Parul® University

Web Designing (03010501PC01)

Unit 4 : Web Design & Advanced CSS

Parul®
University

DIGITAL LEARNING CONTENT

Responsive Web Design & Advanced CSS

- Responsive Web Design (RWD)
- Advanced CSS Techniques
- Media Queries, Transitions, Animations
- CSS Variables & Styling
- Templates & Practical Usage

Parul®
University

Why Responsive Design?

- Websites accessed on mobile, tablet, desktop
- Consistent experience across devices
- Google prioritizes mobile-friendly website
- Increases user engagement & retention
- Reduces bounce rate

Problems Without RWD

- Fixed layouts don’t fit smaller screens
- Requires zooming & horizontal scrolling
- Poor readability (tiny text/images)
- Higher bounce rate → users leave quickly
- Harder maintenance across devices

Core Principles of RWD

- **Flexible Layouts** – relative units instead of fixed px
- **Flexible Media** – images & videos scale automatically
- **Media Queries** – apply CSS rules conditionally
- Progressive enhancement for better UX
- Mobile-first design approach

Fixed vs Fluid vs Responsive

- **Fixed Layout**: Width set in px (rigid, not adaptable)
- **Fluid Layout**: Width in % (scales, but can look awkward)
- **Responsive Layout**: Combines flexible units + breakpoints
- Responsive ensures usability across all devices

Viewport Meta Tag

html:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- Instructs browser to scale page correctly
- Matches page width to device width
- Without it → mobile shows shrunken desktop view
- Essential for modern mobile design

Units for Responsiveness

- **%** → relative to parent container
- **em** → relative to parent font size
- **rem** → relative to root font size
- **vw/vh** → viewport width/height based
- Flexible units adapt better than px

Benefits of Responsive Design

- One site works everywhere → single codebase
- Improves **SEO & ranking**
- Enhances **usability & accessibility**
- Cost-effective vs multiple versions
- Improves site performance

Demo Idea

- Open page without viewport meta → looks tiny & broken
- Add meta tag → auto-fits screen width
- Show image with width:500px vs max. width:100%
- Resize browser → see flexibility
- (Diagram: before vs after viewport meta)

Parul®
University

DIGITAL LEARNING CONTENT

What are Media Queries?

- Conditional CSS rules
- Apply styles based on device width, height, orientation
- Syntax: @media (condition) { … }
- Foundation of RWD

Media Query Syntax

css:

```
@media (max-width: 768px) {
body { background: lightblue; }
}
```

- Common properties: max-width, min-width
- Multiple conditions can be combined

Common Breakpoints

- 1200px – Large screens (desktop)
- 992px – Laptops
- 768px – Tablets
- 576px – Phones
  (Diagram: device icons with breakpoints)

Mobile-First vs Desktop-First

- Mobile-first → start small, add min-width queries
- Desktop-first → start large, use max-width queries
- Modern trend → Mobile-first

Multiple Conditions

css:

```
@media (min-width: 600px) and (orientation: landscape) {
  body { background: yellow; }
}
```

Parul®
University

Responsive Text Example

- Large font on desktop
- Smaller font on mobile
- Flexible layouts

Navbar Example

- Horizontal menu on large screens
- Stacked vertical on small screens
  (Diagram: navbar two states)

Best Practices

- Keep breakpoints consistent
- Use relative units
- Test on multiple devices
- Avoid too many breakpoints

Parul®
University

Demo Idea

- Background changes at 900px
- Font size adjusts at 600px
- Navbar stacks on small screens

Beyond Screen Width

- Media queries can target:

- Orientation (portrait/landscape)
- Device resolution
- User preferences

Orientation Example

css:

```
@media (orientation: portrait) {
  body { background: lightgreen; }
}
@media (orientation: landscape) {
  body { background: lightcoral; }
}
```

Resolution Queries

- High-DPI screens (Retina)
- Example:

```
css:
@media (min-resolution: 2dppx) {
  img { border: 5px solid blue; }
}
```

Dark Mode Preference

css:

```
@media (prefers-color-scheme: dark) {
  body { background: #222; color: #fff; }
}
```

Reduced Motion Preference

css:

```
@media (prefers-reduced-motion: reduce) {
  * { animation: none !important; }
}
```

Responsive Typography

- Use clamp (min, preferred, max)

- Example:
- h1 { font-size: clamp(1.5rem, 5vw, 3rem); }

css:

```
h1 { font-size: clamp(1.5rem, 5vw, 3rem); }
```

Responsive Images

- Hide/show images depending on screen size
- Save bandwidth

Print Media Queries

css:

```
@media print {
  nav, footer { display: none; }
}
```

Parul®
University

Demo Idea

- Show orientation change
- Dark mode support
- Text resizing with vw

What is a Transition?

- Smoothly animate changes in CSS properties
- Triggered by events (hover, focus, click)
- Make UI feel modern

Transition Properties

- transition-property → which CSS property to animate
- transition-duration → how long the animation lasts
- transition-timing-function → speed curve (ease, linear, ease-in, ease-out, cubic-bezier)
- transition-delay → wait before starting

Example (Button Hover)

```
css:

button {
  transition: background 0.5s ease, transform 0.3s;
}
button:hover {
  background: blue;
  transform: scale(1.1);
}
```

Common Use Cases

- Hover effects
- Dropdown menus
- Image zoom-in/out
- Smooth color changes

Best Practices

- Don’t animate too many properties (use opacity, transform)
- Keep durations short (0.2–0.5s typical)
- Test across devices

Diagram

- (Before hover → After hover, with arrow showing transition)

Demo Idea

- Box changes color + grows smoothly on hover

Parul®
University

DIGITAL LEARNING CONTENT

What is an Animation?

- More powerful than transitions
- Define keyframes for multiple states
- Can repeat indefinitely

Key Properties

- @keyframes name
- animation-name → which animation to use
- animation-duration
- animation-timing-function
- animation-delay
- animation-iteration-count
- animation-direction

Keyframe Syntax

css:

```
@keyframes bounce {
  0%   { transform: translateY(0); }
  50%  { transform: translateY(-50px); }
  100% { transform: translateY(0); }
}
```

Example (Bouncing Ball)

- Define bounce movement with keyframes
- Apply infinite looping

Animation Directions

- normal → default
- reverse → backwards
- alternate → back and forth

Diagram

- (Ball bouncing up and down with keyframes markers)

Demo Idea

- A circle bouncing infinitely

Parul®
University

DIGITAL LEARNING CONTENT

What are CSS Variables?

- Custom properties defined with –name
- Reusable, easier theming
- Access with var(--name)

Declaring Variables

```
css:

:root {
  --main-color: teal;
  --padding: 20px;
}
```

Using Variables

```
css:

button {
  background: var(--main-color);
  padding: var(--padding);
}
```

Theming with Variables

- Dark/light themes
- Change one variable → entire UI updates

Overriding Variables

- Variables can be redefined in child selectors

Demo Idea

- Theme switcher (light/dark mode) using variables

Box Shadows

- Add depth and elevation to elements
- Syntax:

css:

```
box-shadow: offset-x offset-y blur-radius spread-radius color;
```

- Example:

css:

```
box-shadow: 2px 4px 10px rgba(0,0,0,0.3);
```

Text Shadows

- Similar syntax but for text

css:

```
text-shadow: 2px 2px 5px gray;
```

Gradients (Linear)

- Smooth color transitions

css:

```
background: linear-gradient(to right, red, blue);
```

Gradients (Radial)

css:

```
background: radial-gradient(circle, yellow, green);
```

Border Radius

- Round corners easily

css:

```
border-radius: 10px;
border-radius: 50%; /* circle */
```

Combined Styling

- Shadows + gradients + radius = modern UI
  (Diagram: card with shadow, rounded corners, gradient background)

What are Transforms?

- Modify element shape, size, position, rotation
- No layout braking

Translate

css:

transform: translate(50px, 20px);

Rotate

css:

transform: rotate(45deg);

Scale

css:

transform: scale(1.5);

Skew

css:

transform: skew(20deg, 10deg);

Combine Transforms

css:

```
transform: translate(50px) rotate(30deg) scale(1.2);
```

Parul®
University

DIGITAL LEARNING CONTENT

What are Templates?

- Pre-built HTML/CSS layouts
- Save development time
- Good for learning & rapid prototyping

Parul®
University

DIGITAL LEARNING CONTENT

Sources of Templates

- Bootstrap templates
- Tailwind starter kits
- FreeCodeCamp, W3Schools examples

Parul®
University

DIGITAL LEARNING CONTENT

How to Use Templates

- Download template (HTML/CSS files)
- Open in editor
- Customize branding, content, images

Parul®
University

DIGITAL LEARNING CONTENT

Common Template Types

- Landing page
- Portfolio
- Dashboard
- Blog

Advantages

- Responsive out of the box
- Consistent design system
- Save efforts in CSS boilerplate

Parul®
University

DIGITAL LEARNING CONTENT

Why Customize Templates?

- Branding consistency
- Unique design identify
- Adapt template to project needs

Parul®
University

Typical Customizations

- Colors (brand palette)
- Typography
- Layout changes
- Adding/removing sections

Example – Changing Theme Colors

```
css:

:root {
  --primary: #007bff;
}
button {
  background: var(--primary);
}
```

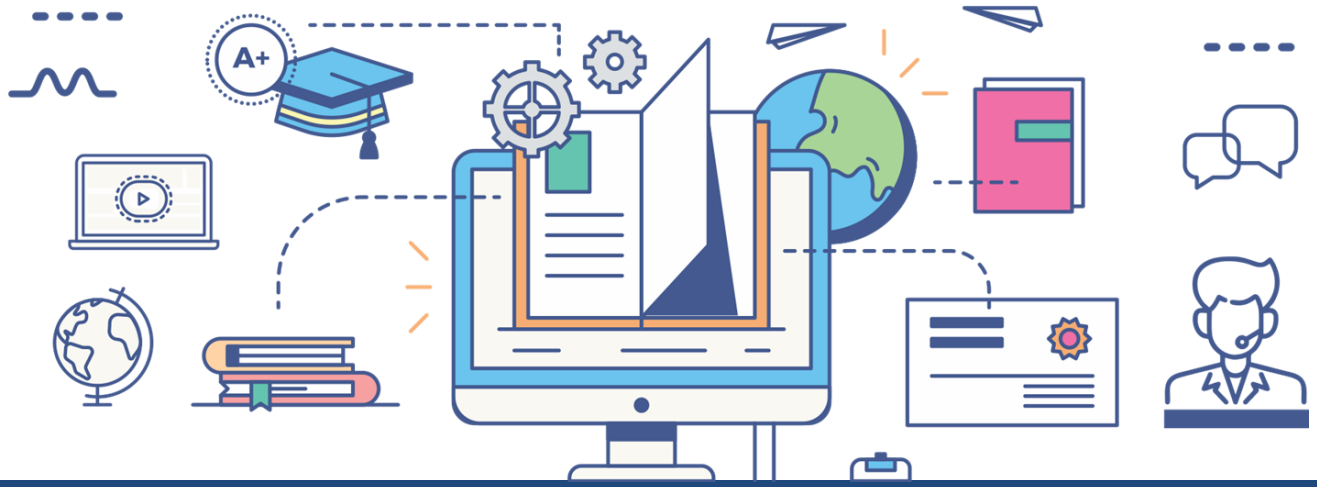
Example – Adding Custom CSS

- Override default template styles
- Use a separate stylesheet

Best Practices

- Don’t edit vendor files directly
- Keep changes modular
- Comment your overrides

DIGITAL LEARNING CONTENT



Parul® University

www.paruluniversity.ac.in