Complete Implementation Guide: Locomotive-Style Kinetic Typography & Mouse Effects

Overview

The Locomotive careers page features two sophisticated web effects:

1. **Kinetic Typography**: Text elements that cycle through with different entrance and exit animations
2. **Velocity-Based Image Spawning**: Images appear near the cursor based on mouse movement speed

1. Kinetic Typography Implementation

Core Concept

Each text phrase has three animation phases:

- **Entrance**: Text slides in from a specific direction
- **Display**: Text remains visible for a duration
- **Exit**: Text slides out in a different direction

HTML Structure

```html
<div class="kinetic-container">
  <div class="text-wrapper" data-text="GOOD TIMES" data-enter="left" data-exit="right">
    <h2>GOOD TIMES</h2>
  </div>
  <div class="text-wrapper" data-text="ANNUAL TRIP" data-enter="top" data-exit="bottom">
    <h2>ANNUAL TRIP</h2>
  </div>
  <div class="text-wrapper" data-text="GOOD PEOPLE" data-enter="right" data-exit="left">
    <h2>GOOD PEOPLE</h2>
  </div>
</div>
```

CSS Implementation

CSS

```css
.kinetic-container {
  position: relative;
  width: 100vw;
  height: 100vh;
  display: flex;
  align-items: center;
  justify-content: center;
  overflow: hidden;
}

.text-wrapper {
  position: absolute;
  font-size: 4rem;
  font-weight: 900;
  letter-spacing: 0.1em;
  white-space: nowrap;
  opacity: 0;
  transition: all 1s cubic-bezier(0.25, 0.46, 0.45, 0.94);
}

/* Entrance animations */
.text-wrapper[data-enter="left"] {
  transform: translateX(-100%);
}

.text-wrapper[data-enter="right"] {
  transform: translateX(100%);
}

.text-wrapper[data-enter="top"] {
  transform: translateY(-100%);
}

.text-wrapper[data-enter="bottom"] {
  transform: translateY(100%);
}

/* Active state */
.text-wrapper.active {
  opacity: 1;
  transform: translate(0, 0);
}
```

```css
/* Exit animations */
.text-wrapper.exiting[data-exit="left"] {
  transform: translateX(-100%);
  opacity: 0;
}

.text-wrapper.exiting[data-exit="right"] {
  transform: translateX(100%);
  opacity: 0;
}

.text-wrapper.exiting[data-exit="top"] {
  transform: translateY(-100%);
  opacity: 0;
}

.text-wrapper.exiting[data-exit="bottom"] {
  transform: translateY(100%);
  opacity: 0;
}
```

JavaScript Control

javascript

```
class KineticTypography {
  constructor(container) {
    this.container = container;
    this.textWrappers = container.querySelectorAll('.text-wrapper');
    this.currentIndex = 0;
    this.displayDuration = 4000; // 4 seconds per text
    this.transitionDuration = 1000; // 1 second transition

    this.start();
  }

  start() {
    this.showText(this.currentIndex);
    setInterval(() => {
      this.nextText();
    }, this.displayDuration + this.transitionDuration);
  }

  showText(index) {
    const wrapper = this.textWrappers[index];
    wrapper.classList.add('active');
    wrapper.classList.remove('exiting');
  }

  hideText(index) {
    const wrapper = this.textWrappers[index];
    wrapper.classList.add('exiting');
    wrapper.classList.remove('active');
  }

  nextText() {
    // Hide current text
    this.hideText(this.currentIndex);

    // Show next text after transition
    setTimeout(() => {
      this.currentIndex = (this.currentIndex + 1) % this.textWrappers.length;
      this.showText(this.currentIndex);
    }, this.transitionDuration);
  }
}

// Initialize
```

```
const kineticsContainer = document.querySelector('.kinetic-container');
new KineticTypography(kineticsContainer);
```

2. Velocity-Based Image Spawning

Core Concept

- Track mouse movement and calculate velocity
- Spawn images when velocity exceeds threshold
- Number of images spawned correlates with speed
- Images have random positioning, rotation, and fade out

HTML Structure

```
html
<div class="mouse-effect-container" id="mouseContainer">
  <!-- Images will be spawned here dynamically -->
</div>

<!-- Hidden image pool for performance -->
<div class="image-pool" style="display: none;">
  <img src="image1.jpg" alt="">
  <img src="image2.jpg" alt="">
  <img src="image3.jpg" alt="">
  <!-- Add more images -->
</div>
```

CSS for Spawned Images

css

```css
.mouse-effect-container {
  position: fixed;
  top: 0;
  left: 0;
  width: 100vw;
  height: 100vh;
  pointer-events: none;
  z-index: 1000;
}

.spawned-image {
  position: absolute;
  width: 80px;
  height: 80px;
  object-fit: cover;
  border-radius: 50%;
  border: 2px solid white;
  box-shadow: 0 4px 20px rgba(0,0,0,0.3);
  pointer-events: none;
  transition: all 1.5s ease-out;
  transform-origin: center;
}

.spawned-image.fade-out {
  opacity: 0;
  transform: scale(0.5) rotate(180deg);
}
```

JavaScript Implementation

javascript

```
class MouseImageEffect {
  constructor(container, imagePool) {
    this.container = container;
    this.imagePool = Array.from(imagePool.querySelectorAll('img'));
    this.lastMousePosition = { x: 0, y: 0 };
    this.lastTimestamp = Date.now();
    this.velocityThreshold = 5;
    this.maxImages = 10;
    this.imageCounter = 0;

    this.bindEvents();
  }

  bindEvents() {
    document.addEventListener('mousemove', (e) => {
      this.handleMouseMove(e);
    });
  }

  calculateVelocity(currentPos, currentTime) {
    const deltaX = currentPos.x - this.lastMousePosition.x;
    const deltaY = currentPos.y - this.lastMousePosition.y;
    const deltaTime = currentTime - this.lastTimestamp;

    const distance = Math.sqrt(deltaX * deltaX + deltaY * deltaY);
    const velocity = distance / (deltaTime || 1) * 10; // Scale factor

    return velocity;
  }

  handleMouseMove(e) {
    const currentPos = { x: e.clientX, y: e.clientY };
    const currentTime = Date.now();

    const velocity = this.calculateVelocity(currentPos, currentTime);

    if (velocity > this.velocityThreshold) {
      this.spawnImages(currentPos, velocity);
    }

    this.lastMousePosition = currentPos;
    this.lastTimestamp = currentTime;
  }
```

```
  spawnImages(position, velocity) {
    // Calculate number of images based on velocity
    const numImages = Math.min(
      Math.floor(velocity / 3),
      this.maxImages
    );

    for (let i = 0; i < numImages; i++) {
      this.createImage(position);
    }
  }

  createImage(centerPos) {
    const img = document.createElement('img');
    const randomImage = this.imagePool[
      Math.floor(Math.random() * this.imagePool.length)
    ];

    img.src = randomImage.src;
    img.className = 'spawned-image';
    img.id = `img-${this.imageCounter++}`;

    // Random positioning around mouse
    const offsetX = (Math.random() - 0.5) * 120;
    const offsetY = (Math.random() - 0.5) * 120;
    const rotation = Math.random() * 360;
    const scale = 0.5 + Math.random() * 0.8;

    img.style.left = (centerPos.x + offsetX - 40) + 'px';
    img.style.top = (centerPos.y + offsetY - 40) + 'px';
    img.style.transform = `rotate(${rotation}deg) scale(${scale})`;
    img.style.opacity = '0.8';

    this.container.appendChild(img);

    // Trigger fade out animation
    setTimeout(() => {
      img.classList.add('fade-out');
    }, 100);

    // Remove from DOM
    setTimeout(() => {
      if (img.parentNode) {
```

```
            img.parentNode.removeChild(img);
        }
    }, 1600);
  }
}

// Initialize
const container = document.getElementById('mouseContainer');
const imagePool = document.querySelector('.image-pool');
new MouseImageEffect(container, imagePool);
```

3. Performance Optimizations

For Kinetic Typography:

- Use CSS transforms instead of changing position properties
- Use `transform3d()` to enable hardware acceleration
- Implement `will-change` property for animated elements
- Use `requestAnimationFrame` for smooth animations

For Mouse Effects:

- Limit maximum number of simultaneous images
- Use object pooling for image elements
- Implement throttling for mouse move events
- Clean up DOM elements after animations complete

javascript

```javascript
// Throttled mouse move for better performance
function throttle(func, limit) {
  let inThrottle;
  return function() {
    const args = arguments;
    const context = this;
    if (!inThrottle) {
      func.apply(context, args);
      inThrottle = true;
      setTimeout(() => inThrottle = false, limit);
    }
  }
}

// Apply throttling
document.addEventListener('mousemove', throttle(handleMouseMove, 16)); // ~60fps
```

4. Advanced Enhancements

Kinetic Typography:

- Add easing functions for more natural movement
- Implement text splitting for character-by-character animations
- Add parallax effects for depth
- Include text morphing between words

Mouse Effects:

- Add physics simulation for image movement
- Implement magnetic effects near other elements
- Create image trails with fading opacity
- Add sound effects triggered by velocity

5. Browser Compatibility

- Use CSS vendor prefixes for transforms
- Provide fallbacks for older browsers
- Test performance on mobile devices
- Consider reduced motion preferences

```css
@media (prefers-reduced-motion: reduce) {
  .text-wrapper {
    transition: opacity 0.3s ease;
  }

  .spawned-image {
    animation: none;
    transition: opacity 0.5s ease;
  }
}
```

6. Implementation Checklist

Kinetic Typography:

- [ ] HTML structure with data attributes
- [ ] CSS animations for entrance/exit
- [ ] JavaScript loop control
- [ ] Smooth transitions between texts
- [ ] Responsive font sizing

Mouse Effects:

- [ ] Mouse movement tracking
- [ ] Velocity calculation
- [ ] Image spawning logic
- [ ] Performance optimization
- [ ] Mobile touch support

General:

- [ ] Cross-browser testing
- [ ] Performance monitoring
- [ ] Accessibility considerations
- [ ] Mobile responsiveness
- [ ] Loading optimization

This implementation provides a solid foundation for recreating the Locomotive careers page effects while maintaining good performance and user experience.