Coding Guidelines Summary for Paul Chatterton's Portfolio Website

> **For beginners**: This document explains the coding rules we'll follow to build a website that *never breaks*, even when some data is missing.

🎯 The Big Idea: "Defensive Coding"

Think of it like wearing a seatbelt—you hope you won't need it, but it keeps you safe just in case.

The problem we're solving: - The Sanity CMS has 170+ documents (publications, projects, media, etc.) - Some documents are incomplete (missing images, dates, descriptions) - Older content may never have certain fields (e.g., 1980s publications won't have DOIs)

The solution: Write code that assumes data *might* be missing, and handles it gracefully instead of crashing.

🔑 The 7 Essential Patterns

Pattern 1: Always Have a Backup Value

When displaying text or numbers, always provide a fallback.

```
// ❌ BAD - crashes if title is missing
<h2>{publication.title}</h2>

// ✅ GOOD - shows "Untitled" if title is missing
<h2>{publication.title || 'Untitled'}</h2>
```

Plain English: "Show the title, *but if there isn't one*, show 'Untitled' instead."

Pattern 2: Safe Navigation with `?.`

Use `?.` to safely access nested data.

```
// ❌ BAD - crashes if author is missing
```

```
<p>{publication.author.name}</p>

// ✅ GOOD - safely checks each step
<p>{publication.author?.name || 'Unknown author'}</p>
```

Plain English: "Try to get the author's name, *but if author doesn't exist*, don't crash—just show 'Unknown author'."

Pattern 3: Only Show What Exists

Hide sections when their data is missing.

```
// ✗ BAD - shows broken/empty link
<a href={publication.doi}>Read Paper</a>

// ✅ GOOD - only shows link if DOI exists
{publication.doi && (
  <a href={publication.doi}>Read Paper</a>
)}
```

Plain English: "Only show the 'Read Paper' link *if* there's actually a DOI to link to."

Pattern 4: Safely Loop Through Lists

Always check that a list exists before looping.

```
// ✗ BAD - crashes if tags is missing
{publication.tags.map(tag => <span>{tag}</span>)}

// ✅ GOOD - uses empty array as fallback
{(publication.tags || []).map(tag => <span key={tag}>{tag}</span>)}

// EVEN BETTER - only show if there are tags
{publication.tags?.length > 0 && (
  <div className="tags">
    {publication.tags.map(tag => <span key={tag}>{tag}</span>)}
  </div>
)}
```

Plain English: "Show tags *only if* we have some. If the tags list is missing or empty, don't show anything."

Pattern 5: Handle Numbers Carefully

Zero (0) is a valid number—don't hide it by mistake!

```
// ✗ BAD - hides "0 citations" because 0 is falsy
{publication.citations && <p>Citations: {publication.citations}</p>}

// ✅ GOOD - only hides if truly missing
{typeof publication.citations === 'number' && (
  <p>Citations: {publication.citations}</p>
)}
```

Plain English: "Show the citation count *if* it's a number (including zero)."

Pattern 6: Image Fallbacks

Never show broken image icons—provide a placeholder instead.

```
// ✗ BAD - shows ugly broken image icon
<img src={publication.coverImage?.url} />

// ✅ GOOD - shows placeholder when no image
{publication.coverImage?.url ? (
  <img src={publication.coverImage.url} alt={publication.title ||
        'Publication'} />
) : (
  <div className="placeholder">📄</div>
)}
```

Plain English: "Show the image if we have one; otherwise, show a nice placeholder icon."

Pattern 7: Handle Dates Safely

Dates can be missing or invalid—check before formatting.

```
// ✗ BAD - crashes on invalid dates
<p>{new Date(publication.date).toLocaleDateString()}</p>

// ✅ GOOD - checks first, provides fallback
<p>
  {publication.date
    ? new Date(publication.date).toLocaleDateString()
    : 'Date TBA'}
</p>
```

Plain English: "Format and show the date *if* we have one; otherwise show 'Date TBA'."

🔒 Security Rules (Super Important!)

What’s Safe to Put in Code

✅ Safe (Public)	❌ Never Put in Code
Sanity Project ID	Write tokens (`sk_...`)
Dataset name (“production”)	Admin passwords
API version	Secret keys

Why This Matters

The website code is visible to anyone who views the page source. Sensitive credentials would be exposed!

The Safe Setup

```
const client = createClient({
  projectId: 'abc123xyz',  // ✅ Safe - public identifier
  dataset: 'production',    // ✅ Safe - public name
  useCdn: true,             // ✅ Uses fast CDN
  apiVersion: '2024-01-01', // ✅ Safe - API version
  // NO TOKEN = read-only public access ✅
});
```

📋 Quick Checklist Before Writing Any Component

Data Access Safety

- ☐ Every property uses `?.` for safe navigation
- ☐ Every text/number has `|| fallback`
- ☐ Every array uses `(array || [])` or checks `.length > 0`
- ☐ Every image has a fallback placeholder

Rendering Safety

- ☐ Optional sections use `{condition && (<element>)}`
- ☐ Numbers use `typeof x === 'number'`
- ☐ No raw `undefined` or `null` shown to users

Component Safety

- ☐ Has loading state (“Loading...”)
- ☐ Has error state (“Something went wrong”)
- ☐ Has empty state (“No publications found”)

🛠 Useful Helper Functions

These are ready-to-use utilities from the templates:

```
// Format a date safely
formatDate(dateString, fallback = 'Date unknown')

// Format a year with fallback
formatYear(year)  // Returns "Year unknown" if missing

// Shorten long text
truncateText(text, maxLength = 150)

// Format arrays nicely: ["A", "B", "C"] → "A, B, and C"
formatList(items, conjunction = 'and')
```

🏗 How This Applies to Professor Chatterton's Website

Content Types from Sanity

Type	Count	Notes
Publications	124	Mix of complete and incomplete data
Media	19	May have missing dates/descriptions
Projects	9	Some may lack images
Roles	9	Various date ranges
Images	8	Gallery items
Timeline	1	Career milestones

Recommended Approach

1. **All components must be defensive** - Assume any field could be missing
2. **Use the template patterns** - Don't reinvent the wheel
3. **Test with real data** - The Sanity data has gaps, use them to test!
4. **Graceful degradation** - Pages should work with any combination of data

---

📝 Example: A Defensive Publication Card

Here's what a properly defensive component looks like:

```
function PublicationCard({ publication }) {
  // Guard clause - return early if no data
  if (!publication) return null;

  return (
    <article className="publication-card">
      {/* Image with fallback */}
      {publication.coverImage ? (
        <img src={publication.coverImage} alt={publication.title ||
        'Publication'} />
      ) : (
        <div className="placeholder">📄</div>
      )}

      {/* Title with fallback */}
      <h3>{publication.title || 'Untitled Publication'}</h3>

      {/* Year - only show if exists */}
      {publication.year && <span>{publication.year}</span>}

      {/* Authors - array safety */}
      {publication.authors?.length > 0 && (
        <p>By {publication.authors.join(', ')}</p>
      )}

      {/* DOI link - conditional */}
      {publication.doi && (
        <a href={publication.doi}>Read Paper</a>
      )}
    </article>
  );
}
```

✅ Success Criteria

The website is ready when: - ✅ Works with current incomplete data - ✅ Works when data is later completed - ✅ No console errors - ✅ No "undefined" shown to visitors - ✅ No broken images - ✅ No sensitive data exposed

---

🚀 Key Takeaways

1. **Always assume data might be missing** - This isn't pessimism, it's professionalism
2. **Use `?.` and `||` everywhere** - They're your safety net

3. **Test with real (incomplete) data** - Don't just test with perfect mock data
4. **Security matters** - Never put tokens or secrets in frontend code
5. **These patterns are permanent** - This isn't "temporary scaffolding"—it's how professional code is written

---

Document created: January 30, 2026
Based on: frontend-defensive-coding-prompt.md and defensive-component-templates.js