. Design Methodology

1. Iterative Design–Implement–Test Cycle

- **Design:** Sketch out layouts or write user stories.
- **Implement:** Build a quick HTML/CSS/JS prototype.
- **Test:** Gather feedback (peer review, usability testing) and note what works or doesn't.
- **Repeat:** Refine designs based on what you've learned.

2. Wireframes & Mock-ups

- **Wireframes:** Low-fidelity, black-and-white sketches showing page structure (header, nav, content, footer).
- **Mock-ups:** Higher-fidelity designs with typography, colours, and imagery, often created in tools like Figma or Adobe XD.

3. Style Guides

- **Purpose:** Ensure consistency across pages and team members.
- **Contents:**
 - Palette of brand colours (primary, secondary, accents)
 - Typography rules (font families, sizes, line-heights)
 - UI components (buttons, forms, cards) with states (hover, active, disabled)

4. Prototyping

- **Interactive Prototypes:** Click-through designs that behave like real sites.
- **Tools:** Figma, Adobe XD, Sketch – allow you to share prototypes for feedback before any code is written.
-

B. Web Typography

1. Web-Safe Fonts vs. Custom Web Fonts

- **Web-Safe Fonts:** Fonts installed on nearly all devices (e.g., Arial, Times New Roman).
- **Custom Web Fonts:** Hosted services such as **Google Fonts** or **Adobe Fonts**, which give access to a broader range of typefaces.

2. Embedding Web Fonts

- **<link> Method (Google Fonts):**

```

html
<head>
  <link rel="stylesheet"

href="https://fonts.googleapis.com/css?family=Roboto:400,700&display=
swap">
</head>

CSS
body { font-family: 'Roboto', sans-serif; }

```

- **@font-face Method:**

```

CSS
@font-face {
  font-family: 'MyCustomFont';
  src: url('fonts/MyCustomFont.woff2') format('woff2'),
       url('fonts/MyCustomFont.woff') format('woff');
  font-weight: normal;
  font-style: normal;
}
body { font-family: 'MyCustomFont', sans-serif; }

```

C. Core Design Principles

1. Visual Hierarchy

- **Size & Weight:** Larger, bolder headlines draw the eye first.
- **Spacing & Alignment:** Generous white space around important elements makes them stand out.

2. Flow (Reading Patterns)

- **F-Pattern:** Users scan across the top, then down the left side—good for content-heavy pages.
- **Z-Pattern:** Ideal for simpler landing pages—eyes follow a zig-zag from top-left to bottom-right.

3. Colour Theory

- **Complementary Palettes:** Colours opposite each other on the colour wheel (e.g., blue & orange) for contrast.
- **Analogous Palettes:** Neighbouring colours (e.g., green, teal, blue) for harmony.

- **Accessibility:** Always check contrast ratios (e.g., text vs. background must meet WCAG AA standards).

4. **Balance & Contrast**

- **Balance:** Distribute visual “weight” evenly—symmetrical or asymmetrical arrangements can both work.
- **Contrast:** Use light vs. dark or warm vs. cool colours to make elements pop and improve readability.

Interactive Activity (Optional Extension)

- **Wireframe Workshop:**

1. In pairs, sketch a homepage wireframe on paper (include header, nav, hero section, content blocks, footer).
2. Define a simple style guide: choose two brand colours and one web font.
3. Swap with another pair to critique visual hierarchy, flow, and balance.

This framework gives students a practical, hands-on view of web design principles, ensuring they understand not only **how** to code pages but **why** they are structured and styled in particular ways.

2. PRINCIPLES OF ACCESSIBILITY

A. Accessibility Overview

- **WCAG (Web Content Accessibility Guidelines):**
 - Developed by the W3C, WCAG defines how to make web content more accessible to people with disabilities.
 - **Four core principles (POUR):**
 1. **Perceivable:** Information must be presentable to all senses (e.g., text alternatives for images).
 2. **Operable:** Interface elements must be usable via keyboard or other assistive technologies.
 3. **Understandable:** Content and controls must be clear and predictable.
 4. **Robust:** Content must work across a variety of user agents, including assistive tools.
- **Why It Matters:**
 - **Legal Compliance:** Many countries require public websites to meet accessibility standards.
 - **Wider Audience:** Improves usability for everyone—people with visual, auditory, motor, or cognitive impairments.
 - **Better SEO & UX:** Semantic, well-structured content often ranks higher and is easier to maintain.

B. Key Techniques

1. **Semantic HTML & Landmarks**
 - Use proper heading hierarchy (<h1>–<h6>) so screen readers can navigate sections.
 - Employ HTML5 landmarks (<header>, <nav>, <main>, <aside>, <footer>) to define page regions.
2. **Text Alternatives for Non-Text Content**
 - **Images:** Always include meaningful alt attributes.
 - **Multimedia:** Provide captions for videos (<track kind="captions">) and transcripts for audio.
3. **Colour Contrast & Visual Clarity**

- Ensure text has a contrast ratio of at least **4.5:1** against its background (WCAG AA).
- Avoid relying on color alone to convey information (use icons or labels).

4. **Keyboard Navigation**

- All interactive elements (links, buttons, form controls) must be reachable via the Tab key.
- Use `tabindex="0"` to include custom elements in the tab order and `tabindex="-1"` to remove non-interactive elements.

5. **ARIA Roles & Properties**

- When semantic HTML isn't enough, add ARIA attributes:
 - **role="button"** for clickable `<div>`s
 - **aria-label="Close menu"** to name unlabeled icons
 - **aria-live="polite"** to announce dynamic content changes

C. Interactive Activity

1. **Tool:** Choose **WAVE** (<https://wave.webaim.org>) or **Lighthouse** (built into Chrome DevTools).
2. **Task:**
 - Evaluate a provided sample page URL.
 - Identify at least **three** accessibility issues (e.g., missing alt, low contrast, missing landmarks).
 - Record each issue and its WCAG principal violation.
3. **Group Discussion:**
 - Propose concrete fixes for each issue:
 - Add or improve alt text.
 - Adjust CSS to meet contrast ratios.
 - Convert `<div>` menus into `<nav>` or add ARIA roles.
 - Share findings and solutions with the class.

3. PRINCIPLES OF USABILITY

A. Navigation Design

1. Horizontal & Vertical Menus

- **Horizontal Menus:**
 - Best placed at the top of the page (header) for immediate visibility.
 - Limit to **5–7** main items to avoid overcrowding.
 - Use clear labels (e.g., “Home,” “Products,” “Contact”).
 - Indicate the current page with a highlight or underline.
- **Vertical Menus:**
 - Ideal for sidebars or dashboards.
 - Supports deeper hierarchies without compacting space.
 - Collapsible submenus can save screen real estate.
- **Interaction Tips:**
 - Ensure menu items are large enough for easy clicking/tapping (minimum 44×44 px).
 - Provide hover and focus states (color change, underline) for clarity.

2. Dropdown & Breadcrumb Navigation

- **Dropdown Menus:**
 - Group related links under a parent label (e.g., “Services ►”).
 - Use on-click for mobile to avoid hover-only interactions.
 - Keep dropdowns shallow (no more than two levels deep).
- **Breadcrumbs:**
 - Show the user’s path within the site hierarchy (e.g., Home › Products › Electronics).
 - Place near the top of the content area.
 - Improves findability and allows users to backtrack easily.

3. Button Groups

- **Grouping Related Actions:**
 - Place buttons that perform similar tasks (e.g., Save, Cancel, Delete) close together.
 - Use consistent styling (size, color) for primary vs. secondary actions.
- **Layout Tips:**

- Align groups in a predictable location (bottom-right of forms, top for toolbars).
- Provide sufficient spacing so users don't click the wrong button by accident.

B. Device & Browser Targeting

1. Cross-Browser Compatibility

- **Vendor Prefixes & Polyfills:**
 - Use -webkit-, -moz- prefixes for newer CSS features when needed.
 - Include polyfills or fallbacks for JavaScript features (e.g., fetch polyfill).
- **Progressive Enhancement:**
 - Build basic functionality that works everywhere, then layer on advanced features for modern browsers.

2. Functionality & Usability Testing

- **Mobile vs. Desktop:**
 - Test on actual devices when possible.
 - In browser dev tools, use device emulation to check layout and touch interactions.
- **Key Checks:**
 - Menu visibility and touch-target size on small screens.
 - Dropdown behavior (tap vs. hover).
 - Breadcrumb readability and truncation on narrow viewports.

3. Code Validation & Debugging

- **W3C Validators:**
 - HTML: <https://validator.w3.org>
 - CSS: <https://jigsaw.w3.org/css-validator>
 - Fix errors and warnings to improve robustness and accessibility.
- **Browser Dev Tools:**
 - **Console:** Catch JavaScript errors.
 - **Elements Panel:** Inspect live DOM and style overrides.
 - **Network Panel:** Ensure resources (CSS, JS) load without 404s.
 - **Accessibility Panel:** Check ARIA roles, color contrast.

Interactive Activity

1. Menu Audit:

- Provide a sample website and have students identify and critique its navigation:
 - Is the menu clear and easy to use?
 - Does it adapt well between desktop and mobile views?
- Ask them to propose one improvement for horizontal/vertical menus, dropdowns, or breadcrumbs.

2. Validation & Debugging Challenge:

- Give students a small HTML/CSS snippet with deliberate markup errors and CSS compatibility issues.
- In pairs, they must:
 - Run it through the W3C validators and note errors.
 - Use browser dev tools to inspect and fix a broken menu or misaligned button group.

4. SUMMARY AND REVIEW

Recap Key Takeaways

- **Design Planning:**
 - Use iterative design (design → implement → test) with wireframes, mock-ups, and prototypes.
 - Establish a style guide for colours, typography, and components.
- **Web Typography:**
 - Choose web-safe or custom web fonts and embed them correctly.
 - Maintain clear hierarchy with headings and spacing.
- **Accessibility:**
 - Follow WCAG principles (Perceivable, Operable, Understandable, Robust).
 - Use semantic HTML, text alternatives, sufficient colour contrast, keyboard navigation, and ARIA where needed.
- **Navigation & Usability:**
 - Design clear horizontal/vertical menus, dropdowns, breadcrumbs, and button groups.
 - Ensure menus and interactive elements work equally well on desktop and mobile.
- **Testing & Validation:**
 - Validate HTML/CSS with W3C tools.
 - Debug and test across browsers and devices using dev tools and accessibility auditors (Lighthouse, WAVE).

Quick Q&A

- **“Why is a style guide important?”**

Ensures consistency in visual language, speeds up development, and maintains brand cohesion.
- **“Name one WCAG guideline for images.”**

All images must have meaningful alt text so that screen readers can convey their purpose.

ASSESSMENT IDEAS

Formative

- **In-Class Exercise:**
 - Students create a wireframe and accompanying style guide for a given page concept.
- **Peer Review:**
 - Swap accessibility audit reports and proposed fixes, then give constructive feedback.

Summative

- **Project:**
 - Design and code a multi-section web page that incorporates design principles, typography, accessibility, and usability guidelines.
- **Written Reflection:**
 - Describe your iterative design process, challenges faced, and outcomes of usability testing.

Additional Resources

Websites

- **MDN Web Docs**
 - Accessibility: <https://developer.mozilla.org/docs/Web/Accessibility>
 - CSS: <https://developer.mozilla.org/docs/Web/CSS>
- **W3C WCAG Overview**
 <https://www.w3.org/WAI/standards-guidelines/wcag/>
- **CSS-Tricks**
 <https://css-tricks.com>

Tools

- **Prototyping:** Figma (<https://www.figma.com>) or Adobe XD (<https://www.adobe.com/products/xd.html>)
- **Accessibility Audit:** Lighthouse (built into Chrome DevTools) or WAVE (<https://wave.webaim.org>)

- **Validation:**

- HTML Validator: <https://validator.w3.org>
- CSS Validator: <https://jigsaw.w3.org/css-validator>