🎨 Complete CSS Units Guide

Master when to use px, %, vw, vh, rem, em and never be confused again!

1. px (Pixels)

Absolute unit - Fixed size that never changes

px Absolute

What it is: 1px = 1 pixel on the screen. Fixed size that doesn't change based on anything.

Think of it as: Drawing with a ruler - exact measurements.

Visual Demo:

200px wide × 100px tall

Try it: Resize your browser window. This box stays exactly 200px × 100px!

```
.button { width: 200px; /* Always 200 pixels */ height: 50px; /* Always 50 pixels */ border: 2px solid; /* Always 2 pixels */ }
```

✅ When to Use px:

- **Borders:** border: 1px solid #ccc; (borders should be crisp)
- **Shadows:** box-shadow: 0 2px 4px rgba(0,0,0,0.1);
- **Small icons:** width: 24px; height: 24px;
- **Fixed-width elements:** Logos, avatars, specific design elements
- **Media queries:** @media (min-width: 768px)

❌ Avoid px for:

- **Font sizes:** Use rem instead (accessibility!)
- **Responsive layouts:** Use %, vw, or fr
- **Spacing (padding/margin):** Use rem or em for scalability

2. % (Percentage)

Relative to parent element

% Relative to Parent

What it is: Size relative to the parent element.

Think of it as: "Take up X% of my parent's space"

Visual Demo:

Parent container (100% width)

I'm 50% of parent width

Try it: Resize browser. The blue box is always 50% of its parent!

```
.parent { width: 1000px; /* Parent is 1000px */ } .child { width: 50%; /* Child will be 500px (50% of 1000px) */ margin: 10%; /* Margin is 100px (10% of 1000px) */ }
```

When to Use %:

- **Layout widths:** width: 50%; (columns, grids)
- **Responsive images:** width: 100%; (fit parent)
- **Padding/margin in grids:** padding: 5%;
- **Positioning:** left: 50%; (centering)
- **Fluid layouts:** Container-based sizing

Avoid % for:

- **Font sizes:** Compounds with nested elements (confusing!)
- **When you need viewport-relative:** Use vw/vh instead

Important: Height in % only works if parent has defined height!

3. vw (Viewport Width)

Relative to browser window width

VW

Viewport Relative

What it is: 1vw = 1% of viewport (browser window) width.

Math: If browser is 1200px wide, 50vw = 600px

Think of it as: "Take up X% of the entire screen width"

Visual Demo:

I'm 50vw (50% of viewport width)

Try it: Resize browser. This is ALWAYS 50% of window width (not parent)!

```
.hero-section { width: 100vw; /* Full screen width */ height: 100vh; /* Full screen height */ } .heading { font-size: 5vw; /* Scales with screen */ }
```

When to Use vw:

- **Full-width sections:** width: 100vw;
- **Responsive typography:** font-size: 5vw; (hero headings)
- **Break out of container:** margin-left: -20vw;
- **Consistent spacing:** gap: 2vw;

- **Landing page layouts:** Viewport-aware designs

❌ **Avoid vw for:**

- **Body text:** Too small on mobile, too big on desktop
- **Buttons/small elements:** They'll shrink too much on mobile
- **When parent-relative is needed:** Use % instead

💡 **Pro Tip:** Use clamp() for responsive typography: font-size: clamp(16px, 3vw, 48px);

4. vh (Viewport Height)

Relative to browser window height

vh

Viewport Relative

What it is: 1vh = 1% of viewport (browser window) height.

Math: If browser is 800px tall, 50vh = 400px

Think of it as: "Take up X% of the entire screen height"

Visual Demo:

I'm 20vh (20% of viewport height)

Try it: Resize browser vertically. This box changes height!

```
.hero { height: 100vh; /* Full screen height */ } .section { min-height: 80vh; /* At least 80% of screen */ } .modal { max-height: 90vh; /* Max 90% of screen */ }
```

✅ **When to Use vh:**

- **Hero sections:** height: 100vh; (full screen)
- **Modals/overlays:** max-height: 90vh;
- **Sticky sections:** min-height: 100vh;
- **Vertical spacing:** margin-top: 10vh;
- **Split-screen layouts:** height: 50vh each

❌ **Avoid vh for:**

- **Mobile (sometimes):** 100vh includes address bar (use svh/dvh if available)
- **Text containers:** Height should be content-driven
- **Small elements:** Buttons, inputs, etc.

⚠️ **Mobile Issue:** 100vh on mobile includes the URL bar, which disappears when scrolling. Consider using min-height: 100vh or newer units like svh (small viewport height).

5. rem (Root EM)

Relative to root font size (html element)

rem <kbd>Root Relative</kbd>

What it is: Relative to the root (html) element's font-size.

Default: 1rem = 16px (browser default)

Think of it as: "Scale with the base font size"

Visual Demo:

Root font-size: 16px (default)

This text is 2rem (32px)

```
html { font-size: 16px; /* 1rem = 16px */ } .text { font-size: 2rem; /* 2 × 16px = 32px */ padding: 1rem; /* 16px */ margin: 0.5rem; /* 8px */ }
```

When to Use rem:

- **Font sizes:** font-size: 1.5rem; (scales with user preferences!)
- **Spacing:** padding: 2rem; margin: 1rem; (consistent scaling)
- **Component sizing:** width: 20rem; (scales proportionally)
- **Media queries:** @media (min-width: 48rem) (accessible)
- **Border radius:** border-radius: 0.5rem;

Avoid rem for:

- **Borders:** Use px (1px borders should stay 1px)
- **Shadows:** Use px for precise control
- **When nesting matters:** Use em instead

REM Best Practice

✓ Set html { font-size: 62.5%; } makes 1rem = 10px (easier math!)

✓ Then: 1.6rem = 16px, 2.4rem = 24px, etc.

```
html { font-size: 62.5%; /* 1rem = 10px */ } body { font-size: 1.6rem; /* 16px */ } h1 { font-size: 3.2rem; /* 32px - Easy to calculate! */ }
```

Accessibility Win: If a user increases their browser's default font size from 16px to 20px, everything sized in rem will scale up automatically!

6. em (Relative to Parent)

Relative to parent element's font size

em <kbd>Parent Relative</kbd>

What it is: Relative to the parent element's font-size.

Warning: Compounds in nested elements!

Think of it as: "Scale based on my parent's font size"

Visual Demo:

Parent font-size: 20px

This is 1.5em (30px = 1.5 × 20px)

```
.parent { font-size: 20px; /* Parent is 20px */ } .child { font-size: 1.5em; /* 1.5 × 20px = 30px */ padding: 1em; /* Based on CHILD'S font (30px) = 30px */ }
```

When to Use em:

- **Component padding/margin:** padding: 0.5em; (scales with font)
- **Button sizing:** padding: 0.5em 1em; (proportional to text)
- **Icon sizes:** width: 1em; (same size as text)
- **Line-height:** line-height: 1.5em;
- **Self-contained components:** Everything scales together

Avoid em for:

- **Deeply nested elements:** Compounds! (1.5em × 1.5em × 1.5em = chaos)
- **Global spacing:** Use rem instead
- **When you want consistency:** Use rem

Compounding Example:

```
.parent { font-size: 1.5em; } /* 24px (1.5 × 16) */ .child { font-size: 1.5em; } /* 36px (1.5 × 24) - COMPOUNDS! */ .grandchild { font-size: 1.5em; } /* 54px (1.5 × 36) - Gets huge! */
```

This is why **rem is usually better for font sizes!**

Quick Comparison Table

All units side by side

Unit	Type	Relative To	Best For	Example
px	Absolute	Nothing (fixed)	Borders, shadows, icons	border: 1px solid;
%	Relative	Parent element	Layout widths, responsive images	width: 50%;
vw	Viewport	Viewport width	Full-width sections, responsive text	width: 100vw;
vh	Viewport	Viewport height	Hero sections, modals	height: 100vh;
rem	Relative	Root (html) font-size	Font sizes, spacing, components	font-size: 1.5rem;
em	Relative	Parent font-size	Component padding, icons	padding: 1em;

Decision Tree: Which Unit Should I Use?

Follow this guide to never be confused again!

🤔 Start Here: What are you styling?

📝 Font Sizes

→ Use **rem** (accessible, scales with user preferences)
→ Avoid: px, em (compounds), % (confusing)

📏 Spacing (padding, margin, gap)

→ Use **rem** (consistent, scalable)
→ Alternative: em for component-specific spacing
→ Avoid: px (not scalable), % (parent-dependent)

📐 Layout Widths

→ Container-relative? Use **%**
→ Full screen width? Use **100vw**
→ Fixed width? Use **px** or **rem**
→ Grid columns? Use **fr** units

📏 Layout Heights

→ Full screen? Use **100vh**
→ Relative to parent? Use **%** (parent must have height!)
→ Content-based? Use **auto** (don't set height)

🖼️ Borders & Shadows

→ Always use **px** (need precision)
→ Example: border: 1px solid; box-shadow: 0 2px 4px;

🎨 Border Radius

→ Use **rem** (scales with design)
→ Alternative: px for small, fixed corners

🔘 Buttons

→ Font: **rem**
→ Padding: **em** (scales with button text)
→ Min-width: **rem**

📱 Media Queries

→ Use **rem** or **em** (accessible)
→ @media (min-width: 48rem) /* 768px */
→ Avoid: px (doesn't respect user font size)

🌎 Real-World Example: Complete Component

See all units working together

```
/* ===== ROOT SETUP ===== */ html { font-size: 62.5%; /* 1rem = 10px for easy math */ } body { font-size: 1.6rem; /* 16px - REM for accessibility */ margin: 0; padding: 2rem; /* 20px - REM for consistent spacing */ } /* ===== HERO SECTION ===== */ .hero { height: 100vh; /* VH - Full viewport height */ width: 100vw; /* VW - Full viewport width */ padding: 4rem; /* REM - Scalable spacing */ margin-left: -2rem; /* REM - Break out of body padding */ } .hero h1 { font-size: clamp(3rem, 5vw, 6rem); /* VW with limits! */ margin-bottom: 2rem; /* REM - Consistent spacing */ } /* ===== CONTAINER ===== */ .container { max-width: 120rem; /* REM - 1200px, scales with zoom */ margin: 0 auto; padding: 0 2rem; /* REM - Consistent spacing */ } /* ===== GRID LAYOUT ===== */ .grid { display: grid; grid-template-columns: repeat(12, 1fr); /* FR - Equal columns */ gap: 2rem; /* REM - Scalable gaps */ } .card { grid-column: span 4; /* Span 4 columns (33%) */ padding: 2rem; /* REM - Consistent spacing */ border: 1px solid #ccc; /* PX - Crisp border */ border-radius: 0.8rem; /* REM - Scalable corners */ box-shadow: 0 2px 8px rgba(0,0,0,0.1); /* PX - Precise shadow */ } /* ===== BUTTON ===== */ .button { font-size: 1.6rem; /* REM - Accessible font size */ padding: 0.75em 1.5em; /* EM - Scales with button text */ border: 2px solid; /* PX - Crisp border */
```

```
border-radius: 0.5rem; /* REM - Consistent corners */ min-width: 12rem; /* REM - Minimum width */ } /* ===== IMAGE ===== */ .image { width: 100%; /* % - Fill parent */ height: auto; /* AUTO - Maintain aspect ratio */ border-radius: 0.8rem; /* REM - Consistent corners */ } /* ===== MODAL ===== */ .modal { position: fixed; top: 50%; /* % - Center vertically */ left: 50%; /* % - Center horizontally */ transform: translate(-50%, -50%); width: 90vw; /* VW - Responsive width */ max-width: 60rem; /* REM - Max width limit */ max-height: 90vh; /* VH - Don't overflow screen */ padding: 3rem; /* REM - Consistent spacing */ } /* ===== RESPONSIVE ===== */ @media (min-width: 48rem) { /* REM - Accessible breakpoint (768px) */ .card { grid-column: span 4; /* 3 columns on tablet+ */ } } @media (min-width: 64rem) { /* REM - Accessible breakpoint (1024px) */ .card { grid-column: span 3; /* 4 columns on desktop */ } }
```

✨ Why This Works

- ✓ **px** for borders, shadows (need precision)
- ✓ **rem** for fonts, spacing, components (accessible + scalable)
- ✓ **em** for button padding (scales with button text)
- ✓ **%** for layout positioning, image widths (parent-relative)
- ✓ **vw/vh** for hero sections, modals (viewport-aware)
- ✓ **fr** for grid columns (flexible layouts)

🎯 Final Recommendations (TL;DR)

The golden rules to remember

px	rem	em	%
Borders, shadows, small icons, precise measurements	Font sizes, spacing, most sizing (DEFAULT CHOICE)	Component padding that scales with text	Layout widths, parent-relative sizing

vw	vh
Full-width sections, responsive typography	Hero sections, full-height layouts

🏆 Best Practices Summary

- ✓ Default to **rem** for almost everything
- ✓ Use **px** only for borders, shadows, and fine details
- ✓ Use **em** for padding in buttons/components
- ✓ Use **%** for responsive layouts relative to parent
- ✓ Use **vw/vh** for viewport-aware designs
- ✓ Set html { font-size: 62.5%; } for easier rem calculations
- ✓ Use media queries in **rem** for accessibility
- ✓ Combine units: clamp(1.6rem, 3vw, 4rem) for responsive text