Peak Sales Website Documentation

Overview

The Peak Sales website is a modern, responsive business website built using HTML5, Bootstrap 5, and various JavaScript libraries. The site features animated elements, interactive components, and a professional design focused on showcasing sales services.

Technical Stack

- **HTML5** - Core structure
- **Bootstrap 5.3.0** - Frontend framework
- **JavaScript** - Interactivity and animations
- **CSS3** - Custom styling

Key Libraries

1. Bootstrap Icons 1.10.5
2. Font Awesome 6.0.0-beta3
3. AOS (Animate On Scroll) 2.3.1
4. GSAP (GreenSock Animation Platform)
5. Typed.js 2.0.12

Site Structure

Header Section

- Meta tags for SEO optimization
- Responsive viewport configuration
- External CSS and JavaScript resource links
- Custom styling for hero section and animations

Navigation

- Responsive navbar with collapsible menu
- Logo integration
- Navigation links: Home, About, Contact

Hero Section

Features:

- Animated typing effect for tagline

- Responsive layout with two columns
- Call-to-action button
- Animated scroll-down indicator
- AOS (Animate On Scroll) integration

Services Section

Components:

- Three service cards:
    1. Sales Team Support
    2. Lead Closing
    3. Training Programs
- Dynamic content panel
- Interactive card selection system

Testimonials Section

- Dynamic testimonial loading
- Responsive grid layout
- GSAP animations for smooth scrolling effects

Call-to-Action Section

- Gradient background
- Centered content layout
- Large action button

Footer

Structure:

- Company overview with logo
- Quick links navigation
- Contact and social media links
- Legal information
- Copyright notice

Animations and Interactive Elements

GSAP Animations

```
// Hero Section Animation
gsap.to("#hero-section", {
    opacity: 1,
    y: 0,
    duration: 1,
    scrollTrigger: {
        trigger: "#hero-section",
        start: "top 80%"
    }
});
```

Typing Animation

```
var typed = new Typed(".typing", {
    strings: ["Success", "Victory", "Peak", "Win"],
    typeSpeed: 100,
    backSpeed: 80,
    loop: true,
    smartBackspace: true,
    showCursor: true,
    cursorChar: '|'
});
```

Special Features

Responsive Design

- Mobile-first approach
- Breakpoint considerations:
    - Mobile: < 768px
    - Tablet: 768px - 992px
    - Desktop: > 992px

Performance Optimizations

1. Deferred script loading
2. CDN usage for libraries
3. Optimized image loading
4. Efficient animation triggers

Styling

Color Scheme

- Gradient backgrounds

- Light text on dark backgrounds
- Consistent brand colors throughout

Typography

- Responsive font sizing
- Custom letter spacing
- Font weight variations for emphasis

Component Styles

1. Buttons
    - Custom hover effects
    - Shadow effects
    - Transition animations
2. Cards
    - Border radius
    - Box shadows
    - Hover state transformations

Maintenance Notes

SEO Considerations

- Meta descriptions implemented
- Alt tags for images
- Semantic HTML structure
- Robots meta tag configuration

Accessibility Features

- ARIA labels
- Semantic HTML elements
- Keyboard navigation support
- Focus states for interactive elements

File Structure

```
File Structure
root/
│
├── assets/
│   ├── css/
│   │   └── styles.css
│   ├── js/
│   │   └── scripts.js
│   └── images/
│       ├── logo.png
│       ├── logo-white.png
│       └── hero-bg.png
│
└── index.html
```

Integration Guidelines

Adding New Services

1. Add new service card to services section
2. Create corresponding content panel
3. Update JavaScript event listeners
4. Add necessary styling

Modifying Animations

1. Locate animation in GSAP configuration
2. Adjust timing and effects as needed
3. Test across different screen sizes
4. Ensure performance isn't impacted

Content Updates

1. Modify HTML content directly
2. Update meta tags if necessary
3. Maintain consistent styling
4. Test responsive behavior

Browser Support

- Chrome (latest)
- Firefox (latest)
- Safari (latest)
- Edge (latest)
- Mobile browsers

CSS Documentation

Overview

This CSS file implements a modern, responsive design system with advanced gradients, enhanced typography, and dark mode aesthetics. The stylesheet follows a modular approach with carefully crafted components and utility classes.

Color System

Base Color Variables

```css
:root {
    --primary-color: #5433ff
    --secondary-color: #20bdff
    --accent-color: #20bdff
    --dark-bg: #121212
    --light-text: rgba(255, 255, 255, 0.87)
}
```

Typography

Base Font Stack

- Primary: 'Inter'
- Fallbacks: 'Segoe UI', Roboto, system-ui, -apple-system, BlinkMacSystemFont, 'Helvetica Neue', sans-serif

Responsive Type Scale

- Desktop Display: 4rem
- Mobile Display: 2.5rem
- Body Text: 16px
- Line Height: 1.6

Custom Scrollbar Implementation

Webkit Scrollbar

- Width: 12px (vertical)

- Height: 12px (horizontal)
- Track: Dark grey (#333333)
- Thumb: Black with white border
- Hover effects included

Firefox Scrollbar

- Width: thin
- Colors: Black thumb, dark grey track

Mobile Considerations

- Hidden scrollbar on devices < 768px
- Optimized for touch interactions

Component Documentation

1. Navigation Bar

Desktop Version (>992px)

- Transparent background
- Position: absolute
- Scroll-activated background change
- Animated underline effects
- Logo dimensions: 30px height

Mobile Version (<991px)

- Gradient background
- Fixed padding
- Collapsed menu support

2. Hero Section

- Full viewport height
- Flexible layout with centered content
- Gradient background animation
- Responsive text sizing
- Z-index management for overlays

3. Services Section

Layout

```
.services-section {
    min-height: 100vh
    background: var(--dark-bg)
    padding: 80px 0
}
```

Service Cards

- Glass morphism effect
- Hover animations
- Active state styling
- Border radius: 24px
- Transition effects

Content Panel

- Sticky positioning
- Overflow management
- Glass morphism styling
- Max height: 80vh

4. Testimonials Section

Card Design

- White background
- Box shadow animation
- Author image sizing: 48px
- Hover transform effect

Grid Layout

- Two-column desktop layout
- Single column mobile
- Gap: 2rem

5. CTA Section

- Gradient background
- Centered text alignment
- Large heading (3rem)
- Responsive padding

6. Footer

Desktop Grid (>992px)

- 4-column layout
- Max width: 1200px
- Consistent spacing

Tablet Layout (600px-992px)

- Maintained 4-column grid
- Adjusted spacing
- Full-width buttons

Mobile Layout (<600px)

- 2-column grid
- Logo spans full width
- Reduced spacing
- Centered alignment

Animation System

Gradient Animation

```
@keyframes gradientFlow {
    0% { background-position: 0% 50% }
    50% { background-position: 100% 50% }
    100% { background-position: 0% 50% }
}
```

Transition Effects

- Duration: 0.3s
- Timing: ease
- Properties: all, transform, background-color

Utility Classes

Text Utilities

- `.text-accent`: Accent color text
- Responsive typography classes

Button Utilities

- `.btn-primary`: Gradient background
- Hover transform effect
- Shadow animation

Responsive Breakpoints

Major Breakpoints

1. Mobile: < 600px
2. Tablet: 600px - 992px
3. Desktop: > 992px
4. Large Desktop: > 1200px

Special Considerations

- Mobile-first approach
- Fluid typography
- Grid system adjustments
- Component-specific breakpoints

Performance Optimizations

Transitions

- GPU-accelerated properties
- Minimal repaints
- Efficient animations

Media Queries

- Logical grouping
- Progressive enhancement
- Device-specific optimizations

Best Practices Implementation

Naming Conventions

- BEM-inspired class naming
- Semantic class names
- Consistent prefixing

Code Organization

1. Global variables
2. Base styles
3. Component styles
4. Utilities
5. Media queries

Accessibility

- High contrast ratios
- Focus states
- Screen reader support
- Keyboard navigation

Browser Support

- Chrome (latest)
- Firefox (latest)
- Safari (latest)
- Edge (latest)
- Mobile browsers

Maintenance Guidelines

Adding New Components

1. Follow existing component structure
2. Maintain consistent naming
3. Include responsive styles
4. Document new additions

Modifying Existing Styles

1. Check dependencies
2. Test all breakpoints
3. Verify accessibility
4. Update documentation

CSS Variables

- Use existing color scheme
- Maintain dark theme compatibility
- Follow naming convention

JavaScript Documentation

Overview

This JavaScript codebase manages the interactive features of the Peak Sales website, including service content management, animations, Easter egg functionality, and testimonial loading. The code is organized using modern JavaScript practices and follows an object-oriented approach.

Core Components

1. Service Content Management

Content Structure

```
const serviceContent = {
    salesSupport: {...},
    leadClosing: {...},
    trainingPrograms: {...}
  }
```

Each service contains:

- Title
- HTML content
- Icon configuration
- Feature lists

2. ServicesManager Class

Constructor

```
class ServicesManager {
    constructor() {
      this.serviceCards = document.querySelectorAll('.service-card');
      this.servicesContainer = document.querySelector('.services-container');
      this.contentWrapper = document.querySelector('.content-wrapper');
      this.servicesSection = document.querySelector('.services-section');
      this.defaultContent = document.getElementById('defaultContent');
      this.activeCard = null;
      this.isIntersectionObserverSupported = 'IntersectionObserver' in window;
      this.MOBILE_BREAKPOINT = 768;
      this.cardFocusIndex = 0;
    }
  }
```

Key Methods

1. `init()`
    - Initializes content panels
    - Sets up observers
    - Configures event listeners
    - Enables smooth scrolling
2. `setupContentPanels()`
    - Creates DOM fragments
    - Populates content dynamically
    - Manages panel rendering
3. `setupIntersectionObserver()`
    - Handles section visibility
    - Manages card activation
    - Configures threshold settings
4. `setupEventListeners()`
    - Click handling
    - Keyboard navigation
    - Resize management

3. Animation System

GSAP Integration

```
animateCard(card) {
    if (window.innerWidth > this.MOBILE_BREAKPOINT) return;

    gsap.to(card, {
      opacity: 1,
      y: 0,
      duration: 0.5,
      ease: 'power2.out'
    });
  }
```

Mobile Animations

```
setupMobileAnimations() {
    gsap.utils.toArray('.service-card').forEach((card, index) => {
      gsap.set(card, { opacity: 0, y: 20 });
      gsap.to(card, {
        opacity: 1,
        y: 0,
        duration: 0.5,
        ease: 'power2.out',
        scrollTrigger: {...}
      });
    });
  }
```

4. Testimonials System

Data Loading

```
fetch("assets/data/testimonials.json")
  .then(response => response.json())
  .then(data => {
    // Testimonial rendering logic
  })
```

Card Generation

- Dynamic HTML creation
- AOS animation integration
- Staggered loading effects

Accessibility Features

Keyboard Navigation

1. Enter/Space: Activate cards
2. Arrow Keys: Navigate between cards
3. Escape: Close Easter egg

ARIA Attributes

```
card.setAttribute('aria-selected', 'true');
closeButton.setAttribute('aria-label', 'Close');
```

Performance Optimizations

1. Debouncing

```
debounce(func, wait) {
    let timeout;
    return function executedFunction(...args) {
      const later = () => {
        clearTimeout(timeout);
        func(...args);
      };
      clearTimeout(timeout);
      timeout = setTimeout(later, wait);
    };
  }
```

2. Fragment Usage

```
const fragment = document.createDocumentFragment();
// Population logic
this.contentWrapper.appendChild(fragment);
```

3. Intersection Observer

- Efficient scroll handling
- Optimized animation triggers
- Resource management

Event Handling System

Document Ready Events

```
document.addEventListener('DOMContentLoaded', () => {
    new ServicesManager();
    // Additional initialization
  });
```

User Interaction Events

1. Click handling
2. Keyboard input
3. Scroll management
4. Resize response

Mobile Responsiveness

Breakpoint Management

```
const MOBILE_BREAKPOINT = 768;
```

Mobile-Specific Features

1. Modified animations
2. Adjusted scroll behavior
3. Touch-friendly interactions

Error Handling

Fetch Operations

```
fetch("assets/data/testimonials.json")
  .catch(error => console.error("Error loading testimonials:", error));
```

Feature Detection

```
this.isIntersectionObserverSupported = 'IntersectionObserver' in window;
```

Best Practices Implementation

Code Organization

1. Constants at top
2. Class-based structure
3. Method grouping
4. Event delegation

Performance Considerations

1. Debounced resize handlers
2. Efficient DOM updates
3. Animation optimizations
4. Resource lazy loading

Maintenance Guidelines

Adding New Services

1. Update serviceContent object
2. Add corresponding HTML
3. Configure animations
4. Test interactions

Modifying Animations

1. Adjust GSAP configurations
2. Update trigger points
3. Test performance impact
4. Verify mobile behavior

Code Updates

1. Maintain class structure
2. Follow naming conventions
3. Update documentation
4. Test cross-browser

Browser Support

- Modern browsers with IntersectionObserver
- Fallback handling for older browsers
- Mobile browser optimization