README

SCORM Learning Object Development Framework

Overview

This framework enables learning designers to create custom, interactive learning objects using Claude Code and deploy them as SCORM 1.2 packages to any LMS. Built specifically for **higher education** with focus on **scalability**, **engagement**, and **cost-effectiveness**.

Why This Framework?

For Learning Designers

- **Full creative control** - No limitations of authoring tools
- **Reusable components** - Build once, customize many times
- **Fast iteration** - Test locally before LMS upload
- **Version control** - Track changes with Git
- **Work with Claude Code** - AI-assisted development

For Higher Ed Institutions

- **Scalability** - Deploy across multiple courses/campuses
- **Cost-effective** - No per-seat authoring tool licenses
- **Unique engagement** - Stand out from standard content
- **Better margins** - Faster development = lower costs
- **Accessibility built-in** - WCAG 2.1 AA compliant

For IT Teams

- **Standard SCORM 1.2** - Works with any LMS
- **Automated validation** - Catch issues before deployment
- **Clear documentation** - Troubleshooting guides included
- **Predictable packages** - Consistent structure every time

Quick Start

1. Project Structure

```
scorm/
├── templates/              # Ready-to-use learning object templates
│   ├── drag-drop-template.html
│   ├── matching-template.html
│   ├── hotspot-template.html
│   ├── accessible-component-base.html
│   └── scorm-wrapper.js   # SCORM API handler
├── testing/               # Validation and testing tools
│   ├── scorm-validator.html
│   ├── test-checklist.md
│   └── browser-compat-test.html
├── docs/                  # Documentation
│   ├── IT-REQUIREMENTS.md
│   ├── LMS-SPECIFICATIONS-TEMPLATE.md
│   ├── TROUBLESHOOTING.md
│   ├── HIGHER-ED-STRATEGY.md
│   └── ACCESSIBILITY-GUIDE.md
├── scripts/               # Build automation
│   ├── build-scorm.js
│   └── validate-package.js
├── example-module/        # Working sample module
└── dist/                  # Generated SCORM packages (ZIP files)
```

2. Development Workflow

```
Learning Designer → Claude Code → Local Test → Build Package → IT
Validation → LMS Deploy
```

Step-by-Step Process:

STEP 1: Design Your Learning Object - Choose a template from `templates/` - Define learning objectives and interactions - Gather assets (images, videos, text content)

STEP 2: Customize with Claude Code - Open template in Claude Code - Describe desired changes (colors, content, behavior) - Claude modifies HTML/CSS/JavaScript - Preview in browser immediately

STEP 3: Local Testing - Open `testing/scorm-validator.html` in browser - Load your learning object - Verify SCORM API calls working - Test all interactions and paths

STEP 4: Build SCORM Package

```
npm run build
```

- Creates ZIP file with imsmanifest.xml
- Validates structure automatically
- Outputs to `dist/` folder

STEP 5: IT Validation - Hand off ZIP to IT team - IT uses `test-checklist.md` for validation - Test in LMS staging environment

STEP 6: Deploy to Production - Upload to production LMS - Monitor learner data and engagement - Iterate based on analytics

---

For Learning Designers

Using Templates

Each template is **self-contained** - everything you need in one HTML file: - SCORM tracking built-in - Responsive design (works on mobile) - Accessibility features included - Professional styling with Tailwind CSS

Customization Without Coding

You can ask Claude Code to: - "Change the color scheme to blue and gold (school colors)" - "Add 3 more matching pairs to this exercise" - "Make the feedback messages more encouraging" - "Add a timer to this activity" - "Increase font size for better readability"

What You Control

Content: - Text, images, videos - Question/answer pairs - Feedback messages - Instructions

Design: - Colors and branding - Layout and spacing - Fonts and typography - Animations and transitions

Behavior: - Scoring logic - Completion criteria - Branching paths - Feedback timing

---

For IT Teams

Prerequisites

Required: - Node.js 16+ (for build scripts) - Modern web browser (Chrome, Firefox, Safari, Edge) - Text editor or Claude Code

LMS Requirements: - SCORM 1.2 support - JavaScript enabled - HTML5 support

Validation Process

Before deploying any package:

1. **Automated Check:**

```
node scripts/validate-package.js dist/your-module.zip
```

2. **Manual Testing:**
   - Follow `testing/test-checklist.md`
   - Test in multiple browsers
   - Verify SCORM tracking
3. **LMS Staging:**
   - Upload to test environment
   - Complete full learner path
   - Check gradebook/reporting

Common Issues

See `docs/TROUBLESHOOTING.md` for solutions to: - Package import failures - Blank screens after launch - SCORM tracking not working - Mobile compatibility issues - Browser-specific bugs

Security Considerations

All packages are: - ✅ Self-contained (no external dependencies) - ✅ No server-side code required - ✅ No database connections - ✅ No personal data collection (only SCORM standard data) - ✅ XSS-safe (sanitized inputs)

Component Library

1. Drag-and-Drop (`drag-drop-template.html`)

Best for: - Categorization activities - Sequencing tasks - Labeling diagrams

Features: - Touch-friendly (works on tablets) - Snap-to-grid positioning - Visual feedback on correct/incorrect - Unlimited items/categories

Customization: - Add/remove items - Change categories - Adjust difficulty (hints, tries)

---

2. Matching Exercise (`matching-template.html`)

Best for: - Vocabulary building - Concept pairing - Term/definition matching

Features: - Click or drag to connect - Shuffle on each attempt - Partial credit scoring - Timed or untimed modes

Customization: - Number of pairs - Scoring rules - Feedback messages - Visual theme

---

3. Hotspot Interaction (`hotspot-template.html`)

Best for: - Image-based learning - Anatomy/geography - Equipment identification - Process diagrams

Features: - Multiple hotspot types (click, hover) - Layered feedback - Progressive reveal - Scoring by accuracy

Customization: - Upload any image - Define hotspot regions - Custom feedback per spot - Branching scenarios

---

4. Accessible Base (`accessible-component-base.html`)

Use this as starting point for: - Custom interactions - Text-heavy content - Unique activity types

Features: - WCAG 2.1 AA compliant - Screen reader tested - Keyboard navigation - High contrast support - Skip links and ARIA labels

---

Analytics & Learning Outcomes

What Gets Tracked (SCORM 1.2 Standard)

Automatically captured: - Completion status (complete/incomplete) - Score (0-100) - Time spent - Number of attempts - Success/failure status

Custom tracking available: - Which distractors chosen - Time per question - Path through content - Interaction patterns

Measuring Engagement

Higher engagement indicators: - ✅ Multiple attempts (learner persistence) - ✅ Time on task (deep engagement) - ✅ Completion rate (finishing activity) - ✅ Score improvement (learning happening)

Lower engagement indicators: - ⚠️ Quick exit (< 30 seconds) - ⚠️ No interaction clicks - ⚠️ Incomplete status - ⚠️ Single attempt with low score

Optimization Tips

To increase engagement: 1. Add immediate feedback (not just at end) 2. Use gamification (points, progress bars) 3. Provide hints/scaffolding 4. Allow multiple attempts 5. Make it visually appealing 6. Keep activities short (5-10 min max)

Cost & Scalability

Development Costs

Initial setup (one-time): - Framework setup: 2-4 hours - IT validation: 1-2 hours - Designer training: 2-3 hours

Per learning object: - Template customization: 30-60 min - Content creation: varies by complexity - Testing & QA: 20-30 min - **Total: ~2-3 hours per unique interaction**

Scaling Strategy

Year 1: Build Component Library - Create 10-15 core templates - Cover common interaction types - Establish quality standards

Year 2: Deploy at Scale - Reuse/remix existing components - 5-10 min per customization - **10x faster than Year 1**

Year 3: Competitive Advantage - Unique interaction library - Consistent brand experience - Rapid course development - **Significant cost savings**

ROI Calculation

Traditional approach (Storyline): - License: $1,398/year per designer - Learning curve: 20+ hours - Development time: 4-6 hours per interaction - Customization limits: High

This framework: - Cost: $0 (open source tools) - Learning curve: 2-3 hours - Development time: 2-3 hours first, 30 min after - Customization: Unlimited

Break-even: After creating ~5-10 learning objects

Higher Ed Specific Advantages

1. Accessibility Compliance

- ADA requirements built-in
- Section 508 compliant
- WCAG 2.1 AA tested
- **Reduce legal risk**

2. Multi-Campus Deployment

- Same package works everywhere
- Consistent quality across locations
- Centralized updates
- **Economies of scale**

3. Student Engagement

- Interactive beats passive video
- Immediate feedback increases retention
- Mobile-friendly for modern learners
- **Better learning outcomes**

4. Faculty Adoption

- Easy to integrate into existing courses
- No faculty training required
- Works in any LMS
- **Low adoption friction**

5. Competitive Differentiation

- Stand out from competitors
- "Interactive learning experiences"
- Modern, polished content
- **Marketing advantage**

Next Steps

For Learning Teams

1. Read `docs/HIGHER-ED-STRATEGY.md`
2. Explore `example-module/`
3. Try customizing `templates/matching-template.html`
4. Review `docs/ACCESSIBILITY-GUIDE.md`

For IT Teams

1. Complete `docs/IT-REQUIREMENTS.md` questionnaire
2. Fill out `docs/LMS-SPECIFICATIONS-TEMPLATE.md`
3. Set up staging environment for testing
4. Review `docs/TROUBLESHOOTING.md`

For Administrators

1. Review cost analysis above
2. Read `docs/HIGHER-ED-STRATEGY.md`
3. Plan pilot project (1-2 courses)
4. Establish success metrics

Support & Resources

Documentation

- All docs in `docs/` folder
- Troubleshooting guide included
- Code comments in every template

Testing Tools

- SCORM validator: `testing/scorm-validator.html`
- Browser test: `testing/browser-compat-test.html`
- Test checklist: `testing/test-checklist.md`

Getting Help

- Check `docs/TROUBLESHOOTING.md` first
- Review template comments for examples
- Use Claude Code for customization help

Version Control Best Practices

Using Git

```
# Initial setup
git init
git add .
git commit -m "Initial SCORM framework setup"

# Before making changes
git checkout -b feature/new-activity-type

# After changes
git add .
git commit -m "Add biology matching exercise"

# If something breaks
git revert HEAD
```

Benefits

- Track all changes over time
- Revert to working versions
- Collaborate with team members
- Maintain multiple course versions

License & Attribution

This framework uses: - **pipwerks SCORM Wrapper** (MIT License) - SCORM API handling - **Tailwind CSS** (MIT License) - Styling framework - Templates created for educational use

You are free to: - Use for commercial purposes - Modify templates - Create derivative works - Share with colleagues

---

Questions?

For Learning Designers: - Not sure which template to use? Start with `example-module/` - Need custom interaction? Ask Claude Code to build from `accessible-component-base.html`

For IT Teams: - Complete `docs/IT-REQUIREMENTS.md` first - Check `docs/TROUBLESHOOTING.md` for common issues

For Strategy/Planning: - Review `docs/HIGHER-ED-STRATEGY.md` for scaling approach - See cost analysis section above for ROI

Last Updated: 2025-10-28 **Framework Version:** 1.0 **SCORM Version:** 1.2