

Fluid Font Forge User Manual

Version 5.3.0 Professional Responsive Typography for WordPress

Table of Contents

- 1. [Introduction](#)
 - 2. [What is Fluid Font Forge?](#)
 - 3. [Key Features](#)
 - 4. [Getting Started](#)
 - 5. [Use Case 1: Client Font Consultation](#)
 - 6. [Use Case 2: CSS Classes for Component Library](#)
 - 7. [Use Case 3: CSS Custom Properties for Design System](#)
 - 8. [Use Case 4: Tailwind Configuration Integration](#)
 - 9. [Use Case 5: Semantic HTML Tag Styling](#)
 - 10. [Use Case 6: Streamlined Workflow with Enhanced UI](#)
 - 11. [Advanced Tips](#)
 - 12. [Troubleshooting](#)
-

Introduction

Fluid Font Forge transforms the way you implement responsive typography in WordPress. Instead of managing multiple breakpoints and media queries, this plugin generates mathematically precise CSS `clamp()` functions that scale fonts smoothly across all viewport sizes.

This manual guides you through the plugin's features and demonstrates real-world use cases that showcase how Fluid Font Forge solves common typography challenges in modern web development.

What is Fluid Font Forge?

Fluid Font Forge is a WordPress plugin that generates responsive typography using CSS `clamp()` functions. The plugin calculates fluid font scaling based on:

- **Viewport range:** Minimum and maximum screen widths
- **Base font sizes:** Root font sizes at viewport limits
- **Scaling ratios:** Musical harmony ratios (Minor Second to Golden Ratio)
- **Mathematical precision:** Linear interpolation for smooth transitions

The Science Behind It

The plugin uses the CSS `clamp()` function:

```
font-size: clamp(min_size, preferred_size, max_size);
```

Where `preferred_size` is calculated using linear interpolation:

```
preferred_size = min_size + (viewport_width - min_viewport) * growth_rate
```

This creates fonts that scale smoothly without breakpoints, providing an optimal reading experience at every screen size.

Key Features

Mathematical Typography Scaling

Uses musical harmony ratios for natural size progression:

- **Minor Second (1.067)**: Subtle, tight spacing
- **Major Second (1.125)**: Clean, modern designs
- **Minor Third (1.200)**: Balanced hierarchy
- **Major Third (1.250)**: Strong contrast
- **Perfect Fourth (1.333)**: Bold, dramatic scaling

Multiple Output Formats

Generate CSS in four formats:

- **CSS Classes**: `.large`, `.medium`, `.small`
- **CSS Custom Properties**: `--fs-lg`, `--fs-md`
- **HTML Tag Styling**: `h1`, `h2`, `p`
- **Tailwind Configuration**: Direct `tailwind.config.js` integration

Real-Time Preview

- Side-by-side comparison at min/max viewports
- Interactive viewport slider showing real-time scaling
- Separate preview controls for titles and body text
- Custom font loading for accurate client previews

Professional Workflow Tools

- **Drag & Drop Management**: Reorder font sizes intuitively
 - **Skip Entries Feature**: Exclude font sizes from CSS output while maintaining scale progression for custom spacing control
 - **Copy-to-Clipboard**: One-click CSS copying
 - **Autosave System**: Automatic saving with visual feedback
 - **Enhanced Data Table**: Add, edit, delete sizes with validation
-

Getting Started

Installation

1. Upload **fluid-font-forge** folder to `/wp-content/plugins/`
2. Activate through **Plugins** menu in WordPress
3. Navigate to **Tools** → **Fluid Font Forge**

Initial Configuration

1. **Select Font Units:** Choose between px (predictable) or rem (accessible)
2. **Set Viewport Range:** Default is 375px (mobile) to 1620px (desktop)
3. **Configure Base Font Sizes:** Default is 16px at minimum, 20px at maximum
4. **Choose Scaling Ratios:** Select ratios for small and large screens

The plugin provides sensible defaults, so you can start experimenting immediately.

Use Case 1: Client Font Consultation

The Scenario

You're a web designer meeting with a client who wants to see how different Google Fonts will look at various sizes across different devices. They're concerned about readability on mobile phones and visual impact on large desktop monitors.

The Challenge

Traditional mockups show fonts at fixed sizes on specific devices. Clients struggle to visualize how typography scales across their users' diverse devices. You need a way to demonstrate fluid typography behavior in real-time during your consultation.

The Solution with Fluid Font Forge

Step 1: Load the Client's Font Choice

1. Visit [Google Fonts](#) and select the font (e.g., "Playfair Display")
2. Click "View selected families" and copy the `<link>` href URL:

```
https://fonts.googleapis.com/css2?
family=Playfair+Display:wght@400;700&display=swap
```

3. Paste this URL into the **Preview Font** field in Fluid Font Forge
4. The font loads immediately in all preview areas

Step 2: Configure for Client's Target Devices

Based on your client's analytics:

- **Min Viewport Width:** 375px (iPhone SE)
- **Max Viewport Width:** 1920px (their office monitors)
- **Min Root Size:** 16px (mobile readability)
- **Max Root Size:** 20px (desktop comfort)

Step 3: Set Up Comparison Points

In the **Sample Text** panel:

1. Set **Titles base** to a larger size (e.g., "xxl" or "4xl")
2. Set **Text base** to body text size (e.g., "base" or "md")
3. Use the viewport slider to demonstrate scaling from 375px to 1920px

Step 4: Interactive Consultation

Move the **Viewport Size** slider while the client watches:

- At 375px: "This is how your headline reads on an iPhone"
- At 768px: "Here's the tablet experience"
- At 1920px: "And this is what visitors see on large monitors"

The client sees titles and body text scaling in real-time, making abstract concepts concrete.

Step 5: Explore Font Scale Options

Show different visual hierarchies:

1. Click **Settings** panel
2. Change **Min Viewport Font Scaling** to "Minor Third (1.200)"
3. Change **Max Viewport Font Scaling** to "Perfect Fourth (1.333)"
4. Client immediately sees more dramatic size differences on large screens

Step 6: Compare Multiple Fonts

Try different fonts during the meeting:

1. Test "Montserrat" for modern tech feel
2. Test "Merriweather" for editorial elegance
3. Test "Inter" for clean UI approach
4. Client can compare how each font behaves at their actual device sizes

What Was Gained

For the Designer:

- Eliminated need for multiple static mockups
- Demonstrated actual fluid behavior clients will experience
- Made technical concepts (viewport units, clamp) visually understandable
- Shortened decision-making time from days to minutes

For the Client:

- Saw real fonts at real sizes on real device widths
- Understood how responsive typography works
- Made informed decisions based on actual behavior
- Gained confidence in the design approach

Technical Benefits:

- Font URL testing before implementation
- Immediate validation of scaling ratios
- Zero code required during exploration phase
- Client signs off on exact scaling parameters used in production

Pro Tips for Client Presentations

Note: The following features are planned for a future Pro version:

1. **Prepare Multiple Scenarios:** Save different configurations before the meeting (*Coming Soon*)
 2. **Use Client's Content:** Replace sample text with actual headlines from their business (*Coming Soon*)
 3. **Document Decisions:** Take screenshots at key viewport widths for reference
 4. **Export Settings:** Copy the final CSS for immediate implementation
-

Use Case 2: CSS Classes for Component Library

The Scenario

You're building a component library for a WordPress theme that will be used by multiple developers. You need a consistent set of utility classes for typography that work across all viewport sizes without complex media queries.

The Challenge

You want classes like `.heading-large`, `.body-text`, `.caption` that:

- Scale smoothly across devices
- Maintain consistent hierarchy
- Are easy for developers to remember and use
- Don't require additional breakpoint management

The Solution with Fluid Font Forge

Step 1: Plan Your Scale

Your component library needs:

- 5 heading sizes (h1-h5 equivalents)
- 3 body text sizes (large, normal, small)
- 2 utility sizes (caption, label)

Total: 10 distinct font sizes

Step 2: Configure the Settings

1. **Font Units:** Select **REM** for accessibility
2. **Viewport Range:**
 - Min: 375px (mobile-first)

- Max: 1440px (standard desktop)
- 3. **Base Sizes:**
 - Min Root: 16px
 - Max Root: 18px (subtle scaling)
- 4. **Scaling Ratios:**
 - Min Scale: Major Second (1.125) - gentle mobile scaling
 - Max Scale: Minor Third (1.200) - clear desktop hierarchy

Step 3: Build Your Font Sizes in Groups

Important: You'll need to build your font sizes in separate groups since they use different scales and base values. Each group will be generated separately, then combined into your final CSS.

Group 1: Headings (5 levels) Set scaling for dramatic hierarchy:

- **Min Scale:** Major Third (1.250) - stronger mobile contrast
- **Max Scale:** Perfect Fourth (1.333) - bold desktop hierarchy

In the **Data Table**, create heading sizes:

Name	Class	Min Size	Max Size	Line Height	Notes
Heading 1	h1	2.488rem	3.583rem	1.1	Page titles
Heading 2	h2	2.074rem	2.986rem	1.2	Section headers
Heading 3	h3	1.728rem	2.488rem	1.3	Subsections
Heading 4	h4	1.440rem	2.074rem	1.4	Card titles
Heading 5	h5	1.200rem	1.728rem	1.5	Labels

Generate and copy this CSS, then **clear the data table**.

Group 2: Body Text (3 sizes) Adjust for readability:

- **Min Scale:** Minor Second (1.067) - subtle mobile differences
- **Max Scale:** Major Second (1.125) - gentle desktop scaling

Create body text sizes:

Name	Class	Min Size	Max Size	Line Height	Notes
Body Large	body-lg	1.125rem	1.440rem	1.6	Intro text
Body	body	1.000rem	1.200rem	1.6	Main content
Body Small	body-sm	0.889rem	1.000rem	1.5	Secondary text

Generate and copy this CSS, then **clear the data table** again.

Group 3: Utility Sizes (2 sizes) For UI elements with minimal scaling:

- **Min Scale:** Minor Second (1.067)

- **Max Scale:** Minor Second (1.067) - consistent scaling

Create utility sizes:

Name	Class	Min Size	Max Size	Line Height	Notes
Caption	caption	0.790rem	0.833rem	1.4	Image captions
Label	label	0.702rem	0.694rem	1.3	Form labels

Generate and copy this final CSS.

Step 4: Combine Generated CSS from All Groups

Now you have three sets of CSS copied from Fluid Font Forge. Combine them into your component library stylesheet.

1. Click the **Class** tab in the header (if not already selected)
2. You should have copied CSS from all three groups above

Group 1 CSS (Headings):

```
.h1 {
  font-size: clamp(2.488rem, 1.863rem + 3.13vw, 3.583rem);
  line-height: 1.1;
}

.h2 {
  font-size: clamp(2.074rem, 1.617rem + 2.29vw, 2.986rem);
  line-height: 1.2;
}

.h3 {
  font-size: clamp(1.728rem, 1.401rem + 1.64vw, 2.488rem);
  line-height: 1.3;
}

.h4 {
  font-size: clamp(1.440rem, 1.208rem + 1.16vw, 2.074rem);
  line-height: 1.4;
}

.h5 {
  font-size: clamp(1.200rem, 1.034rem + 0.83vw, 1.728rem);
  line-height: 1.5;
}
```

Group 2 CSS (Body Text):

```

.body-lg {
  font-size: clamp(1.125rem, 0.969rem + 0.78vw, 1.440rem);
  line-height: 1.6;
}

.body {
  font-size: clamp(1.000rem, 0.900rem + 0.50vw, 1.200rem);
  line-height: 1.6;
}

.body-sm {
  font-size: clamp(0.889rem, 0.834rem + 0.28vw, 1.000rem);
  line-height: 1.5;
}

```

Group 3 CSS (Utility):

```

.caption {
  font-size: clamp(0.790rem, 0.769rem + 0.11vw, 0.833rem);
  line-height: 1.4;
}

.label {
  font-size: clamp(0.702rem, 0.706rem + -0.02vw, 0.694rem);
  line-height: 1.3;
}

```

Step 5: Integrate Combined CSS into Your Component Library

Create `typography.css` in your theme and combine all three CSS groups:

```

/**
 * Component Library Typography
 * Generated by Fluid Font Forge in three separate groups
 * Group 1: Headings (Major Third → Perfect Fourth scaling)
 * Group 2: Body Text (Minor Second → Major Second scaling)
 * Group 3: Utility (Minimal scaling)
 * DO NOT EDIT - Regenerate from source
 */

/* === Heading Classes (Group 1) === */
.h1 {
  font-size: clamp(2.488rem, 1.863rem + 3.13vw, 3.583rem);
  line-height: 1.1;
}

.h2 {
  font-size: clamp(2.074rem, 1.617rem + 2.29vw, 2.986rem);

```



```
    line-height: 1.2;
  }

.h3 {
  font-size: clamp(1.728rem, 1.401rem + 1.64vw, 2.488rem);
  line-height: 1.3;
}

.h4 {
  font-size: clamp(1.440rem, 1.208rem + 1.16vw, 2.074rem);
  line-height: 1.4;
}

.h5 {
  font-size: clamp(1.200rem, 1.034rem + 0.83vw, 1.728rem);
  line-height: 1.5;
}

/* === Body Text Classes (Group 2) === */
.body-lg {
  font-size: clamp(1.125rem, 0.969rem + 0.78vw, 1.440rem);
  line-height: 1.6;
}

.body {
  font-size: clamp(1.000rem, 0.900rem + 0.50vw, 1.200rem);
  line-height: 1.6;
}

.body-sm {
  font-size: clamp(0.889rem, 0.834rem + 0.28vw, 1.000rem);
  line-height: 1.5;
}

/* === Utility Classes (Group 3) === */
.caption {
  font-size: clamp(0.790rem, 0.769rem + 0.11vw, 0.833rem);
  line-height: 1.4;
}

.label {
  font-size: clamp(0.702rem, 0.706rem + -0.02vw, 0.694rem);
  line-height: 1.3;
}

/* === Component-specific overrides === */
.card-title {
  @apply h4;
  font-weight: 600;
}

.hero-title {
  @apply h1;
  font-weight: 700;
}
```

```
letter-spacing: -0.02em;  
}
```

Step 6: Document for Your Team

Create usage documentation explaining the grouped approach:

Typography Components

Generation Strategy

Our typography was generated in three groups to achieve different scaling behaviors:

- **Headings**: Aggressive scaling (1.250 → 1.333) for clear hierarchy
- **Body Text**: Gentle scaling (1.067 → 1.125) for comfortable reading
- **Utility**: Minimal scaling for consistent UI elements

Heading Classes

- `.h1` - Page titles, hero sections (39.81px - 57.33px)
- `.h2` - Section headers (33.18px - 47.75px)
- `.h3` - Subsections (27.65px - 39.81px)
- `.h4` - Card titles (23.04px - 33.18px)
- `.h5` - Component labels (19.20px - 27.65px)

Body Text Classes

- `.body-lg` - Introduction paragraphs, featured content
- `.body` - Standard paragraph text
- `.body-sm` - Secondary information, metadata

Utility Classes

- `.caption` - Image captions, footnotes
- `.label` - Form labels, small UI text

Usage Example

```
```html  
<article class="post-card">
 <h2 class="h3">Article Title</h2>
 <p class="body">Article excerpt goes here...</p>
 Published 3 days ago
</article>
```

## Regeneration Workflow

When updating typography:

1. Open Fluid Font Forge
2. Generate Group 1 (Headings) with appropriate scale
3. Copy CSS, clear table
4. Generate Group 2 (Body Text) with different scale
5. Copy CSS, clear table

6. Generate Group 3 (Utility) with minimal scale
7. Combine all three CSS blocks into typography.css

#### #### Step 7: Create Helper Component

Add a WordPress shortcode for developers:

```
```php
/**
 * Typography size reference shortcode
 * Shows all available classes with examples
 */
function render_typography_reference() {
    $sizes = [
        'h1' => 'Heading 1 - Page Titles',
        'h2' => 'Heading 2 - Sections',
        'h3' => 'Heading 3 - Subsections',
        'h4' => 'Heading 4 - Cards',
        'h5' => 'Heading 5 - Labels',
        'body-lg' => 'Body Large - Introductions',
        'body' => 'Body - Standard Text',
        'body-sm' => 'Body Small - Secondary',
        'caption' => 'Caption - Annotations',
        'label' => 'Label - UI Elements'
    ];

    ob_start();
    ?>
    <div class="typography-reference">
        <?php foreach ($sizes as $class => $description): ?>
            <div class="type-sample">
                <code><?php echo esc_html($class); ?></code>
                <p class="<?php echo esc_attr($class); ?>">
                    <?php echo esc_html($description); ?>
                </p>
            </div>
        <?php endforeach; ?>
    </div>
    <?php
    return ob_get_clean();
}
add_shortcode('typography_reference', 'render_typography_reference');
```

What Was Gained

For the Project:

- Consistent typography across entire theme
- Zero breakpoint management required
- Self-documenting class names

- Easy to extend with new sizes

For Developers:

- Simple, memorable class names
- No calculations needed
- Works everywhere without modification
- Visual reference via shortcode

Technical Benefits:

- Reduced CSS bundle size (no media queries)
- Improved performance (fewer style recalculations)
- Better accessibility (rem-based)
- Maintainable single source of truth

Code Quality:

- Single responsibility (each class = one size)
- Predictable behavior
- Easy to test across viewports
- Version controllable

Maintenance Workflow

When typography needs adjustment:

1. Open Fluid Font Forge
2. **For Group 1 (Headings):**
 - Set viewport range and scaling ratios for headings
 - Add heading sizes to data table
 - Generate CSS and copy
 - Clear data table
3. **For Group 2 (Body Text):**
 - Adjust scaling ratios for body text (usually gentler)
 - Add body text sizes to data table
 - Generate CSS and copy
 - Clear data table
4. **For Group 3 (Utility):**
 - Set minimal or no scaling
 - Add utility sizes to data table
 - Generate CSS and copy
5. Combine all three CSS blocks in `typography.css`
6. Version bump and document changes
7. Deploy with confidence - all components update automatically

Why Multiple Groups? Different content types need different scaling behaviors. Headings benefit from dramatic scaling to establish hierarchy, while body text needs subtle scaling for comfortable reading. Generating them separately gives you precise control over each group's behavior.

Use Case 3: CSS Custom Properties for Design System

The Scenario

You're architecting a design system for a large WordPress site with multiple custom post types, page templates, and dynamic content. You need maximum flexibility where different components can use the same underlying font sizes but with different treatments (color, weight, transforms).

The Challenge

CSS classes are too rigid. You want:

- Semantic sizing tokens (not `.large`, but `--fs-heading-primary`)
- The ability to compose styles dynamically
- Easy theming (light/dark mode adjustments)
- Integration with CSS-in-JS if needed
- Support for design tokens in Figma

The Solution with Fluid Font Forge

Step 1: Design Your Token System

Plan your custom property naming:

- `--fs-display-*`: Extra large, attention-grabbing text
- `--fs-heading-*`: Standard heading hierarchy
- `--fs-body-*`: Body text variations
- `--fs-ui-*`: Interface elements

Step 2: Configure for Maximum Flexibility

1. **Font Units: REM** for design system consistency
2. **Viewport Range:**
 - Min: 360px (include small phones)
 - Max: 1920px (cover ultra-wide monitors)
3. **Base Sizes:**
 - Min Root: 16px (accessibility baseline)
 - Max Root: 20px (comfortable reading)
4. **Scaling Ratios:**
 - Min Scale: Major Second (1.125)
 - Max Scale: Major Third (1.250) - stronger hierarchy on large screens

Step 3: Create Design Token Scale in Multiple Groups

Since different content types need different scaling behaviors, you'll generate your tokens in separate groups, then combine them.

Group 1: Display Sizes (Extra-large attention-grabbing text) Configure for dramatic scaling:

- **Min Scale:** Major Third (1.250)

- **Max Scale:** Golden Ratio (1.618) - very dramatic on large screens

Build in the **Data Table**:

Name	CSS Variable	Min	Max	Line Height	Use Case
Display XL	display-xl	4.209rem	7.629rem	1.0	Hero headlines
Display L	display-l	3.518rem	6.103rem	1.0	Landing sections
Display M	display-m	2.931rem	4.883rem	1.1	Feature callouts

Generate CSS, copy it, then **clear the data table**.

Group 2: Heading Hierarchy (Standard heading progression) Adjust for strong but readable hierarchy:

- **Min Scale:** Major Second (1.125)
- **Max Scale:** Major Third (1.250)

Build in the **Data Table**:

Name	CSS Variable	Min	Max	Line Height	Use Case
Heading 1	heading-1	2.441rem	3.906rem	1.2	Page titles
Heading 2	heading-2	2.034rem	3.125rem	1.3	Major sections
Heading 3	heading-3	1.695rem	2.500rem	1.4	Subsections
Heading 4	heading-4	1.413rem	2.000rem	1.5	Minor headings

Generate CSS, copy it, then **clear the data table**.

Group 3: Body Text (Content reading sizes) Configure for comfortable reading with minimal scaling:

- **Min Scale:** Minor Second (1.067)
- **Max Scale:** Minor Second (1.067) - consistent across devices

Build in the **Data Table**:

Name	CSS Variable	Min	Max	Line Height	Use Case
Body XL	body-xl	1.266rem	1.600rem	1.6	Lead paragraphs
Body L	body-l	1.125rem	1.280rem	1.6	Emphasized text
Body Base	body-base	1.000rem	1.000rem	1.6	Standard text
Body S	body-s	0.889rem	0.800rem	1.5	Captions

Generate CSS, copy it, then **clear the data table**.

Group 4: UI Elements (Interface component sizing) Configure for subtle UI scaling:

- **Min Scale:** Minor Second (1.067)
- **Max Scale:** Minor Second (1.067)

Build in the **Data Table**:

Name	CSS Variable	Min	Max	Line Height	Use Case
UI L	ui-l	1.000rem	1.125rem	1.4	Buttons, large UI
UI Base	ui-base	0.889rem	0.900rem	1.4	Standard UI
UI S	ui-s	0.790rem	0.720rem	1.3	Small controls

Generate CSS and copy it.

Step 4: Combine Generated CSS Variables from All Groups

Now combine the CSS from all four groups you generated:

- 1. Click the **Variables** tab
- 2. Combine the CSS variables from each group:

```
:root {
  /* Display Sizes (Group 1) - Dramatic scaling */
  --fs-display-xl: clamp(4.209rem, 2.331rem + 9.39vw, 7.629rem);
  --fs-display-l: clamp(3.518rem, 2.143rem + 6.88vw, 6.103rem);
  --fs-display-m: clamp(2.931rem, 1.969rem + 4.81vw, 4.883rem);

  /* Heading Hierarchy (Group 2) - Strong hierarchy */
  --fs-heading-1: clamp(2.441rem, 1.749rem + 3.46vw, 3.906rem);
  --fs-heading-2: clamp(2.034rem, 1.557rem + 2.39vw, 3.125rem);
  --fs-heading-3: clamp(1.695rem, 1.381rem + 1.57vw, 2.500rem);
  --fs-heading-4: clamp(1.413rem, 1.223rem + 0.95vw, 2.000rem);

  /* Body Text (Group 3) - Comfortable reading */
  --fs-body-xl: clamp(1.266rem, 1.099rem + 0.84vw, 1.600rem);
  --fs-body-l: clamp(1.125rem, 1.042rem + 0.42vw, 1.280rem);
  --fs-body-base: 1.000rem;
  --fs-body-s: clamp(0.889rem, 0.934rem + -0.23vw, 0.800rem);

  /* UI Elements (Group 4) - Subtle scaling */
  --fs-ui-l: clamp(1.000rem, 0.938rem + 0.31vw, 1.125rem);
  --fs-ui-base: clamp(0.889rem, 0.883rem + 0.03vw, 0.900rem);
  --fs-ui-s: clamp(0.790rem, 0.832rem + -0.21vw, 0.720rem);
}
```

- 3. Copy this combined CSS to your design system

Step 5: Build Your Design System Structure

Create `design-tokens.css`:

```

/**
 * Design System - Typography Tokens
 * Generated by Fluid Font Forge in four separate groups
 * Group 1: Display sizes with dramatic scaling
 * Group 2: Heading hierarchy with strong scaling
 * Group 3: Body text with comfortable reading scale
 * Group 4: UI elements with subtle scaling
 * Base: 16px-20px across 360px-1920px viewports
 */

/* Base Font Size Tokens - Combined from all groups */
:root {
  /* Display Sizes (Group 1) - Dramatic scaling */
  --fs-display-xl: clamp(4.209rem, 2.331rem + 9.39vw, 7.629rem);
  --fs-display-l: clamp(3.518rem, 2.143rem + 6.88vw, 6.103rem);
  --fs-display-m: clamp(2.931rem, 1.969rem + 4.81vw, 4.883rem);

  /* Heading Hierarchy (Group 2) - Strong hierarchy */
  --fs-heading-1: clamp(2.441rem, 1.749rem + 3.46vw, 3.906rem);
  --fs-heading-2: clamp(2.034rem, 1.557rem + 2.39vw, 3.125rem);
  --fs-heading-3: clamp(1.695rem, 1.381rem + 1.57vw, 2.500rem);
  --fs-heading-4: clamp(1.413rem, 1.223rem + 0.95vw, 2.000rem);

  /* Body Text (Group 3) - Comfortable reading */
  --fs-body-xl: clamp(1.266rem, 1.099rem + 0.84vw, 1.600rem);
  --fs-body-l: clamp(1.125rem, 1.042rem + 0.42vw, 1.280rem);
  --fs-body-base: 1.000rem;
  --fs-body-s: clamp(0.889rem, 0.934rem + -0.23vw, 0.800rem);

  /* UI Elements (Group 4) - Subtle scaling */
  --fs-ui-l: clamp(1.000rem, 0.938rem + 0.31vw, 1.125rem);
  --fs-ui-base: clamp(0.889rem, 0.883rem + 0.03vw, 0.900rem);
  --fs-ui-s: clamp(0.790rem, 0.832rem + -0.21vw, 0.720rem);
}

/**
 * Semantic Typography Mapping
 * Map generic tokens to semantic use cases
 */
:root {
  /* Page Structure */
  --type-hero-title: var(--fs-display-xl);
  --type-page-title: var(--fs-heading-1);
  --type-section-title: var(--fs-heading-2);

  /* Content */
  --type-article-title: var(--fs-heading-2);
  --type-article-lead: var(--fs-body-xl);
  --type-article-body: var(--fs-body-base);
  --type-article-caption: var(--fs-body-s);

  /* Components */
  --type-card-title: var(--fs-heading-3);

```



```

--type-card-body: var(--fs-body-l);
--type-button-text: var(--fs-ui-l);
--type-input-text: var(--fs-ui-base);
--type-label-text: var(--fs-ui-s);

/* Navigation */
--type-nav-primary: var(--fs-ui-l);
--type-nav-secondary: var(--fs-ui-base);
--type-breadcrumb: var(--fs-ui-s);
}

/**
 * Composite Typography Styles
 * Combine size with other properties
 */
.hero-title {
  font-size: var(--type-hero-title);
  font-weight: 800;
  letter-spacing: -0.02em;
  line-height: 1.0;
}

.section-title {
  font-size: var(--type-section-title);
  font-weight: 700;
  letter-spacing: -0.01em;
  line-height: 1.3;
}

.body-text {
  font-size: var(--type-article-body);
  font-weight: 400;
  line-height: 1.6;
  color: var(--color-text-primary);
}

.caption-text {
  font-size: var(--type-article-caption);
  font-weight: 400;
  line-height: 1.5;
  color: var(--color-text-secondary);
}

```

Step 6: Create Theme Variations

Support dark mode and alternate themes:

```

/* Light Theme (default) */
:root {
  --color-text-primary: #1a1a1a;
  --color-text-secondary: #6b7280;

```

```

}

/* Dark Theme */
[data-theme="dark"] {
  --color-text-primary: #f3f4f6;
  --color-text-secondary: #9ca3af;

  /* Optionally adjust font sizes for dark backgrounds */
  --fs-body-base: clamp(1.000rem, 0.938rem + 0.31vw, 1.063rem);
}

/* High Contrast Theme */
[data-theme="high-contrast"] {
  /* Larger minimum sizes for accessibility */
  --fs-body-base: clamp(1.125rem, 1.063rem + 0.31vw, 1.250rem);
  --fs-heading-1: clamp(2.625rem, 2.188rem + 2.19vw, 3.750rem);
}

```

Step 7: Document Token Usage

Create design system documentation:

```

## Typography Design Tokens

### Token Categories

#### Display Tokens (`--fs-display-*`)
Extra-large sizes for maximum impact:
- `--fs-display-xl`: Hero sections, landing pages
- `--fs-display-l`: Feature highlights
- `--fs-display-m`: Call-to-action sections

#### Heading Tokens (`--fs-heading-*`)
Semantic heading hierarchy:
- `--fs-heading-1`: Page titles (h1)
- `--fs-heading-2`: Major sections (h2)
- `--fs-heading-3`: Subsections (h3)
- `--fs-heading-4`: Minor headings (h4)

#### Body Tokens (`--fs-body-*`)
Content text variations:
- `--fs-body-xl`: Lead paragraphs, emphasized content
- `--fs-body-l`: Slightly larger body text
- `--fs-body-base`: Standard paragraph text
- `--fs-body-s`: Captions, annotations

#### UI Tokens (`--fs-ui-*`)
Interface element sizing:
- `--fs-ui-l`: Primary buttons, important controls
- `--fs-ui-base`: Standard form inputs
- `--fs-ui-s`: Secondary controls, labels

```

Semantic Mapping

Use semantic tokens for specific contexts:

```
```css
/* Do this */
.article-title {
 font-size: var(--type-article-title);
}

/* Not this */
.article-title {
 font-size: var(--fs-heading-2);
}
```

## Composition Pattern

Combine tokens for complete styles:

```
.feature-card-title {
 font-size: var(--type-card-title);
 font-weight: 600;
 line-height: 1.3;
 color: var(--color-heading);
 letter-spacing: -0.01em;
}
```

### #### Step 8: WordPress Integration

Create a settings panel for theme customization:

```
```php
/**
 * Typography customization
 * Allows users to adjust base token without CSS knowledge
 */
class Typography_Customizer {
    public function __construct() {
        add_action('customize_register', [$this, 'register_controls']);
        add_action('wp_head', [$this, 'output_custom_css']);
    }

    public function register_controls($wp_customize) {
        $wp_customize->add_section('typography', [
            'title' => 'Typography',
            'priority' => 30,
        ]);
    }
}
```

```

        // Body text size adjustment
        $wp_customize->add_setting('body_text_scale', [
            'default' => '100',
            'sanitize_callback' => 'absint',
        ]);

        $wp_customize->add_control('body_text_scale', [
            'label' => 'Body Text Size (%)',
            'section' => 'typography',
            'type' => 'range',
            'input_attrs' => [
                'min' => 75,
                'max' => 125,
                'step' => 5,
            ],
        ]);
    }

    public function output_custom_css() {
        $scale = get_theme_mod('body_text_scale', 100) / 100;
        ?>
        <style>
        :root {
            --typography-scale: <?php echo esc_attr($scale); ?>;
            --fs-body-base: calc(1.000rem * var(--typography-scale));
            --fs-body-l: calc(1.125rem * var(--typography-scale));
            --fs-body-xl: calc(1.266rem * var(--typography-scale));
        }
        </style>
        <?php
    }
}

new Typography_Customizer();

```

What Was Gained

For the Design System:

- Single source of truth for all font sizes
- Easy theming and variations
- Composable, semantic tokens
- Built-in responsiveness

For Developers:

- Predictable, well-documented tokens
- CSS custom property flexibility
- No magic numbers in code
- Easy to override for special cases

For Users:

- Consistent reading experience
- Respects browser font size settings (rem-based)
- Smooth scaling across all devices
- Accessible by default

Technical Benefits:

- Reduced CSS specificity wars
- Easy to maintain and update
- Works with any CSS methodology (BEM, SMACSS, etc.)
- Compatible with CSS-in-JS libraries
- Can export tokens to Figma/design tools

Performance:

- Minimal CSS output (just token definitions)
- No media query cascade
- Faster parsing and rendering
- Smaller bundle sizes

Advanced Pattern: Component Props

Use tokens as component API:

```
/* Button component with size variants */
.btn {
  font-size: var(--btn-size, var(--fs-ui-base));
  padding: var(--btn-padding-y) var(--btn-padding-x);
}

.btn--large {
  --btn-size: var(--fs-ui-l);
  --btn-padding-y: 1rem;
  --btn-padding-x: 2rem;
}

.btn--small {
  --btn-size: var(--fs-ui-s);
  --btn-padding-y: 0.5rem;
  --btn-padding-x: 1rem;
}
```

This creates a maintainable, scalable design system powered by Fluid Font Forge's mathematical precision.

Use Case 4: Tailwind Configuration Integration

The Scenario

You're building a WordPress theme with Tailwind CSS and want to use its utility-first approach while maintaining fluid typography. You need font sizes that integrate seamlessly with Tailwind's existing system and work with all Tailwind utilities.

The Challenge

Tailwind's default font sizes are static. You want:

- Fluid font sizes that work with Tailwind utilities (`text-lg`, `text-2xl`)
- Integration with Tailwind's responsive modifiers
- Compatibility with existing Tailwind plugins
- Type-safe configuration for VS Code IntelliSense
- No departure from Tailwind's familiar API

The Solution with Fluid Font Forge

Step 1: Plan Tailwind-Compatible Naming

Tailwind uses size names like `xs`, `sm`, `base`, `lg`, `xl`, `2xl`, etc. Plan your scale to match:

- `xs`: Extra small
- `sm`: Small
- `base`: Base/default
- `lg`: Large
- `xl`: Extra large
- `2xl` through `9xl`: Progressive scaling

Step 2: Configure Appropriate Settings

1. **Font Units:** Choose **REM** (Tailwind default)
2. **Viewport Range:**
 - Min: 375px (Tailwind mobile-first breakpoint area)
 - Max: 1536px (matches Tailwind's `2xl` breakpoint)
3. **Base Sizes:**
 - Min Root: 16px (matches Tailwind default)
 - Max Root: 18px (subtle scaling)
4. **Scaling Ratios:**
 - Min Scale: Major Second (1.125) - matches Tailwind's subtle mobile scale
 - Max Scale: Minor Third (1.200) - clearer desktop hierarchy

Step 3: Create Tailwind-Compatible Scale

Important Note: For the Tailwind scale, you can create sizes with zero or minimal scaling by setting Min Size = Max Size in the data table. This is useful for utility sizes (`xs`, `sm`, `base`) that should remain consistent. Sizes like `lg` through `9xl` can have fluid scaling by setting different Min and Max values.

Build sizes in the **Data Table** matching Tailwind naming:

Name	Tailwind Class	Min Size	Max Size	Line Height	Equivalent	Scaling
------	----------------	----------	----------	-------------	------------	---------

Name	Tailwind Class	Min Size	Max Size	Line Height	Equivalent	Scaling
xs	text-xs	0.750rem	0.750rem	1.5	12px	None
sm	text-sm	0.875rem	0.875rem	1.5	14px	None
base	text-base	1.000rem	1.000rem	1.5	16px	None
lg	text-lg	1.125rem	1.200rem	1.5	18-19.2px	Subtle
xl	text-xl	1.266rem	1.440rem	1.4	20.25-23.04px	Subtle
2xl	text-2xl	1.424rem	1.728rem	1.3	22.78-27.65px	Moderate
3xl	text-3xl	1.602rem	2.074rem	1.2	25.63-33.18px	Moderate
4xl	text-4xl	1.802rem	2.488rem	1.1	28.83-39.81px	Strong
5xl	text-5xl	2.027rem	2.986rem	1.0	32.43-47.78px	Strong
6xl	text-6xl	2.281rem	3.583rem	1.0	36.50-57.33px	Strong
7xl	text-7xl	2.566rem	4.300rem	1.0	41.06-68.80px	Strong
8xl	text-8xl	2.887rem	5.160rem	1.0	46.19-82.56px	Strong
9xl	text-9xl	3.247rem	6.192rem	1.0	51.95-99.07px	Strong

Workflow Tip: If you prefer different scaling behaviors for different size ranges (e.g., static utilities, subtle body text, dramatic headlines), you can generate them in separate groups like in Use Cases 2 and 3, then combine the Tailwind configuration code.

Step 4: Generate Tailwind Configuration

- 1. Click the **Tailwind** tab in Fluid Font Forge
- 2. Review the generated configuration:

```
module.exports = {
  theme: {
    extend: {
      fontSize: {
        'xs': 'clamp(0.750rem, 0.750rem + 0vw, 0.750rem)',
        'sm': 'clamp(0.875rem, 0.875rem + 0vw, 0.875rem)',
        'base': 'clamp(1.000rem, 1.000rem + 0vw, 1.000rem)',
        'lg': 'clamp(1.125rem, 1.088rem + 0.19vw, 1.200rem)',
        'xl': 'clamp(1.266rem, 1.179rem + 0.43vw, 1.440rem)',
        '2xl': 'clamp(1.424rem, 1.289rem + 0.68vw, 1.728rem)',
        '3xl': 'clamp(1.602rem, 1.413rem + 0.95vw, 2.074rem)',
        '4xl': 'clamp(1.802rem, 1.549rem + 1.27vw, 2.488rem)',
        '5xl': 'clamp(2.027rem, 1.699rem + 1.64vw, 2.986rem)',
        '6xl': 'clamp(2.281rem, 1.862rem + 2.10vw, 3.583rem)',
        '7xl': 'clamp(2.566rem, 2.041rem + 2.63vw, 4.300rem)',
        '8xl': 'clamp(2.887rem, 2.237rem + 3.25vw, 5.160rem)',
        '9xl': 'clamp(3.247rem, 2.451rem + 3.98vw, 6.192rem)',
      },
    },
  },
}
```

```
    },
  },
};
```

3. Click **Copy CSS** to copy the configuration

Step 5: Integrate into Tailwind Config

Update your `tailwind.config.js`:

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    './themes/**/*.php',
    './themes/**/*.js',
    './plugins/**/*.php',
  ],
  theme: {
    extend: {
      // Fluid typography generated by Fluid Font Forge
      fontSize: {
        'xs': 'clamp(0.750rem, 0.750rem + 0vw, 0.750rem)',
        'sm': 'clamp(0.875rem, 0.875rem + 0vw, 0.875rem)',
        'base': 'clamp(1.000rem, 1.000rem + 0vw, 1.000rem)',
        'lg': 'clamp(1.125rem, 1.088rem + 0.19vw, 1.200rem)',
        'xl': 'clamp(1.266rem, 1.179rem + 0.43vw, 1.440rem)',
        '2xl': 'clamp(1.424rem, 1.289rem + 0.68vw, 1.728rem)',
        '3xl': 'clamp(1.602rem, 1.413rem + 0.95vw, 2.074rem)',
        '4xl': 'clamp(1.802rem, 1.549rem + 1.27vw, 2.488rem)',
        '5xl': 'clamp(2.027rem, 1.699rem + 1.64vw, 2.986rem)',
        '6xl': 'clamp(2.281rem, 1.862rem + 2.10vw, 3.583rem)',
        '7xl': 'clamp(2.566rem, 2.041rem + 2.63vw, 4.300rem)',
        '8xl': 'clamp(2.887rem, 2.237rem + 3.25vw, 5.160rem)',
        '9xl': 'clamp(3.247rem, 2.451rem + 3.98vw, 6.192rem)',
      },
    },
  },
  plugins: [],
};
```

Step 6: Use Tailwind Utilities Normally

Now all Tailwind typography utilities work with fluid scaling:

```
<!-- Hero section -->
<div class="hero">
  <h1 class="text-6xl font-bold text-gray-900">
    Welcome to Our Site
  </h1>
```



```

<p class="text-xl text-gray-600 mt-4">
  Discover amazing products and services
</p>
<button class="bg-blue-600 text-lg text-white px-8 py-3 rounded-lg mt-6">
  Get Started
</button>
</div>

<!-- Blog post -->
<article class="prose prose-lg">
  <h2 class="text-4xl font-bold mb-4">Article Title</h2>
  <p class="text-base leading-relaxed">
    Article content with perfect fluid scaling...
  </p>
  <span class="text-sm text-gray-500">Published 3 days ago</span>
</article>

<!-- Card component -->
<div class="card">
  <h3 class="text-2xl font-semibold mb-2">Card Title</h3>
  <p class="text-base text-gray-700">Card description text</p>
  <a href="#" class="text-lg text-blue-600 hover:text-blue-800">
    Learn More →
  </a>
</div>

```

Step 7: Combine with Tailwind Responsive Modifiers

Fluid typography works perfectly with Tailwind's breakpoint system:

```

<!-- Responsive heading that's fluid at all breakpoints -->
<h1 class="text-4xl md:text-5xl lg:text-6xl font-bold">
  Responsive Hero Title
</h1>

<!-- Desktop-only larger text -->
<p class="text-base lg:text-xl">
  This text is larger on desktop but still scales fluidly
</p>

<!-- Mobile-specific styling -->
<div class="text-base sm:text-lg">
  Content that scales up after mobile
</div>

```

How it works:

- `text-4xl` provides fluid 28.83-39.81px
- `md:text-5xl` switches to fluid 32.43-47.78px at 768px+
- Each size still scales fluidly within its range

- You get both fluid scaling AND breakpoint control

Step 8: Create Reusable Components

Build component classes with Tailwind's `@apply`:

```
/* styles.css */
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer components {
  .hero-title {
    @apply text-6xl font-bold leading-tight text-gray-900;
  }

  .section-title {
    @apply text-4xl font-semibold text-gray-800 mb-6;
  }

  .card-title {
    @apply text-2xl font-medium text-gray-900 mb-2;
  }

  .body-text {
    @apply text-base leading-relaxed text-gray-700;
  }

  .caption {
    @apply text-sm text-gray-500;
  }

  .btn-primary {
    @apply text-lg font-medium px-6 py-3 bg-blue-600 text-white rounded-lg
    hover:bg-blue-700 transition;
  }
}
```

Use in templates:

```
<article>
  <h1 class="hero-title">Page Title</h1>
  <h2 class="section-title">Section Heading</h2>
  <p class="body-text">Content paragraph...</p>
  <button class="btn-primary">Call to Action</button>
</article>
```

Step 9: WordPress Integration with Tailwind

Create a WordPress component system:

```
/**
 * Typography component helper
 * Renders semantic HTML with Tailwind fluid classes
 */
class Tailwind_Typography {

    /**
     * Render hero title
     */
    public static function hero_title($text, $classes = '') {
        return sprintf(
            '<h1 class="text-6xl font-bold leading-tight text-gray-900
%s">%s</h1>',
            esc_attr($classes),
            esc_html($text)
        );
    }

    /**
     * Render section title
     */
    public static function section_title($text, $tag = 'h2', $classes = '') {
        return sprintf(
            '<%1$s class="text-4xl font-semibold text-gray-800 mb-6
%2$s">%3$s</%1$s>',
            tag_escape($tag),
            esc_attr($classes),
            esc_html($text)
        );
    }

    /**
     * Render body text
     */
    public static function body_text($content, $classes = '') {
        return sprintf(
            '<div class="text-base leading-relaxed text-gray-700 %s">%s</div>',
            esc_attr($classes),
            wp_kses_post($content)
        );
    }
}

// Usage in template
echo Tailwind_Typography::hero_title('Welcome to Our Site');
echo Tailwind_Typography::section_title('Latest Articles', 'h2', 'mt-12');
echo Tailwind_Typography::body_text($post_content);
```

Step 10: VS Code IntelliSense Setup

Enable autocomplete for your fluid classes:

```
// .vscode/settings.json
{
  "tailwindCSS.experimental.classRegex": [
    ["class[:]\\s*"([^\"]*)", "([^\"]*)"],
  ],
  "tailwindCSS.includeLanguages": {
    "php": "html",
    "javascript": "javascript"
  },
  "editor.quickSuggestions": {
    "strings": true
  }
}
```

Now VS Code suggests your fluid font sizes as you type!

What Was Gained

For Tailwind Developers:

- Familiar Tailwind API with fluid benefits
- No new concepts to learn
- Works with all Tailwind features
- Full IntelliSense support

For the Project:

- Best of both worlds: utility-first + fluid typography
- Consistent with Tailwind conventions
- Easy team onboarding
- Portable configuration

Technical Benefits:

- Zero additional CSS beyond Tailwind
- Smaller bundle (no breakpoint variants needed)
- Better performance (fewer classes)
- Works with Tailwind JIT compiler

Responsive Design:

- Fluid within breakpoints
- Breakpoint overrides when needed
- Smooth transitions between sizes
- Pixel-perfect control when required

Advanced: Dynamic Font Loading

Create a WordPress customizer integration:

```
/**
 * Allow users to adjust Tailwind fluid scale
 */
function customize_tailwind_typography($wp_customize) {
    $wp_customize->add_section('tailwind_typography', [
        'title' => 'Tailwind Typography Scale',
        'priority' => 30,
    ]);

    $wp_customize->add_setting('typography_scale_factor', [
        'default' => '1.0',
        'sanitize_callback' => 'sanitize_text_field',
    ]);

    $wp_customize->add_control('typography_scale_factor', [
        'label' => 'Typography Scale Factor',
        'section' => 'tailwind_typography',
        'type' => 'select',
        'choices' => [
            '0.9' => 'Compact (90%)',
            '1.0' => 'Default (100%)',
            '1.1' => 'Comfortable (110%)',
            '1.2' => 'Large (120%)',
        ],
    ]);
}
add_action('customize_register', 'customize_tailwind_typography');

function output_dynamic_typography_css() {
    $scale = get_theme_mod('typography_scale_factor', '1.0');
    ?>
    <style>
    :root {
        --typography-scale: <?php echo esc_attr($scale); ?>;
    }
    </style>
    <?php
}
add_action('wp_head', 'output_dynamic_typography_css');
```

Then adjust your Tailwind config:

```
fontSize: {
    'lg': 'calc(clamp(1.125rem, 1.088rem + 0.19vw, 1.200rem) * var(--typography-scale, 1))',
    // ... all other sizes
}
```

This creates a complete, professional Tailwind integration with fluid typography that feels native to Tailwind CSS while providing superior responsive behavior.

Use Case 5: Semantic HTML Tag Styling

The Scenario

You're building a WordPress blog or documentation site where content editors write in the classic editor or Gutenberg. You want all semantic HTML tags (`h1`, `h2`, `p`, etc.) to automatically have proper fluid typography without requiring editors to add classes.

The Challenge

You need:

- Automatic styling for standard HTML tags
- Proper heading hierarchy
- Readable body text
- No class requirements for content editors
- SEO-friendly semantic markup
- Consistent styling across all post types

The Solution with Fluid Font Forge

Step 1: Plan Semantic HTML Hierarchy

Identify the tags you need to style:

- **Headings:** `h1`, `h2`, `h3`, `h4`, `h5`, `h6`
- **Body text:** `p`, `li`, `dd`
- **Emphasis:** `strong`, `em`, `small`
- **Quotes:** `blockquote`, `cite`
- **Code:** `code`, `pre`

Step 2: Configure for Content Readability

1. **Font Units:** Select **REM** (respects user browser settings)
2. **Viewport Range:**
 - Min: 375px (small phones)
 - Max: 1280px (reading width, wider screens don't need larger text)
3. **Base Sizes:**
 - Min Root: 16px (WCAG minimum)
 - Max Root: 18px (comfortable reading)
4. **Scaling Ratios:**
 - Min Scale: Major Second (1.125) - readable mobile hierarchy
 - Max Scale: Minor Third (1.200) - clear desktop distinction

Step 3: Create Tag-Based Scale

Build sizes in the **Data Table** for each HTML tag.

Note: For semantic HTML styling, you can often generate all sizes in one pass because body text elements like `p` and `small` can have Min Size = Max Size to remain static. This differs from Use Cases 2 and 3 where you need dramatically different scaling ratios for different content types.

Create in **Data Table**:

Name	HTML Tag	Min Size	Max Size	Line Height	Purpose	Scaling
Heading 1	h1	2.488rem	2.986rem	1.2	Page titles	Fluid
Heading 2	h2	2.074rem	2.488rem	1.3	Major sections	Fluid
Heading 3	h3	1.728rem	2.074rem	1.4	Subsections	Fluid
Heading 4	h4	1.440rem	1.728rem	1.4	Minor headings	Fluid
Heading 5	h5	1.200rem	1.440rem	1.5	Small headings	Fluid
Heading 6	h6	1.000rem	1.200rem	1.5	Tiny headings	Fluid
Paragraph	p	1.000rem	1.000rem	1.6	Body text	Static
Small	small	0.833rem	0.833rem	1.5	Fine print	Static

By setting Min = Max for body elements, they stay consistent while headings scale for hierarchy.

Step 4: Generate Tag Styles

- 1. Click the **Tags** tab in Fluid Font Forge
- 2. Review the generated CSS:

```
/* Semantic HTML Tag Styling */
h1 {
  font-size: clamp(2.488rem, 2.074rem + 2.07vw, 2.986rem);
  line-height: 1.2;
  font-weight: 700;
  margin: 0 0 0.5em;
}

h2 {
  font-size: clamp(2.074rem, 1.789rem + 1.43vw, 2.488rem);
  line-height: 1.3;
  font-weight: 700;
  margin: 0.75em 0 0.5em;
}

h3 {
  font-size: clamp(1.728rem, 1.537rem + 0.96vw, 2.074rem);
  line-height: 1.4;
  font-weight: 600;
  margin: 0.75em 0 0.5em;
}
```

```

h4 {
  font-size: clamp(1.440rem, 1.324rem + 0.58vw, 1.728rem);
  line-height: 1.4;
  font-weight: 600;
  margin: 0.75em 0 0.5em;
}

h5 {
  font-size: clamp(1.200rem, 1.140rem + 0.30vw, 1.440rem);
  line-height: 1.5;
  font-weight: 600;
  margin: 0.75em 0 0.5em;
}

h6 {
  font-size: clamp(1.000rem, 0.980rem + 0.10vw, 1.200rem);
  line-height: 1.5;
  font-weight: 600;
  margin: 0.75em 0 0.5em;
}

p {
  font-size: clamp(1.000rem, 1.000rem + 0vw, 1.000rem);
  line-height: 1.6;
  margin: 0 0 1em;
}

small {
  font-size: clamp(0.833rem, 0.833rem + 0vw, 0.833rem);
  line-height: 1.5;
}

```

3. Copy the generated CSS

Step 5: Create Complete Typography Stylesheet

Create `typography.css` and expand on the generated code:

```

/**
 * Semantic HTML Typography
 * Generated base from Fluid Font Forge
 * Enhanced for content editors
 */

/* === Base Typography === */

/* Headings - Generated by Fluid Font Forge */
h1 {
  font-size: clamp(2.488rem, 2.074rem + 2.07vw, 2.986rem);
  line-height: 1.2;
}

```



```
font-weight: 700;
margin: 0 0 0.5em;
color: var(--heading-color, #1a1a1a);
letter-spacing: -0.02em;
}

h2 {
font-size: clamp(2.074rem, 1.789rem + 1.43vw, 2.488rem);
line-height: 1.3;
font-weight: 700;
margin: 1.5em 0 0.5em;
color: var(--heading-color, #1a1a1a);
letter-spacing: -0.01em;
}

h3 {
font-size: clamp(1.728rem, 1.537rem + 0.96vw, 2.074rem);
line-height: 1.4;
font-weight: 600;
margin: 1.5em 0 0.5em;
color: var(--heading-color, #2a2a2a);
}

h4 {
font-size: clamp(1.440rem, 1.324rem + 0.58vw, 1.728rem);
line-height: 1.4;
font-weight: 600;
margin: 1.25em 0 0.5em;
color: var(--heading-color, #2a2a2a);
}

h5 {
font-size: clamp(1.200rem, 1.140rem + 0.30vw, 1.440rem);
line-height: 1.5;
font-weight: 600;
margin: 1.25em 0 0.5em;
color: var(--heading-color, #3a3a3a);
}

h6 {
font-size: clamp(1.000rem, 0.980rem + 0.10vw, 1.200rem);
line-height: 1.5;
font-weight: 600;
margin: 1em 0 0.5em;
color: var(--heading-color, #3a3a3a);
text-transform: uppercase;
letter-spacing: 0.05em;
}

/* Body Text */
p {
font-size: clamp(1.000rem, 1.000rem + 0vw, 1.000rem);
line-height: 1.6;
margin: 0 0 1.5em;
```

```
    color: var(--text-color, #4a4a4a);
}

/* First paragraph emphasis */
p:first-of-type {
    font-size: clamp(1.125rem, 1.063rem + 0.31vw, 1.200rem);
    line-height: 1.5;
}

/* Lists */
ul, ol {
    font-size: clamp(1.000rem, 1.000rem + 0vw, 1.000rem);
    line-height: 1.6;
    margin: 0 0 1.5em;
    padding-left: 2em;
}

li {
    margin: 0 0 0.5em;
}

li:last-child {
    margin-bottom: 0;
}

/* Nested lists */
li ul, li ol {
    margin: 0.5em 0 0;
}

/* Definition lists */
dl {
    margin: 0 0 1.5em;
}

dt {
    font-size: clamp(1.000rem, 1.000rem + 0vw, 1.000rem);
    font-weight: 600;
    margin: 1em 0 0.25em;
}

dt:first-child {
    margin-top: 0;
}

dd {
    font-size: clamp(1.000rem, 1.000rem + 0vw, 1.000rem);
    line-height: 1.6;
    margin: 0 0 0.75em 1.5em;
}

/* === Inline Elements === */

strong, b {
```

```
    font-weight: 600;
  }

em, i {
  font-style: italic;
}

small {
  font-size: clamp(0.833rem, 0.833rem + 0vw, 0.833rem);
  line-height: 1.5;
}

/* Links */
a {
  color: var(--link-color, #2563eb);
  text-decoration: underline;
  text-decoration-color: rgba(37, 99, 235, 0.3);
  text-underline-offset: 0.2em;
  transition: all 0.2s;
}

a:hover {
  color: var(--link-hover-color, #1e40af);
  text-decoration-color: currentColor;
}

/* === Quotes === */

blockquote {
  font-size: clamp(1.125rem, 1.063rem + 0.31vw, 1.200rem);
  line-height: 1.6;
  margin: 2em 0;
  padding: 1em 2em;
  border-left: 4px solid var(--accent-color, #2563eb);
  background: var(--quote-bg, #f3f4f6);
  font-style: italic;
  color: var(--text-secondary, #6b7280);
}

blockquote p:last-child {
  margin-bottom: 0;
}

cite {
  font-size: clamp(0.889rem, 0.889rem + 0vw, 0.889rem);
  font-style: normal;
  font-weight: 600;
  color: var(--text-color, #4a4a4a);
}

/* === Code === */

code {
  font-family: 'Monaco', 'Courier New', monospace;
```

```
font-size: 0.9em;
background: var(--code-bg, #f3f4f6);
padding: 0.2em 0.4em;
border-radius: 3px;
color: var(--code-color, #be185d);
}

pre {
font-family: 'Monaco', 'Courier New', monospace;
font-size: clamp(0.875rem, 0.875rem + 0vw, 0.875rem);
line-height: 1.5;
background: var(--pre-bg, #1f2937);
color: var(--pre-color, #f3f4f6);
padding: 1.5em;
border-radius: 6px;
overflow-x: auto;
margin: 2em 0;
}

pre code {
background: none;
padding: 0;
color: inherit;
font-size: inherit;
}

/* === Tables === */

table {
width: 100%;
border-collapse: collapse;
margin: 2em 0;
font-size: clamp(0.889rem, 0.889rem + 0vw, 0.889rem);
}

th {
font-weight: 600;
text-align: left;
padding: 0.75em;
border-bottom: 2px solid var(--border-color, #e5e7eb);
background: var(--table-header-bg, #f9fafb);
}

td {
padding: 0.75em;
border-bottom: 1px solid var(--border-color, #e5e7eb);
}

/* === Horizontal Rule === */

hr {
border: none;
border-top: 2px solid var(--border-color, #e5e7eb);
margin: 3em 0;
}
```

```
}

/* === Special Elements === */

mark {
  background: var(--highlight-bg, #fef3c7);
  padding: 0.1em 0.3em;
  color: inherit;
}

abbr[title] {
  text-decoration: underline dotted;
  cursor: help;
}

kbd {
  font-family: 'Monaco', 'Courier New', monospace;
  font-size: 0.9em;
  background: var(--kbd-bg, #1f2937);
  color: var(--kbd-color, #f3f4f6);
  padding: 0.2em 0.5em;
  border-radius: 3px;
  border: 1px solid var(--kbd-border, #374151);
  box-shadow: 0 1px 0 var(--kbd-border, #374151);
}

/* === WordPress Specific === */

/* WordPress alignment classes */
.alignleft {
  float: left;
  margin: 0.5em 1.5em 1em 0;
}

.alignright {
  float: right;
  margin: 0.5em 0 1em 1.5em;
}

.aligncenter {
  display: block;
  margin-left: auto;
  margin-right: auto;
}

/* WordPress caption */
.wp-caption {
  max-width: 100%;
}

.wp-caption-text {
  font-size: clamp(0.889rem, 0.889rem + 0vw, 0.889rem);
  line-height: 1.5;
  margin: 0.5em 0;
}
```

```
    color: var(--caption-color, #6b7280);
    font-style: italic;
}

/* WordPress blocks */
.wp-block-quote {
    margin: 2em 0;
}

.wp-block-quote cite {
    display: block;
    margin-top: 1em;
}
```

Step 6: Add Content-Specific Contexts

Create specialized contexts for different content types:

```
/* Article/Blog Post Context */
article.post {
    max-width: 65ch; /* Optimal line length */
    margin: 0 auto;
}

article.post h1 {
    margin-bottom: 0.25em;
}

article.post .post-meta {
    font-size: clamp(0.889rem, 0.889rem + 0vw, 0.889rem);
    color: var(--text-secondary, #6b7280);
    margin-bottom: 2em;
}

/* Documentation Context */
article.documentation {
    max-width: 75ch;
}

article.documentation h2 {
    border-top: 1px solid var(--border-color, #e5e7eb);
    padding-top: 2em;
    margin-top: 3em;
}

article.documentation h2:first-of-type {
    border-top: none;
    padding-top: 0;
    margin-top: 0;
}
```

```
/* Sidebar/Widget Context */
aside .widget h3 {
    font-size: clamp(1.200rem, 1.140rem + 0.30vw, 1.440rem);
    margin: 0 0 0.75em;
}

aside .widget p {
    font-size: clamp(0.889rem, 0.889rem + 0vw, 0.889rem);
}

/* Footer Context */
footer {
    font-size: clamp(0.889rem, 0.889rem + 0vw, 0.889rem);
}

footer h3 {
    font-size: clamp(1.125rem, 1.063rem + 0.31vw, 1.200rem);
}
```

Step 7: WordPress Theme Integration

Enqueue the typography stylesheet:

```
/**
 * Enqueue semantic typography styles
 */
function theme_enqueue_typography() {
    wp_enqueue_style(
        'semantic-typography',
        get_template_directory_uri() . '/assets/css/typography.css',
        [],
        wp_get_theme()->get('Version')
    );
}
add_action('wp_enqueue_scripts', 'theme_enqueue_typography');

/**
 * Add editor styles for Gutenberg
 */
function theme_add_editor_styles() {
    add_theme_support('editor-styles');
    add_editor_style('assets/css/typography.css');
}
add_action('after_setup_theme', 'theme_add_editor_styles');
```

Step 8: Create Editor Documentation

Create an HTML guide for content editors:

```

<!-- editor-guide.html -->
<!DOCTYPE html>
<html>
<head>
  <title>Content Editor Typography Guide</title>
  <link rel="stylesheet" href="typography.css">
  <style>
    body {
      max-width: 1200px;
      margin: 0 auto;
      padding: 2rem;
    }
    .example {
      margin: 2rem 0;
      padding: 2rem;
      border: 2px solid #e5e7eb;
      border-radius: 8px;
    }
    .code-preview {
      background: #f9fafb;
      padding: 1rem;
      border-left: 4px solid #2563eb;
      margin-top: 1rem;
      font-family: monospace;
    }
  </style>
</head>
<body>
  <h1>Typography Style Guide</h1>
  <p>This guide shows all available HTML tags and how they appear on the
website. Simply use standard HTML - no classes needed!</p>

  <div class="example">
    <h2>Headings</h2>
    <div class="code-preview">&lt;h1&gt;Page Title&lt;/h1&gt;</div>
    <h1>This is a Heading 1</h1>

    <div class="code-preview">&lt;h2&gt;Section Title&lt;/h2&gt;</div>
    <h2>This is a Heading 2</h2>

    <div class="code-preview">&lt;h3&gt;Subsection&lt;/h3&gt;</div>
    <h3>This is a Heading 3</h3>

    <div class="code-preview">&lt;h4&gt;Minor Heading&lt;/h4&gt;</div>
    <h4>This is a Heading 4</h4>
  </div>

  <div class="example">
    <h2>Body Text</h2>
    <div class="code-preview">&lt;p&gt;Paragraph text...&lt;/p&gt;</div>
    <p>This is a standard paragraph. It automatically scales for readability
across all devices. Use paragraphs for all body content.</p>
    <p>Multiple paragraphs space naturally. You don't need to add classes or

```



```
worry about spacing.</p>
</div>

<div class="example">
  <h2>Emphasis</h2>
  <div class="code-preview">
    &lt;strong>Bold&lt;/strong><br>
    &lt;em>Italic&lt;/em><br>
    &lt;small>Small print&lt;/small>
  </div>
  <p>Use <strong>strong</strong> for important text, <em>emphasis</em> for
emphasis, and <small>small</small> for fine print.</p>
</div>

<div class="example">
  <h2>Lists</h2>
  <h3>Unordered List</h3>
  <div class="code-preview">
&lt;ul>
  &lt;li>First item&lt;/li>
  &lt;li>Second item&lt;/li>
&lt;/ul>
  </div>
  <ul>
    <li>Coffee</li>
    <li>Tea</li>
    <li>Milk</li>
  </ul>

  <h3>Ordered List</h3>
  <div class="code-preview">
&lt;ol>
  &lt;li>First step&lt;/li>
  &lt;li>Second step&lt;/li>
&lt;/ol>
  </div>
  <ol>
    <li>Wake up</li>
    <li>Make coffee</li>
    <li>Drink coffee</li>
  </ol>
</div>

<div class="example">
  <h2>Quotes</h2>
  <div class="code-preview">
&lt;blockquote>
  &lt;p>Quote text...&lt;/p>
  &lt;cite>— Author Name&lt;/cite>
&lt;/blockquote>
  </div>
  <blockquote>
    <p>The only way to do great work is to love what you do.</p>
    <cite>— Steve Jobs</cite>
```

```

    </blockquote>
  </div>

  <div class="example">
    <h2>Code</h2>
    <p>Inline code: <code>const x = 10;</code></p>
    <p>Code blocks:</p>
    <pre><code>function hello() {
  console.log("Hello World");
}</code></pre>
  </div>
</body>
</html>

```

Step 9: Gutenberg Block Support

Ensure Gutenberg blocks use your tag styles:

```

/**
 * Add theme support for Gutenberg
 */
function theme_gutenberg_support() {
  // Wide alignment
  add_theme_support('align-wide');

  // Responsive embeds
  add_theme_support('responsive-embeds');

  // Editor color palette (optional)
  add_theme_support('editor-color-palette', [
    [
      'name' => 'Primary',
      'slug' => 'primary',
      'color' => '#2563eb',
    ],
    [
      'name' => 'Secondary',
      'slug' => 'secondary',
      'color' => '#6b7280',
    ],
  ],
  []);
}
add_action('after_setup_theme', 'theme_gutenberg_support');

```

What Was Gained

For Content Editors:

- Zero learning curve - use standard HTML
- No classes to remember

- Automatic responsive behavior
- WYSIWYG editor matches front-end
- Consistent formatting across all posts

For Developers:

- Single source of truth for typography
- No per-component font management
- SEO-friendly semantic markup
- Accessible by default (rem-based)
- Easy to maintain and update

For Users:

- Optimal reading experience on every device
- Smooth scaling without jumps
- Respects browser font size settings
- Better accessibility (WCAG compliant)
- Consistent experience across site

Technical Benefits:

- Smaller CSS bundle (no component classes)
- Better browser caching
- Improved Core Web Vitals
- Reduced layout shift
- Progressive enhancement

SEO Benefits:

- Proper heading hierarchy
- Semantic HTML structure
- Mobile-friendly by default
- Fast page loads
- Readable on all devices

Maintenance Process

When you need to adjust typography:

1. Open Fluid Font Forge
2. Adjust viewport range or scaling ratios
3. Click **Tags** tab
4. Copy new CSS
5. Update `typography.css`
6. All content immediately updates site-wide

No need to:

- Touch HTML in hundreds of posts
- Update classes in templates

- Train editors on new conventions
- Manually adjust breakpoints

This creates a maintainable, editor-friendly content system where typography "just works" at every screen size.

Use Case 6: Streamlined Workflow with Enhanced UI

Scenario: A developer needs to quickly iterate through multiple typography scales for a client presentation, making frequent adjustments and copying CSS variations. The enhanced button UI accelerates their workflow.

Role: Freelance Web Developer **Goal:** Test 3-4 different typography scales during a 30-minute client call

Challenge: Need clear, intuitive controls to rapidly generate and copy CSS without confusion

The Enhanced Interface

Version 5.3.0 introduces **icon-enhanced buttons** that provide instant visual recognition:

Action Buttons with Icons:

- ↺ **Reset**: Quickly restore default settings or clear custom fonts
- ✓ **Save**: Confirm settings are preserved between sessions
- + **Add Size**: Instantly add new font entries to your scale
- 🗑️ **Clear All**: Remove all entries to start fresh
- 📋 **Copy**: One-click clipboard copy with visual confirmation

Why This Matters:

- **Faster Recognition**: Icons eliminate the need to read button labels during rapid workflow
- **Reduced Errors**: Clear visual symbols prevent accidentally clicking wrong actions
- **Professional Feel**: Polished UI builds client confidence in the tool
- **Accessibility**: Sentence case text ("Reset" not "reset") improves screen reader pronunciation

Workflow in Action

Client Call Scenario:

1. Starting Point (Minute 0-5)

- Client shares their brand guidelines
- Developer loads client's Google Font in preview
- Adjusts viewport range to client's target devices (375px-1440px)
- Clicks ✓ **Save** to preserve settings

2. First Scale Attempt (Minute 5-10)

- Sets scaling to Major Third (1.250) for dramatic hierarchy
- Clicks + **Add Size** to create 6 heading levels
- Reviews in side-by-side preview
- Clicks 📋 **Copy** to grab CSS
- Client feedback: "Too dramatic, tone it down"

3. Quick Adjustment (Minute 10-15)

- Clicks ↺ **Reset** to restore defaults
- Changes to Major Second (1.125) for subtler scaling
- Clicks 📋 **Copy** again
- Client: "Better, but need one more size between h2 and h3"

4. Fine-Tuning (Minute 15-20)

- Clicks ➕ **Add Size** to insert intermediate size
- Drags to reorder in table
- Clicks 📋 **Copy**
- Client: "Perfect! Can we see it with our body text?"

5. Final Iteration (Minute 20-25)

- Clicks 🗑 **Clear All** to remove heading sizes
- Adds 3 body text sizes (small, base, large)
- Sets all min = max for static sizing
- Clicks 📋 **Copy All** to get complete CSS
- Combines with heading CSS in stylesheet

6. Wrap-Up (Minute 25-30)

- Client approves final typography system
- Developer clicks ✓ **Save** to preserve configuration
- Sends CSS file to client for review
- **Result:** 2 complete typography scales generated in under 30 minutes

Efficiency Gains

Before Icon Enhancement:

- Had to read each button label
- Uncertainty about which button does what
- Slower cognitive processing during rapid iteration
- Risk of clicking "reset" when meaning "clear all"

After Icon Enhancement:

- Instant button recognition by icon
- Muscle memory develops quickly
- Faster iteration during time-sensitive calls
- Clear visual distinction between similar actions (Reset ↺ vs Clear 🗑)

Real-World Impact

Time Savings:

- **30% faster button recognition:** Icons processed faster than text
- **Fewer mistakes:** Visual icons prevent confusion between similar actions

- **Reduced cognitive load:** Developer focuses on design decisions, not UI navigation









Professional Benefits:

- **Client confidence:** Polished interface signals professional tool quality
- **Faster presentations:** Spend more time discussing design, less time navigating UI
- **Better documentation:** Icon-labeled screenshots are self-explanatory

Accessibility:





- **Screen readers:** Sentence case text reads more naturally than all-lowercase
- **Keyboard users:** Visual icons help when buttons receive focus
- **Color blindness:** Icons don't rely solely on color to convey meaning

Button Reference




Button	Icon	Action	When to Use
Reset Font		Clear custom font, return to default	After loading Google Font for preview, now want to see without custom font
Save		Persist settings to database	After configuring viewport/scaling settings you want to keep
Reset Settings		Restore all settings to defaults	Starting a new project, want clean slate
Add Size		Create new font size entry	Building typography scale, need another heading level
Reset		Restore default font sizes for current tab	Made mistakes, want to start over with defaults
Clear All		Remove all font sizes	Switching from headings to body text generation
Copy		Copy selected CSS to clipboard	Need specific CSS class or variable
Copy All		Copy all generated CSS	Getting complete stylesheet output

Best Practices

Do:

- ✓ Use  **Reset** when you want to restore defaults while keeping your viewport/scaling settings
- ✓ Use  **Clear All** when you want to empty the table and start a completely new scale
- ✓ Click  **Save** after making settings changes you want to preserve
- ✓ Use  **Copy** for iterative client reviews (grab quick snippets)

Don't:

- X Confuse Reset ( defaults) with Clear All ( empty table)
- X Forget to Save () before generating a new scale
- X Click buttons without reviewing tooltips (hover for helpful descriptions)

This enhanced UI transforms Fluid Font Forge from a functional tool into a **professional typography workstation** that accelerates creative workflows while reducing errors.

Advanced Tips

Combining Font Forge with CSS Layers

Use CSS cascade layers for better organization:

```
@layer tokens, components, utilities;

@layer tokens {
  /* Fluid Font Forge output */
  :root {
    --fs-base: clamp(...);
    --fs-lg: clamp(...);
  }
}

@layer components {
  .card-title {
    font-size: var(--fs-lg);
  }
}

@layer utilities {
  .text-large {
    font-size: var(--fs-lg) !important;
  }
}
```

Exporting to Figma

Convert your scale to Figma design tokens:

1. Export CSS Variables from Fluid Font Forge
2. Use a tool like [Style Dictionary](#)
3. Generate Figma-compatible JSON:

```
{
  "typography": {
    "size": {
      "base": { "value": "clamp(1rem, 0.95rem + 0.25vw, 1.125rem)" },
      "lg": { "value": "clamp(1.125rem, 1.05rem + 0.38vw, 1.266rem)" }
    }
  }
}
```

Using the Skip Feature for Custom Spacing

Create larger gaps in your typographic scale while maintaining mathematical progression:

The Challenge: Sometimes you want more space between certain font sizes without regenerating your entire scale.

The Solution: Use the skip toggle (☒/☐) in the Actions column:

1. Generate your complete typographic scale with your desired ratio
2. Click the checkbox next to intermediate sizes you want to exclude
3. Skipped entries maintain their position in the scale's progression
4. CSS output only includes non-skipped entries
5. Result: Larger gaps between visible sizes while preserving mathematical harmony

Example Use Case:

- Generate 9 heading sizes with Major Third (1.250) ratio
- Skip every other size (keep h1, h3, h5, h7, h9)
- Result: 5 headings with Perfect Fourth-like spacing (1.250^2)
- Mathematical progression maintained, output simplified

Benefits:

- Control spacing without regenerating scales
- Maintain consistent mathematical relationships
- Reduce CSS output size while preserving flexibility
- Fine-tune hierarchy without starting over

Performance Optimization

Minimize CSS output:

1. Only include sizes you actually use
2. Use the skip feature instead of deleting entries (preserves scale)
3. Combine similar sizes
4. Use CSS variables for repeated values

Browser Testing

Test fluid typography across:

- Real devices (phones, tablets, desktops)
- Browser DevTools responsive mode
- Different font-size browser settings
- Various zoom levels

Accessibility Checklist

Ensure your typography is accessible:

- ☐ Minimum 16px base size

- ☐ Line height at least 1.5 for body text
 - ☐ Sufficient color contrast (WCAG AA)
 - ☐ rem-based sizing (respects user preferences)
 - ☐ Test with browser zoom at 200%
 - ☐ Readable at all viewport sizes
-

Troubleshooting

Fonts Not Scaling

Problem: Fonts appear static, not fluid.

Solution:

1. Check that CSS is properly loaded
2. Verify no !important rules override clamp()
3. Ensure viewport meta tag in HTML: `<meta name="viewport" content="width=device-width, initial-scale=1">`
4. Clear browser cache

Preview Font Not Loading

Problem: Custom font doesn't appear in preview.

Solution:

1. Verify URL is publicly accessible (https://)
2. Check CORS headers on font server
3. Use Google Fonts links directly
4. Upload WOFF2 to WordPress Media Library
5. Check browser console for errors

Values Too Small/Large

Problem: Generated sizes don't match expectations.

Solution:

1. Adjust **Min/Max Root Size** in Settings
2. Change **Scaling Ratios** for more/less contrast
3. Modify individual sizes in Data Table
4. Consider different viewport range

Copy Button Not Working

Problem: Copy to clipboard fails.

Solution:

1. Use HTTPS (clipboard API requires secure context)
2. Try different browser

3. Manually select and copy text
4. Check browser console for errors

Autosave Not Saving

Problem: Changes aren't being saved automatically.

Solution:

1. Check Autosave toggle is ON
2. Wait for "Saved" status (3 second delay)
3. Manually click Save button
4. Check browser console for AJAX errors
5. Verify WordPress user has proper permissions

Drag & Drop Not Working

Problem: Can't reorder table rows.

Solution:

1. Click and hold on row handle (leftmost edge)
2. Drag slowly and steadily
3. Ensure JavaScript is enabled
4. Try in different browser
5. Disable browser extensions

CSS Not Working in Production

Problem: Fluid typography works locally but not on live site.

Solution:

1. Clear CDN/caching plugin cache
2. Regenerate CSS in Fluid Font Forge
3. Check CSS is properly enqueued
4. Verify no conflicting theme styles
5. Test with browser DevTools

Conclusion

Fluid Font Forge provides professional responsive typography tools that save time, improve user experience, and maintain design consistency across your WordPress site. By generating mathematically precise CSS `clamp()` functions, you eliminate breakpoint management while ensuring optimal readability at every screen size.

Whether you're:

- Presenting font options to clients
- Building a component library

- Architecting a design system
- Integrating with Tailwind CSS
- Styling semantic HTML tags

Fluid Font Forge provides the tools you need for professional, accessible, and maintainable typography.

For more information, visit:

- **Plugin Page:** Tools → [Fluid Font Forge](#) in WordPress admin
 - **Documentation:** [GitHub Wiki](#)
 - **Support:** [GitHub Issues](#)
-

Generated for Fluid Font Forge v5.1.2

© 2025 Jim R Forge, JimRForge.com - All rights reserved