MindMapper Data Schema

This document describes the data format used by MindMapper for storing mind maps.

File Format

Mind maps are stored as JSON files with the `.mindmap.json` extension. The format is designed to be:
- **Human-readable**: Easy to inspect and debug
- **Version-control friendly**: Works well with Git and other VCS
- **Extensible**: Easy to add new fields without breaking compatibility

---

Top-Level Structure

```
interface MindMap {
  id: string;              // Unique identifier for the mind map
  name: string;            // Display name of the mind map
  rootNodeId: string;      // ID of the root node
  nodes: {                 // Map of node IDs to node objects
    [nodeId: string]: MindMapNode;
  };
  createdAt: number;       // Unix timestamp (milliseconds)
  updatedAt: number;       // Unix timestamp (milliseconds)
}
```

Example

```
{
  "id": "m1a2b3c4d5",
  "name": "Project Planning",
  "rootNodeId": "n1a2b3c4d5",
  "nodes": {
    "n1a2b3c4d5": { /* root node */ },
    "n2b3c4d5e6": { /* child node */ }
  },
  "createdAt": 1697500000000,
  "updatedAt": 1697501234567
}
```

Node Structure

```typescript
interface MindMapNode {
  id: string;              // Unique identifier for the node
  text: string;            // Display text of the node
  parentId: string | null; // ID of parent node (null for root)
  children: string[];      // Array of child node IDs
  style: NodeStyle;        // Visual styling options
  collapsed: boolean;      // Whether children are collapsed
  order: number;           // Order among siblings (0-based)
}
```

Example

```json
{
  "id": "n1a2b3c4d5",
  "text": "Main Idea",
  "parentId": null,
  "children": ["n2b3c4d5e6", "n3c4d5e6f7"],
  "style": {
    "backgroundColor": "#8b5cf6",
    "textColor": "#ffffff",
    "borderColor": "#7c3aed",
    "borderWidth": 2,
    "borderRadius": 15,
    "fontSize": 16,
    "fontWeight": "bold",
    "padding": 15,
    "icon": "💡"
  },
  "collapsed": false,
  "order": 0
}
```

Style Structure

```typescript
interface NodeStyle {
  backgroundColor: string;  // CSS color (hex, rgb, rgba)
  textColor: string;        // CSS color
  borderColor: string;      // CSS color
  borderWidth: number;      // Border width in pixels (0-10)
  borderRadius: number;     // Corner radius in pixels (0-30)
  fontSize: number;         // Font size in pixels (10-24)
  fontWeight: 'normal' | 'bold'; // Font weight
  padding: number;          // Internal padding in pixels (5-30)
  icon?: string;            // Optional emoji icon
}
```

Default Style

```
{
  "backgroundColor": "#3b82f6",
  "textColor": "#ffffff",
  "borderColor": "#2563eb",
  "borderWidth": 2,
  "borderRadius": 8,
  "fontSize": 14,
  "fontWeight": "normal",
  "padding": 12
}
```

Viewport State (Not Persisted)

The viewport state is maintained in memory but not saved to files:

```
interface ViewportState {
  zoom: number;    // Zoom level (0.1 - 3.0)
  panX: number;    // Horizontal pan offset
  panY: number;    // Vertical pan offset
}
```

This allows users to open the same mind map at different zoom levels or positions.

Complete Example

Here’s a complete example of a simple mind map with two levels:

```
{
  "id": "map_abc123",
  "name": "My First Mind Map",
  "rootNodeId": "node_root",
  "nodes": {
    "node_root": {
      "id": "node_root",
      "text": "Central Topic",
      "parentId": null,
      "children": ["node_child1", "node_child2"],
      "style": {
        "backgroundColor": "#8b5cf6",
        "textColor": "#ffffff",
        "borderColor": "#7c3aed",
        "borderWidth": 3,
        "borderRadius": 20,
        "fontSize": 18,
        "fontWeight": "bold",
        "padding": 20,
        "icon": "🎯"
      },
      "collapsed": false,
      "order": 0
    },
    "node_child1": {
      "id": "node_child1",
      "text": "Subtopic 1",
      "parentId": "node_root",
      "children": ["node_grandchild1"],
      "style": {
        "backgroundColor": "#3b82f6",
        "textColor": "#ffffff",
        "borderColor": "#2563eb",
        "borderWidth": 2,
        "borderRadius": 15,
        "fontSize": 16,
        "fontWeight": "normal",
        "padding": 15,
        "icon": "📌"
      },
      "collapsed": false,
      "order": 0
    },
    "node_child2": {
      "id": "node_child2",
      "text": "Subtopic 2",
      "parentId": "node_root",
      "children": [],
      "style": {
        "backgroundColor": "#10b981",
        "textColor": "#ffffff",
        "borderColor": "#059669",
        "borderWidth": 2,
        "borderRadius": 15,
        "fontSize": 16,
        "fontWeight": "normal",
        "padding": 15,
        "icon": "✨"
      },
      "collapsed": false,
      "order": 1
    },
```

```
    "node_grandchild1": {
      "id": "node_grandchild1",
      "text": "Detail 1.1",
      "parentId": "node_child1",
      "children": [],
      "style": {
        "backgroundColor": "#dbeafe",
        "textColor": "#1e40af",
        "borderColor": "#60a5fa",
        "borderWidth": 1,
        "borderRadius": 10,
        "fontSize": 14,
        "fontWeight": "normal",
        "padding": 12
      },
      "collapsed": false,
      "order": 0
    }
  },
  "createdAt": 1697500000000,
  "updatedAt": 1697501234567
}
```

Data Validation

When loading a mind map file, MindMapper validates:

1. **Required Fields**: All required fields are present
2. **Node References**: All parent/child references are valid
3. **Root Node**: Root node exists and has no parent
4. **Circular References**: No circular parent-child relationships
5. **Style Values**: Style values are within valid ranges

Invalid files will show an error message and fail to load.

Data Migration

Future Compatibility

The data format is designed to be backward-compatible. Future versions may:
- Add new optional fields (safely ignored by older versions)
- Add new node types or categories
- Extend the style object with new properties

Version Field (Future)

Future versions may include a version field:

```
{
  "version": "1.0.0",
  "id": "...",
  ...
}
```

Import Formats

Markdown

MindMapper can import Markdown files with the following rules:

- `#` creates a level 1 node (child of root)
- `##` creates a level 2 node (grandchild)
- `###` and beyond create deeper levels
- Non-header lines are treated as children of the last header
- Bullet points (`-`, `*`, `+`) become child nodes

Example:

```
# Planning
## Goals
- Goal 1
- Goal 2
## Timeline
```

Becomes:

```
Root
└── Planning
    ├── Goals
    │   ├── Goal 1
    │   └── Goal 2
    └── Timeline
```

JSON

Standard JSON files are imported directly if they match the MindMap schema.

Export Formats

JSON

Exports the complete mind map structure, identical to the save format.

PDF

Creates a vectorial PDF of the visual representation. Does not preserve the data structure—it's a snapshot of the visual layout.

Node ID Generation

Node IDs are generated using:

```
const generateId = () => Math.random().toString(36).substr(2, 9);
```

This creates IDs like: `"k3j4h5g6f"`

Important: IDs must be unique within a mind map. The application ensures this, but manually editing files requires care.

Best Practices

File Naming

- Use descriptive names: `project-plan.mindmap.json`
- Include dates for versions: `project-plan-2024-10-15.mindmap.json`
- Keep extensions: Always use `.mindmap.json`

Version Control

Mind map files work well with Git:
- Enable pretty-printing (already done by default)
- Use meaningful commit messages
- Consider splitting large maps into multiple files

Backup Strategy

- Save frequently (Ctrl+S)
- Keep multiple versions
- Use cloud storage for automatic backups
- Export to JSON for archival copies

Manual Editing

You can manually edit mind map files in a text editor:
- Ensure valid JSON syntax
- Maintain ID uniqueness
- Keep parent-child relationships consistent
- Validate references before saving

Limitations

Current limitations of the format:

1. **Binary Data**: No support for embedded images or files (yet)
2. **Metadata**: Limited metadata fields (no tags, categories, etc.)
3. **Relationships**: Only hierarchical relationships (no cross-links)
4. **Attachments**: No file attachments or hyperlinks

These may be addressed in future versions.

Future Enhancements

Planned additions to the data schema:

Phase 2

- `metadata` field for tags, categories, and custom fields
- `attachments` array for file references
- `links` array for cross-node relationships
- `notes` field for additional text content

Phase 3

- `version` field for format versioning
- `collaborators` array for multi-user support
- `history` array for change tracking
- `theme` field for custom color schemes

Related Documentation

- USAGE.md (./USAGE.md) - User guide for the application
- ARCHITECTURE.md (./ARCHITECTURE.md) - Technical architecture details

---

For questions or suggestions about the data format, please open an issue on GitHub.