# FRAPPE WEB DEVELOPMENT

# ULTIMATE GUIDE

***The comprehensive handbook for building Web Pages, Web Templates, and Custom Blocks in the Frappe Framework***

---

## Table of Contents



## 1) HTML Rules — Jinja Templates

Frappe leverages the **Jinja2 engine** to render dynamic HTML content. Your code is injected into a global layout containing the Navbar and Footer.

**Do's**

| Do | Description |
|---|---|
| **Content Only** | Write only the page content. Frappe handles the global layout. |
| **Bootstrap Ready** | Use built-in classes: `container`, `row`, `col-md-6`, `btn-primary`, etc. |
| **Semantic HTML** | Use `<section>`, `<article>`, `<nav>` for better SEO & accessibility. |
| **Use Filters** | Apply Jinja filters: `{{ value \| e }}`, `{{ "/path" \| abs_url }}` |

**Don'ts**

| Don't | Why |
|---|---|
| `<html>`, `<head>`, `<body>` | These tags are already provided by Frappe |

| Don't | Why |
|---|---|
| Duplicate `id` attributes | Breaks SPA mode when page loads twice |
| Inline styles | Creates specificity conflicts |
| Hardcoded URLs | Use `{{ "/page" \| abs_url }}` instead |

**Pro Tip:** Always escape user input with `\| e` filter to prevent XSS attacks.

---

## 2) CSS Rules & Scoping

**CRITICAL WARNING:** CSS leaks can break the entire ERPNext Desk UI. Handle with extreme care!

### The Golden Rule

```css
/*  FORBIDDEN - Breaks system UI */
* { margin: 0; padding: 0; }
body { font-size: 16px; }
h1 { color: red; }

/*  CORRECT - Scoped to your wrapper */
.my-wrapper h1 { color: var(--primary-color); }
```

### Namespacing Pattern

```html
<!-- Always wrap in a unique class -->
<div class="my-unique-page">
    <h1>Title</h1>
    <p>Content goes here</p>
</div>

<style>
/* Target only elements inside your wrapper */
.my-unique-page {
    padding: 2rem;
    background: var(--bg-color);
}

.my-unique-page h1 {
    color: var(--text-color);
    font-weight: 600;
```

```
}
</style>
```

### Available CSS Variables

| Variable | Purpose | Example |
|---|---|---|
| `--primary-color` | Main brand color | `#2490E3` |
| `--text-color` | Default text | `#2d3748` |
| `--bg-color` | Page background | `#ffffff` |
| `--card-bg` | Card background | `#f8f9fa` |
| `--border-color` | Borders | `#e2e8f0` |
| `--font-stack` | Typography | System fonts |

---

## 3) JavaScript Rules — Client Side

Frappe provides **jQuery** and **Vanilla JS** support. All code should be wrapped in `frappe.ready()`.

### Initialization

```
//  Always wrap in frappe.ready
frappe.ready(function() {
    console.log("Page loaded and ready");
});

//  Arrow function syntax also supported
frappe.ready(() => {
    initComponents();
});
```

### API Communication

```
// Making a server call
frappe.call({
    method: "my_app.api.get_data",
    args: {
        doctype: "Lead",
        filters: { status: "Open" }
    },
    freeze: true,         // Show loading spinner
    show_spinner: true, // Alternative syntax
    callback: function(response) {
        if (!response.exc) {
```

```
            const data = response.message;
            console.log("Data received:", data);
        }
    },
    error: function(error) {
        frappe.throw(__("Failed to fetch data"));
    }
});
```

### jQuery Integration

```
frappe.ready(() => {
    // jQuery available as $ or frappe$
    $('.my-button').on('click', function() {
        frappe.msgprint("Button clicked!");
    });

    // AJAX with frappe's wrapper
    frappe.call({
        // ... same as above
    });
});
```

### DOM Manipulation Helpers

| Method | Use |
|---|---|
| `frappe.msgprint(__('Message'))` | Show modal dialog |
| `frappe.show_alert({message: __('Done'), indicator: 'green'})` | Toast notification |
| `frappe.throw(__('Error message'))` | Error alert |
| `frappe.confirm(__('Are you sure?'), () => { ... })` | Confirmation dialog |

---

## 4) Web Page Doctype Structure

Organize your code across the three main fields in the **Web Page** doctype:

| Field | Content Type | Notes |
|---|---|---|
| **HTML Template** | HTML + Jinja | Structure, loops, conditionals |
| **Style** | CSS only | No `<style>` tags needed |
| **Script** | JavaScript only | No `<script>` tags needed |

### Field Example

HTML Template

```
<div class="portal-wrapper">
    <h1>{{ doc.page_title }}</h1>
    {% for item in doc.items: %}
        <div class="item">{{ item.name }}</div>
    {% endfor %}
</div>
```

Style

```
.portal-wrapper { padding: 2rem; }
.portal-wrapper h1 { color: var(--primary-color); }
```

Script

```
frappe.ready(() => {
    console.log("Portal loaded");
});
```

---

## 5) Common Mistakes

### Style Bleeding

```
/*  BAD - Overrides system buttons */
.btn { background: red; }

/*  GOOD - Scope to your wrapper */
.my-wrapper .btn { background: var(--primary-color); }
```

### Hardcoded URLs

```
<!--  BAD -->
<a href="http://localhost:8000/about">About</a>

<!--  GOOD -->
<a href="/about">About</a>
<a href="{{ "/about" | abs_url }}">About</a>
```

### Z-Index Overrides

```
/*  BAD - Covers system modals */
.modal { z-index: 999999; }

/*  GOOD - Use reasonable values */
.modal { z-index: 100; }
```

### Synchronous AJAX

```
//  BAD - Freezes the browser
$.ajax({ async: false, ... });

//  GOOD - Async call
$.ajax({ async: true, success: () => {} });
```

### Quick Reference

| Mistake | Solution |
|---|---|
| System styles broken | Always use wrapper classes |
| Links not working on production | Use \| abs_url filter |
| Modals covered | Use z-index < 1000 |
| Page frozen | Use async AJAX calls |

---

## 6) Best Practice Structure

### Complete Example

### HTML

```
<div class="custom-portal-container">
    <section class="hero-section">
        <div class="hero-content">
            <h1>{{ doc.title or "Welcome" }}</h1>
            <p class="text-muted lead">
                {{ doc.subtitle or _("Welcome to our portal") }}
            </p>
```

```
            <div class="hero-actions">
                <button id="submit-btn" class="btn btn-primary btn-lg">
                    <i class="fa fa-paper-plane"></i>
                    {{ _("Get Started") }}
                </button>
            </div>
        </div>
    </section>

    <section class="features-section">
        <!-- Dynamic content here -->
    </section>
</div>
```

### CSS

```
.custom-portal-container {
    max-width: 1200px;
    margin: 0 auto;
    padding: var(--padding, 2rem);
}

.custom-portal-container .hero-section {
    padding: 4rem 2rem;
    text-align: center;
    background: var(--card-bg);
    border-radius: 12px;
    border: 1px solid var(--border-color);
    margin-bottom: 2rem;
}

.custom-portal-container .hero-content h1 {
    font-size: 2.5rem;
    font-weight: 700;
    color: var(--text-color);
    margin-bottom: 1rem;
}

.custom-portal-container .hero-actions {
    margin-top: 2rem;
}

.custom-portal-container .btn-lg {
    padding: 0.75rem 2rem;
    font-size: 1.125rem;
```

```
}
```

**JavaScript**

```
frappe.ready(() => {
    const submitBtn = document.getElementById('submit-btn');

    if (submitBtn) {
        submitBtn.addEventListener('click', () => {
            frappe.confirm(
                __('Are you sure you want to proceed?'),
                () => {
                    // Confirm action
                    processSubmission();
                },
                () => {
                    // Cancel action
                    frappe.show_alert({
                        message: __('Action cancelled'),
                        indicator: 'orange'
                    });
                }
            );
        });
    }

    function processSubmission() {
        frappe.show_alert({
            message: __('Processing your request...'),
            indicator: 'blue'
        });

        frappe.call({
            method: 'my_app.api.process',
            args: { data: getFormData() },
            callback: function(response) {
                if (!response.exc) {
                    frappe.show_alert({
                        message: __('Success!'),
                        indicator: 'green'
                    });
                    setTimeout(() => {
                        window.location.href = '/success';
                    }, 1500);
                }
            }
```

```
        });
    }

    function getFormData() {
        // Collect form data
        return { item: 'value' };
    }
});
```

---

## 7) Advanced Pro Tips

Localization & i18n

```
// JavaScript
const greeting = __("Hello, World!");
const message = n_("item", "items", count);  // Singular/Plural

<!-- Jinja Templates -->
<h1>{{ _("Welcome") }}</h1>
<p>{{ n_("One item", "Multiple items", items | length) }}</p>
```

**Note:** Enable translations in the doctype settings for dynamic content.

Security Checklist

| Check | Implementation |
|---|---|
| CSRF Protection | `frappe.call` handles automatically |
| XSS Prevention | Use `{{ user_input \| e }}` in Jinja |
| Input Validation | Validate in Python with `frappe.validates` |
| Guest Access | Use `@frappe.whitelist(allow_guest=True)` |
| Rate Limiting | Implement in API method |

```
# Python - Secure API Method
@frappe.whitelist()
def secure_method(data):
    # Validate input
    if not frappe.has_permission("DocType", "create"):
        frappe.throw(_("No permission"), frappe.PermissionError)

    # Sanitize data
    safe_data = frappe.sanitize_html(data)

    # Process
    return { "status": "success" }
```

Theming Best Practices

```css
/*  Use CSS Variables for theme support */
.my-component {
    background: var(--card-bg);
    color: var(--text-color);
    border: 1px solid var(--border-color);
    padding: var(--padding-md, 1rem);
}

/*  Dark mode ready */
@media (prefers-color-scheme: dark) {
    .my-component {
        background: var(--bg-dark);
        color: var(--text-dark);
    }
}
```

Performance Optimization

| Technique | Code |
|---|---|
| **Lazy Images** | `<img loading="lazy" src="...">` |
| **Lazy Content** | Use Intersection Observer API |
| **Server Caching** | `frappe.cache().set_value('key', data)` |
| **Query Optimization** | Use `frappe.get_all()` with fields/limit |

```python
# Python - Cached Data Fetch
import frappe

@frappe.whitelist()
def get_cached_data():
    cache_key = "my_app:cached_data"
    data = frappe.cache().get_value(cache_key)

    if not data:
        data = frappe.get_all("DocType",
            fields=["name", "title"],
            limit=10
        )
        frappe.cache().set_value(cache_key, data, expires_in_sec=3600)

    return data
```

Real-time Updates

```javascript
// Subscribe to real-time events
frappe.realtime.on('task_progress', function(data) {
```

```
    frappe.show_progress(__('Processing'), data.percent, 100);
    $('#progress-bar').width(data.percent + '%');
});

frappe.realtime.on('task_complete', function(data) {
    frappe.msgprint(__('Task completed!'));
    // Refresh data
    loadData();
});
```

Form Handling

```
// Create new document
frappe.db.insert({
    doctype: "Lead",
    first_name: "John",
    last_name: "Doe",
    email_id: "john@example.com",
    status: "Lead"
}).then(doc => {
    frappe.show_alert({
        message: __("Lead created: ") + doc.name,
        indicator: 'green'
    });
});

// Update existing
frappe.db.set_value("Lead", "DOC-001", "status", "Converted")
    .then(() => frappe.msgprint("Updated!"));

// Delete
frappe.db.delete("Lead", "DOC-001")
    .then(() => console.log("Deleted"));
```

Hooks Reference

| Hook | File | Purpose |
|---|---|---|
| `app_include_js` | hooks.py | Add JS to Desk + Website |
| `app_include_css` | hooks.py | Add CSS to Desk + Website |
| `web_include_js` | hooks.py | Add JS to Website only |
| `website_route_rules` | hooks.py | Custom URL routing |
| `fixtures` | hooks.py | Export/import configurations |

```
# hooks.py Example
app_include_js = "/assets/my_app/js/app.js"
web_include_js = "/assets/my_app/js/website.js"
```

```
app_include_css = "/assets/my_app/css/app.css"

fixtures = [
    {"dt": "Web Page Template", "filters": [["name", "like", "custom_%"]]}
]
```

Debugging Tips

```
// Debug: Log all requests
frappe.call = function(opts) {
    console.log("API Call:", opts.method, opts.args);
    return this._orig_call(opts);
};

// Debug: Inspect page context
frappe.ready(() => {
    console.log("Context:", frappe.boot);
    console.log("User:", frappe.session.user);
});
```

```
# Debug: Python
import frappe

# Pretty print to bench console
frappe.errprint(f"Debug: {my_variable}")

# Log to website
frappe.flags.in_test = True  # Shows debug in bench
```

---

## Developer

**Mohamed Ayman**

**Frappe Framework Developer**

**Available for Frappe/ERPNext consulting and custom development**

---

## License

**MIT License — Free to use and modify**

**Built with  for the Frappe Community**

---

**Star this guide if you found it helpful!**

Developed by Mohamed Ayman | Frappe Framework Expert

**Contact for projects:** Phone | WhatsApp | GitHub