GSAP Animation Components

A comprehensive collection of reusable GSAP animation components for modern web applications. These components provide various text effects and animations that can be easily integrated into any React project.

🚀 Features

- **Reusable Components**: All animations are modular and reusable
- **Modern Effects**: Contemporary animation styles including glitch, liquid, magnetic, and gradient effects
- **Performance Optimized**: Built with GSAP for smooth 60fps animations
- **Customizable**: Extensive configuration options for each animation type
- **TypeScript Ready**: Full TypeScript support with proper type definitions

📦 Installation

```
npm install gsap
```

🎯 Quick Start

```
import { SplitTextAnimation, GradientText, TypewriterText, BasicRevealText }
from './components/animations';

function MyComponent() {
  return (
    <div>
      <BasicRevealText delay={0}>
        <h1>Hello World</h1>
      </BasicRevealText>
      <SplitTextAnimation
        text="Split Text Animation"
        splitType="words"
        animationType="stagger"
      />
      <GradientText
        text="Animated Gradient"
        gradientColors={['#ff6b6b', '#4ecdc4', '#45b7d1']}
      />
      <TypewriterText
        text="Typewriter effect"
        speed={0.05}
        cursor={true}
      />
    </div>
```

```
  );
}
```

Animation Components

TextAnimations.jsx

AnimatedText

Basic animated text component with multiple animation types.

```
<AnimatedText
  text="Your text here"
  animationType="fadeInUp" // 'typewriter', 'stagger', 'morph', 'fadeInUp',
'slideIn'
  options={{
    duration: 1,
    delay: 0,
    ease: "power2.out"
  }}
/>
```

StaggeredText

Word-by-word staggered animation.

```
<StaggeredText
  text="Staggered animation text"
  staggerDelay={0.1}
  animationOptions={{
    duration: 0.8,
    ease: "power2.out"
  }}
/>
```

TypewriterText

Classic typewriter effect with cursor.

```
<TypewriterText
  text="Typewriter text"
  speed={0.05}
  cursor={true}
  cursorChar="|"
/>
```

MorphingText

Text that morphs between different states.

```
<MorphingText
  texts={["Text 1", "Text 2", "Text 3"]}
  morphDuration={2}
  pauseDuration={3}
/>
```

ModernTextEffects.jsx

GlitchText

Cyberpunk-style glitch effect.

```
<GlitchText
  text="Glitch Text"
  glitchIntensity={0.1}
  glitchDuration={0.1}
/>
```

LiquidText

Liquid/morphing text effect.

```
<LiquidText
  text="Liquid Text"
  liquidIntensity={1}
/>
```

MagneticText

Text that follows mouse movement.

```
<MagneticText
  text="Magnetic Text"
  magneticStrength={0.3}
/>
```

RevealText

Character-by-character reveal animation.

```
<RevealText
  text="Reveal Text"
  revealType="fade" // 'fade', 'slide', 'scale', 'rotate'
  staggerDelay={0.05}
/>
```

GradientText

Animated gradient text effect.

```
<GradientText
  text="Gradient Text"
  gradientColors={['#ff6b6b', '#4ecdc4', '#45b7d1', '#96ceb4']}
  animationSpeed={2}
/>
```

BasicRevealText.jsx

BasicRevealText

Simple reveal animation from bottom with opacity fade-in. Perfect for headers, buttons, and any content that needs a clean reveal effect.

```
<BasicRevealText
  delay={0.2}
  duration={0.6}
  ease="power2.out"
>
  <YourContent />
</BasicRevealText>
```

Props:

- `children`: Content to animate (any React element)
- `delay`: Animation start delay in seconds (default: 0)
- `duration`: Animation duration in seconds (default: 0.6)
- `ease`: GSAP easing function (default: "power2.out")
- `className`: Additional CSS classes

Usage Examples:

Header Elements:

```
<BasicRevealText delay={0}>
  <img src="/logo.png" alt="Logo" />
</BasicRevealText>

<BasicRevealText delay={0.1}>
  <nav>
    <a href="/services">Services</a>
    <a href="/about">About</a>
  </nav>
</BasicRevealText>

<BasicRevealText delay={0.2}>
  <button>Get Started</button>
</BasicRevealText>
```

Text Content:

```
<BasicRevealText delay={0.3} duration={0.8}>
  <h1>Welcome to our website</h1>
</BasicRevealText>
```

AnimationController.jsx

AnimationController

Master controller for complex animations.

```
<AnimationController
  trigger="onLoad" // 'onLoad', 'onHover'
  scrollTrigger={false}
>
  <YourContent />
</AnimationController>
```

SplitTextAnimation

Advanced text splitting with various animation patterns.

```
<SplitTextAnimation
  text="Split Text Animation"
  splitType="words" // 'words', 'characters', 'lines'
```

```
  animationType="stagger" // 'stagger', 'wave', 'random', 'center'
/>
```

ParallaxText

Text with parallax scrolling effect.

```
<ParallaxText
  text="Parallax Text"
  parallaxSpeed={0.5}
/>
```

MorphingTextSequence

Text that morphs between different phrases.

```
<MorphingTextSequence
  texts={["First text", "Second text", "Third text"]}
  morphDuration={1}
  pauseDuration={2}
/>
```

⚙️ Configuration Options

Common Props

- `text`: The text content to animate
- `className`: CSS classes to apply
- `duration`: Animation duration in seconds
- `delay`: Animation delay in seconds
- `ease`: GSAP easing function

Animation-Specific Props

- `speed`: Animation speed (for typewriter effects)
- `staggerDelay`: Delay between staggered elements
- `intensity`: Effect intensity (for glitch, liquid effects)
- `gradientColors`: Array of colors for gradient effects
- `magneticStrength`: Mouse follow strength for magnetic effects

🎯 Usage Examples

Hero Section with Multiple Effects

```
import {
  SplitTextAnimation,
  GradientText,
  TypewriterText,
  AnimationController
} from './components/animations';

function Hero() {
  return (
    <AnimationController trigger="onLoad">
      <h1>
        <SplitTextAnimation
          text="Maximum Profit"
          splitType="words"
          animationType="stagger"
        />
        <GradientText
          text="from Advertising"
          gradientColors={['#ff6b6b', '#4ecdc4']}
        />
      </h1>
      <TypewriterText
        text="We help raise products to the top"
        speed={0.03}
      />
    </AnimationController>
  );
}
```

Interactive Statistics

```
import { MagneticText, LiquidText, GlitchText } from './components/animations';

function Statistics() {
  return (
    <div>
      <MagneticText text="85%" magneticStrength={0.2} />
      <LiquidText text="120K" liquidIntensity={0.5} />
      <GlitchText text="5%" glitchIntensity={0.3} />
    </div>
  );
}
```

🎨 Customization

Custom Easing

```
<AnimatedText
  text="Custom Easing"
  options={{
    ease: "elastic.out(1, 0.3)"
  }}
/>
```

Custom Colors

```
<GradientText
  text="Custom Colors"
  gradientColors={['#your-color-1', '#your-color-2', '#your-color-3']}
/>
```

Custom Timing

```
<StaggeredText
  text="Custom Timing"
  staggerDelay={0.2}
  animationOptions={{
    duration: 1.5,
    ease: "power3.out"
  }}
/>
```

🚀 Performance Tips

1. **Use `will-change` CSS property** for elements that will be animated
2. **Limit concurrent animations** to maintain 60fps
3. **Use `transform` and `opacity`** for best performance
4. **Clean up animations** in useEffect cleanup functions

🐛 Troubleshooting

Common Issues

1. **Animations not working**: Ensure GSAP is properly installed and imported
2. **Performance issues**: Reduce animation complexity or use `will-change` CSS
3. **Text not splitting**: Check that the text prop is a string
4. **ScrollTrigger not working**: Ensure ScrollTrigger plugin is registered

Debug Mode

```
// Add debug mode to see animation timelines
<AnimatedText
  text="Debug Mode"
  options={{
    onStart: () => console.log('Animation started'),
    onComplete: () => console.log('Animation completed')
  }}
/>
```

Advanced Usage

Custom Animation Sequences

```
import { gsap } from 'gsap';

function CustomAnimation() {
  const textRef = useRef(null);

  useEffect(() => {
    const tl = gsap.timeline();
    tl.to(textRef.current, { opacity: 1, duration: 1 })
      .to(textRef.current, { scale: 1.1, duration: 0.5 })
      .to(textRef.current, { scale: 1, duration: 0.5 });
  }, []);

  return <span ref={textRef}>Custom Animation</span>;
}
```

Scroll-Triggered Animations

```
<AnimationController
  scrollTrigger={true}
  trigger="onLoad"
>
  <YourContent />
</AnimationController>
```

Contributing

1. Fork the repository
2. Create a feature branch
3. Add your animation component
4. Update documentation
5. Submit a pull request

License

MIT License - feel free to use in your projects!

Happy Animating! 🎉