Paolo Dalprato

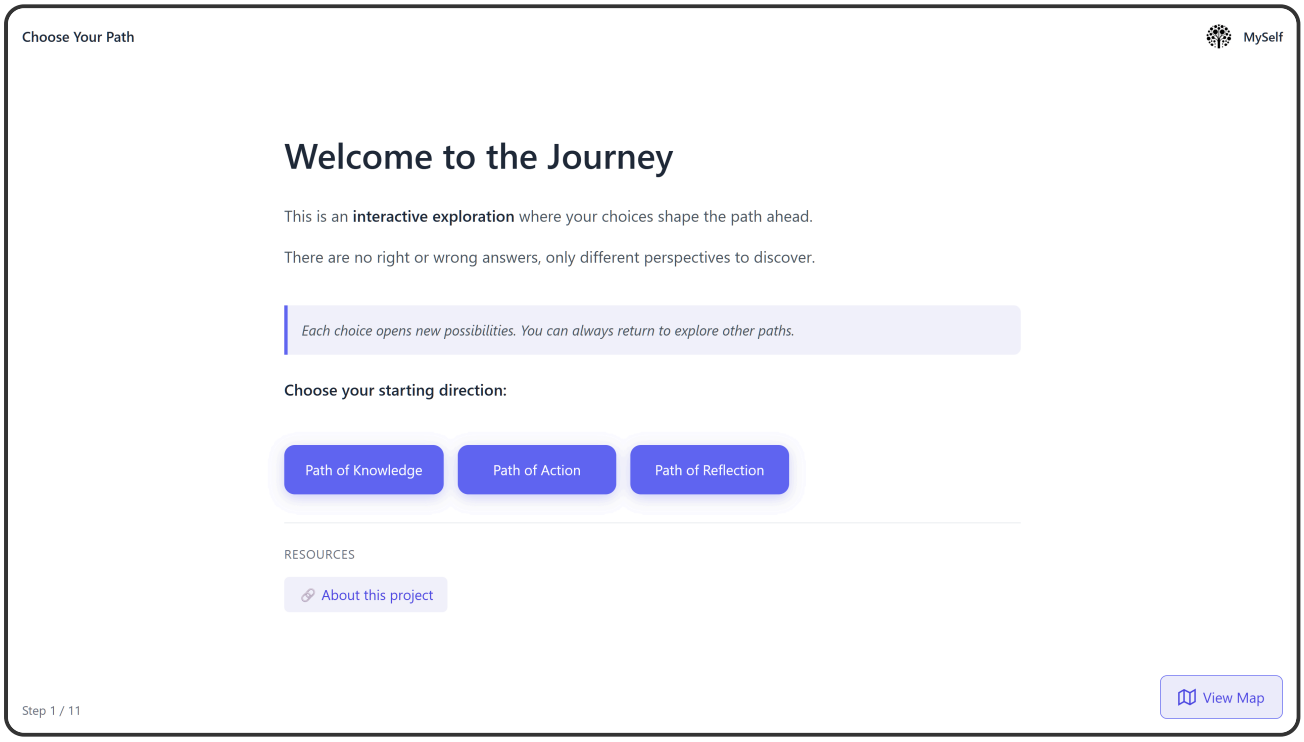
ChoiceMap

Last updated: January 26, 2026

A framework for documenting multiple-choice processes.

ChoiceMap transforms complex workflows into interactive experiences. Instead of reading a document from start to finish, users navigate through a decision tree where each choice opens a different path.

Users see a series of cards with multiple choices leading to different paths. At any time, they can open the visual map to see the complete structure, visited nodes, and current position. This dual mode—guided navigation and overview—reduces the disorientation typical of branching content.



(https://docs.ai-know.pro/choicemap-en/img/navigator-interface.png)

The tool is designed for corporate trainers, instructional designers, and educational content creators who want to transform static materials into engaging experiences. A procedures manual becomes a guided path, troubleshooting becomes an interactive guide, a case study becomes a decision-making simulation.

On GitHub, the repository is github.com/paolodalprato/ChoiceMap (https://github.com/paolodalprato/ChoiceMap)

What you can create

With ChoiceMap you can build different types of interactive content.

Training paths that adapt to user responses: those who answer correctly proceed, those who make mistakes receive additional explanations before continuing.

Process simulations that guide through complex procedures, such as support ticket escalation or customer request handling.

Branching quizzes where each answer leads to specific feedback and the path changes based on choices.

Interactive stories in "choose your own adventure" style for onboarding, safety training, or exploring business scenarios.

How it works

The framework consists of three tools that work together:

- **Navigator**: displays the scenario and manages user interaction.
- **Scenario Editor**: allows creating and modifying content visually.
- **Theme Editor**: enables customizing colors, fonts, and company branding.

→ Try the Navigator (https://paolodalprato.github.io/ChoiceMap/choicemap.html)

The content is stored in simple JSON files, editable even with a text editor. No complex server is needed: a static web server is sufficient to publish scenarios, such as GitHub Pages.

Who this manual is for

The manual guides step by step from first installation to online publication. It requires no programming skills. Basic familiarity with using a browser and performing simple operations like downloading files and editing text is sufficient.

If you already know what a JSON file is, you'll find the data structure familiar. If you don't, the manual explains everything needed to create working scenarios.

Table of contents

- Responsiveness

Scenario Editor interface

- Layout
- Editing area blocks
- Settings: metadata and translations
- The JSON file
- Saving

Theme Editor interface

- General layout
- Left column
- Live preview
- Typical workflow

Creating a scenario

- Plan before building
- Recommended workflow
- Managing terminal nodes
- Managing orphan nodes
- Deleting nodes safely
- Creating loops
- Adding resources
- Modifying an existing scenario
- From creation to publication

Content structure

- File architecture
- The scenario file
- Validation

Customizing the theme

- Recommended workflow
- Branding tips
- Typography tips
- Color tips
- Saving and applying
- Managing multiple themes
- Modifying the theme via JSON

Publication

- Roles and responsibilities
- Administrator workflow
- Preparing for publication
- Publication destinations
- GitHub Pages
- Managing multiple scenarios
- Embedding in other sites
- Security and access

Quick reference

- Start local server
- Create a scenario
- Modify an existing scenario
- Customize the theme
- Configure config.json
- Test locally
- Publish
- Map color codes (Scenario Editor)
- Markdown in content
- Quick troubleshooting

Overview

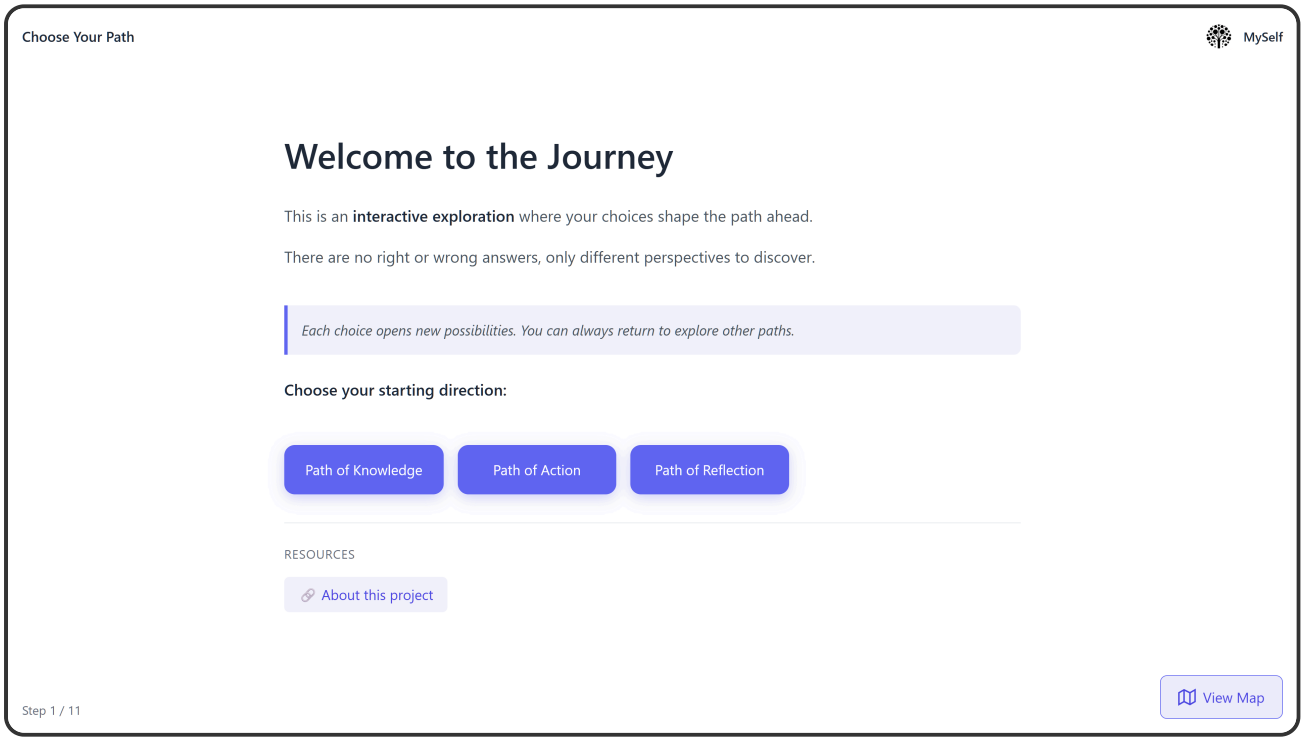
ChoiceMap transforms linear content into interactive experiences. Instead of reading a document from start to finish, users navigate through a tree of choices where each decision opens a different path.

The three components

The framework consists of three distinct tools, each with a specific role.

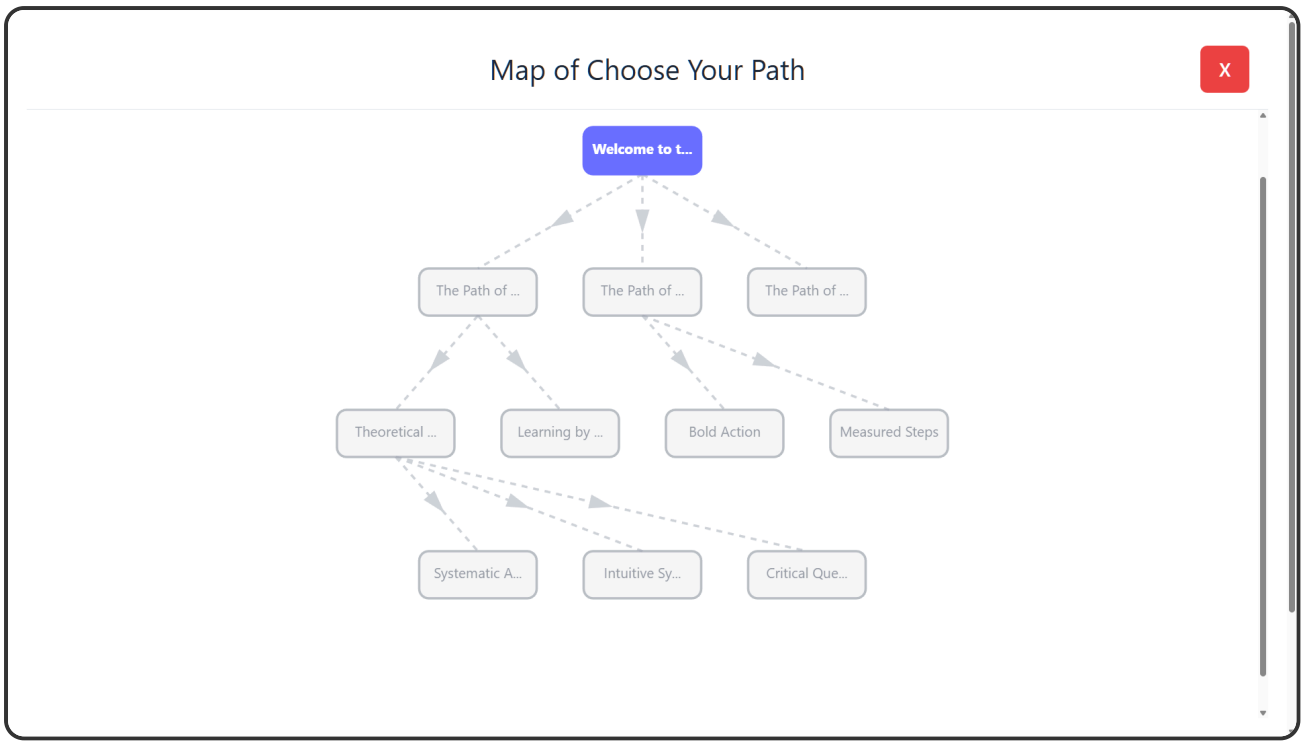
Navigator

This is the visualization engine, what the end user sees and uses. It loads a JSON scenario and presents it as a sequence of screens. Each screen shows text content and a series of buttons to choose from, with the ability to attach files and share links. The choice determines which screen appears next.



(https://docs.ai-know.pro/choicemap-en/img/navigator-interface.png)

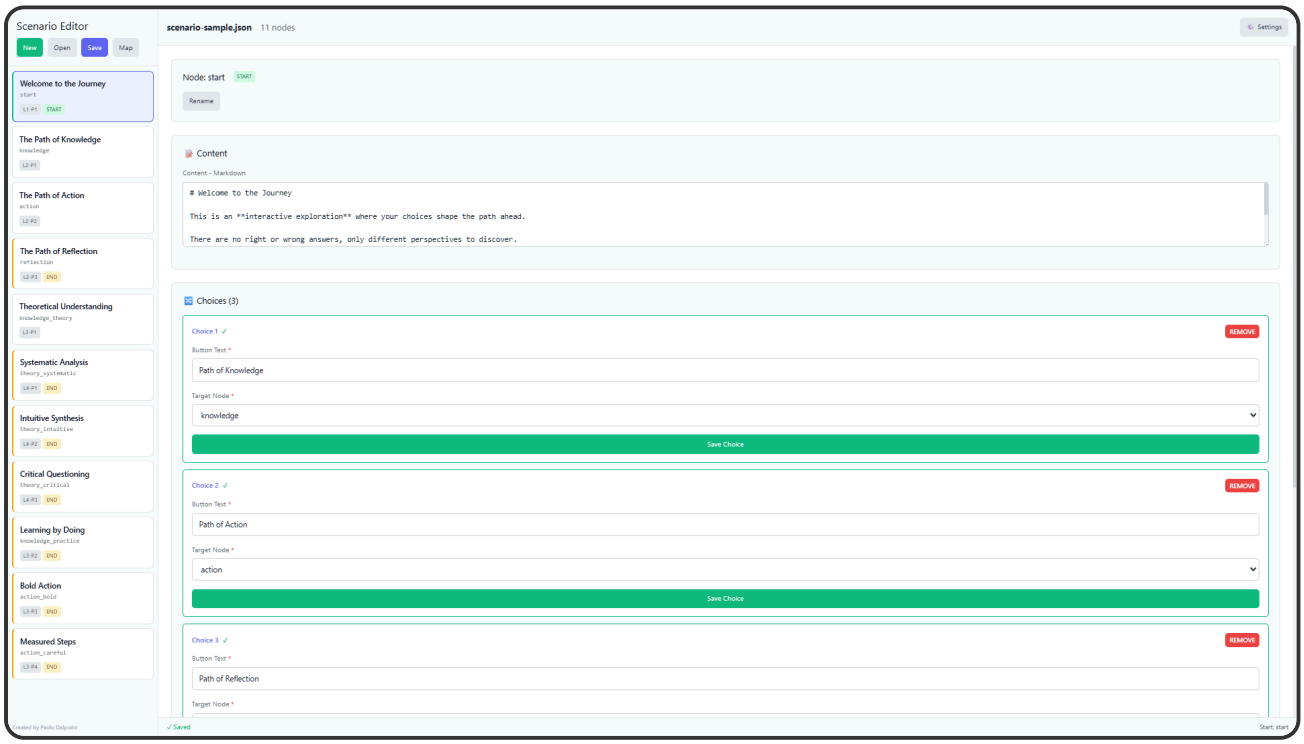
The Navigator includes a visual journey map that shows where the user is in the decision tree, which nodes have been visited, and which remain to explore. Users can go back, start over, or jump to a specific point in the already visited part of the map.



(https://docs.ai-know.pro/choicemap-en/img/navigator-map.png)

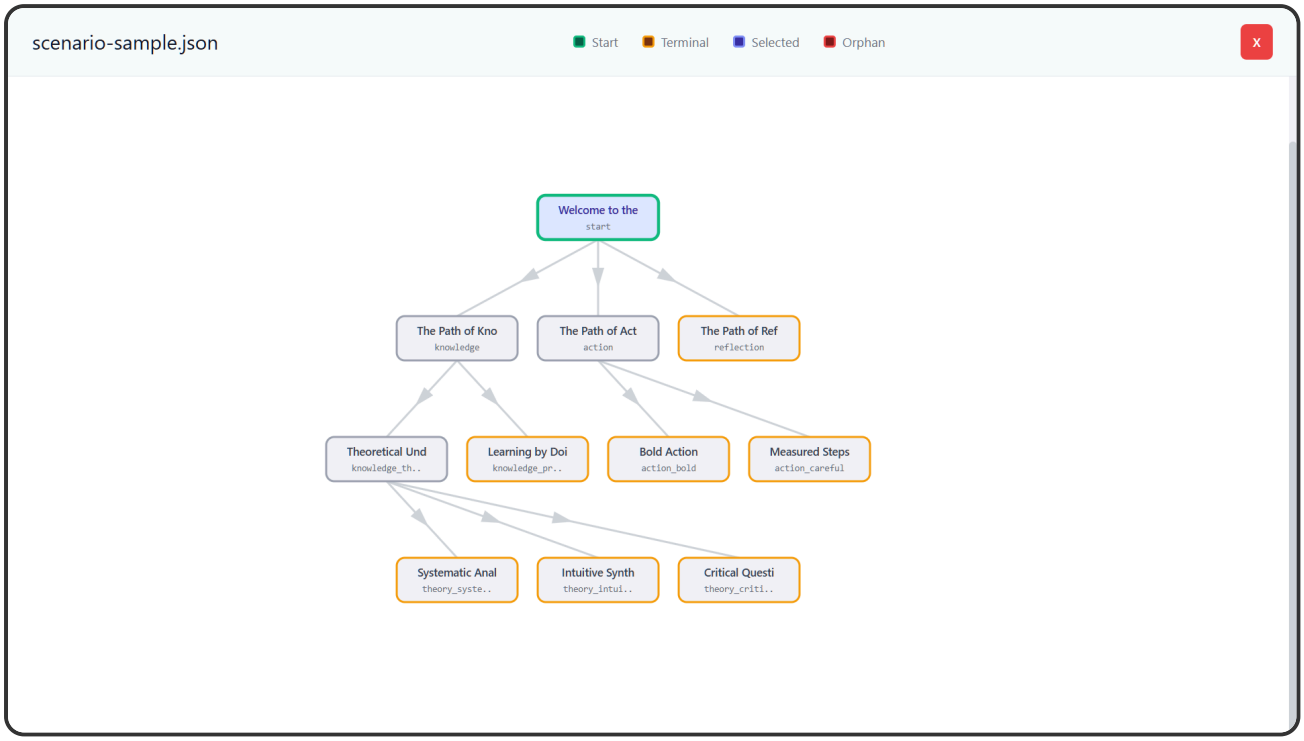
Scenario Editor

This is the tool for creating and modifying scenarios without writing JSON manually. When opening an existing scenario for editing, the editor presents a list of all nodes, allows modifying their content, adding choices, and connecting nodes to each other.



(https://docs.ai-know.pro/choicemap-en/img/scenario-edit.png)

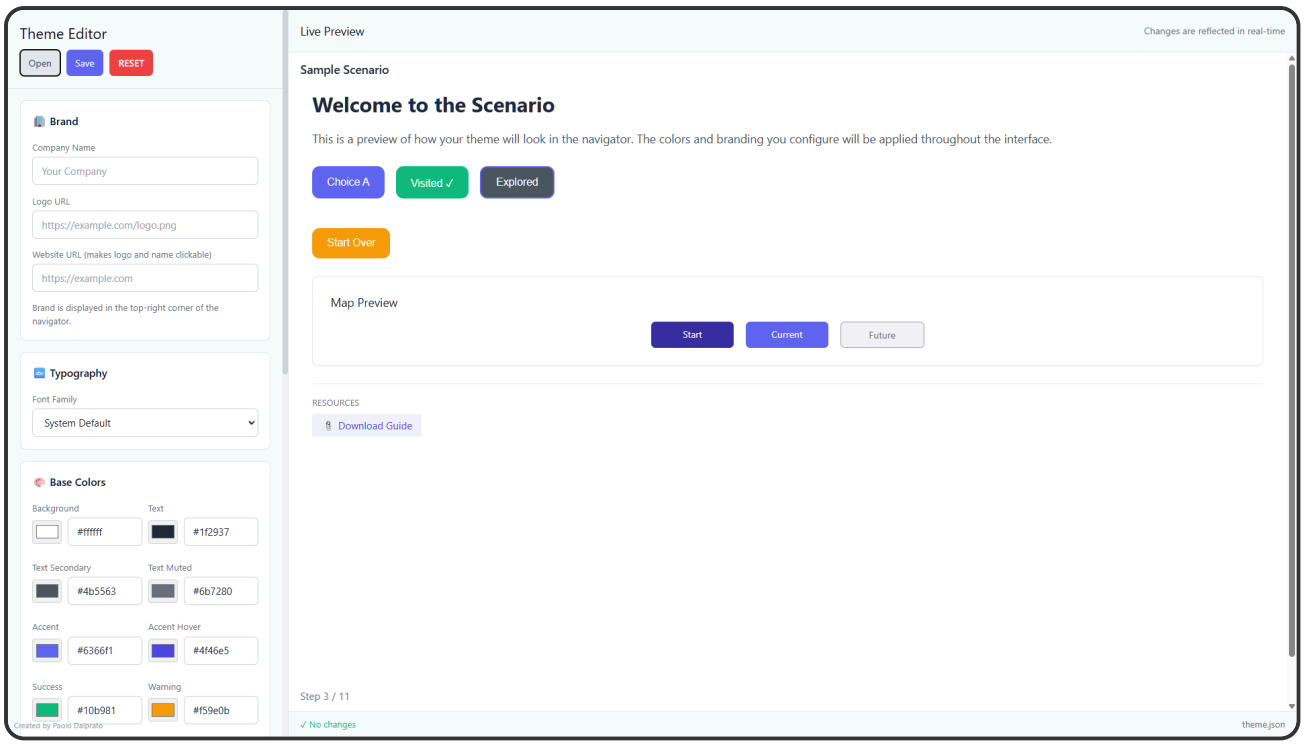
The editor includes a map view showing the scenario's tree structure, highlighting the start node, end nodes, and any "orphan" nodes (disconnected from the main flow). This view helps identify structural problems before publishing.



(https://docs.ai-know.pro/choicemap-en/img/scenario-map.png)

Theme Editor

This is the tool for customizing the Navigator's visual appearance. It allows modifying colors, fonts, adding logo and company name, without touching the code.



(https://docs.ai-know.pro/choicemap-en/img/theme-editor.png)

Changes are visible in real-time in a preview. Once satisfied, save the theme as a JSON file and associate it with the scenario.

Typical use cases

ChoiceMap adapts to different training and communication contexts.

Corporate training

An onboarding course can become an interactive path where new employees explore company procedures by making choices and receiving contextual feedback. Instead of reading a 50-page manual, they navigate through scenarios that simulate real situations, connecting each step to specific chapters in related documents.

Compliance training is particularly well-suited: scenarios that present ambiguous situations, ask for a decision, and show the consequences of correct or incorrect choices.

Quizzes and assessments

A traditional quiz presents questions in sequence. With ChoiceMap, the path changes based on answers: those who answer correctly proceed, those who make mistakes receive an explanation and can retry or are directed to a deeper learning path.

This approach transforms assessment into a learning experience: the error is not just a negative score but an opportunity to understand better.

Process documentation

A workflow like support ticket management can be represented as a decision tree. The operator answers questions about the situation ("Is it a technical or billing issue?", "Is the service completely down?") and is guided toward the correct procedure.

This approach allows representing complex situations, reducing errors and accelerating new staff training.

Interactive storytelling

Stories in "choose your own adventure" style where the reader makes choices between multiple paths. Useful for experiential training, negotiation simulations, exploration of ethical scenarios.

Technical architecture

ChoiceMap is built with standard web technologies: HTML, CSS, JavaScript. It uses React for the interface and Tailwind CSS for styling, both loaded from CDN. It requires no compilation or dependency installation.

Each scenario is a JSON file with a simple structure: metadata (title, author), interface label translations, and a collection of nodes. Each node has text content in Markdown and a list of choices pointing to other nodes.

Themes are also JSON files that define colors, fonts, and branding information.

The Navigator works on any modern browser and automatically adapts to desktop and mobile devices.

Installation and startup

This section details the requirements and procedures to run ChoiceMap on your computer.

Requirements

ChoiceMap has minimal requirements. You need a modern browser (Chrome, Firefox, Safari, Edge in recent versions) and Python to start the local server.

Python is only needed for local development. Once published on a web server, end users don't need Python: they access directly from the browser.

On Windows, Python might not be preinstalled. You can download it from python.org or install it through the Microsoft Store. To check if it's present, open the Command Prompt and type `python --version`. If a version number (3.x) appears, Python is installed.

On Mac, Python is typically already present. On Linux, it's almost always available by default.

Download the package

The complete package is available as a ZIP file from the project's GitHub page, in the Releases section. The download includes:

- `choicemap.html` - the Navigator
- `scenario-editor.html` - the editor for creating scenarios
- `theme-editor.html` - the editor for customizing themes
- `config.json` - configuration file
- `defaults.json` - shared default values
- `theme.json` - default theme
- `scenario-sample.json` - narrative sample scenario
- `scenario-quiz.json` - quiz example
- `scenario-workflow.json` - workflow example
- Batch scripts for Windows
- `docs/` folder for downloadable resources

Extract the ZIP to a folder of your choice. The path should not contain special characters or spaces to avoid problems with some systems.

Starting on Windows

The simplest way on Windows is to use the included batch scripts.

Double-click `start-navigator.bat` to start the server and open the Navigator in the default browser. The script starts a server on port 8000.

Double-click `start-scenario-editor.bat` to directly open the Scenario Editor.

Double-click `start-theme-editor.bat` to open the Theme Editor.

If an error appears indicating that Python is not found, you need to install it or add it to the system PATH.

The scripts create a server that stays active as long as the terminal window is open. Closing the window stops the server.

Manual startup

If you prefer not to use the scripts or you're on Mac/Linux, manual startup requires a few commands.

Open a terminal in the project folder. On Windows, you can right-click while holding Shift and select "Open PowerShell window here" or "Open in terminal".

Run the command:

```
python -m http.server 8000
```

The terminal shows a message like "Serving HTTP on 0.0.0.0 port 8000". The server is active.

Open your browser and navigate to:

- Navigator: `http://localhost:8000/choicemap.html`
- Scenario Editor: `http://localhost:8000/scenario-editor.html`
- Theme Editor: `http://localhost:8000/theme-editor.html`

To stop the server, return to the terminal and press Ctrl+C.

Why a server is needed

Opening HTML files directly (double-click or dragging to browser), the Navigator doesn't work. A blank screen or error appears.

The reason is a browser security restriction. When an HTML file is opened from the local filesystem (URL starting with `file://`), the browser prevents loading other files like scenario JSONs. It's a protection

against malicious scripts.

The local server bypasses this limitation by serving files via HTTP (URL starting with `http://`). It's the same protocol used by websites, so the browser allows loading.

This limitation only exists locally. Once published on a web server (GitHub Pages, corporate hosting), everything works normally.

Ports and conflicts

The server uses port 8000 by default. If that port is already occupied by another program, an error appears.

To use a different port:

```
python -m http.server 8080
```

Remember to update the address in the browser: `http://localhost:8080/choicemap.html`.

Folder structure

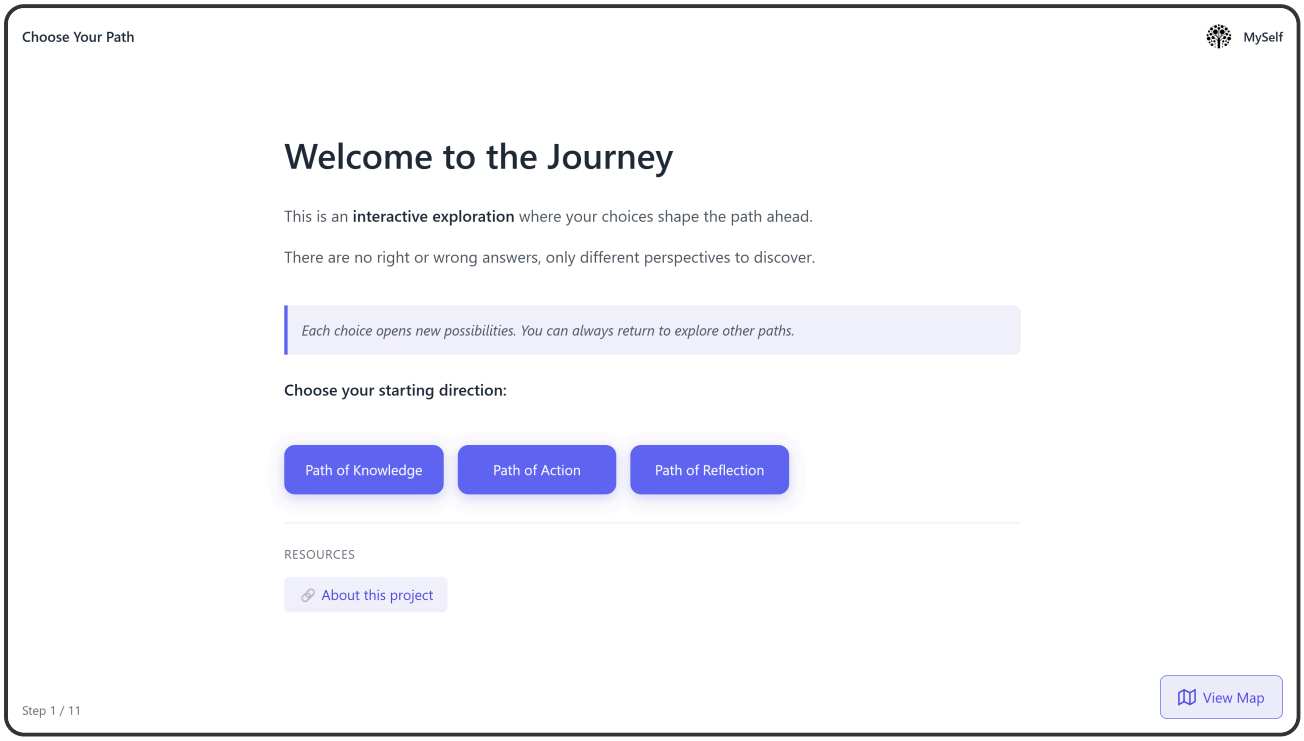
It's not necessary to keep all files in the same folder, but it simplifies management. The recommended structure is:

```
choicemap/
├── choicemap.html
├── scenario-editor.html
├── theme-editor.html
├── config.json
├── defaults.json
├── theme.json
├── scenario-my-project.json
├── theme-company.json
└── docs/
    └── (PDF files, resources)
```

If you prefer to organize differently, for example putting JSON files in a subfolder, you need to update the paths in config.json.

Navigator interface

The Navigator is what the end user sees when interacting with a scenario. The interface is designed to be clean and focused on content, without distractions.



(https://docs.ai-know.pro/choicemap-en/img/navigator-interface.png)

Header area

At the top of the screen is the header with context information.

On the left appears the scenario title, the one defined in the JSON file metadata. It helps identify which path you're in, useful when an organization has multiple scenarios to manage.

On the right, if configured, the company branding appears: logo, company name, optionally a link to the website.

Progress indicator

At the bottom left is the progress indicator, a line showing "Step X" where X is the number of screens traversed from the start. This gives the user a sense of advancement in the path.

The "Step" label is customizable in the scenario translations. For a quiz it could be "Question", for a workflow it could be "Phase".

Map

At the bottom right is the button to activate the map visualization, covered in depth later in this chapter.

Content area

The central section is dedicated to the current node's content. Text is rendered as Markdown, so it supports headings, bold, italic, bullet lists, and blockquotes.

The content can start with a large heading (H1) that serves as the screen header, followed by text paragraphs, lists, highlighted quotes. Formatting helps structure complex information in a readable way.

Below the text content, additional resources may appear: links to documents, files to download, videos to watch. Each resource has an icon indicating its type and a button to access it.

Choice buttons

At the bottom of the content area, choice buttons appear. Each button represents an option the user can select, with descriptive text that anticipates where that choice will lead.

Buttons change color based on state. The primary color indicates unexplored choices. A different color indicates choices already made previously, useful when the user goes back to explore alternative paths. A third color can indicate choices explored but not in the current path.

When a node has no choices, it means it's an endpoint of the path. Instead of choice buttons, a message like "**End of path**" appears (this text is also editable in the scenario editor) and a button to restart.

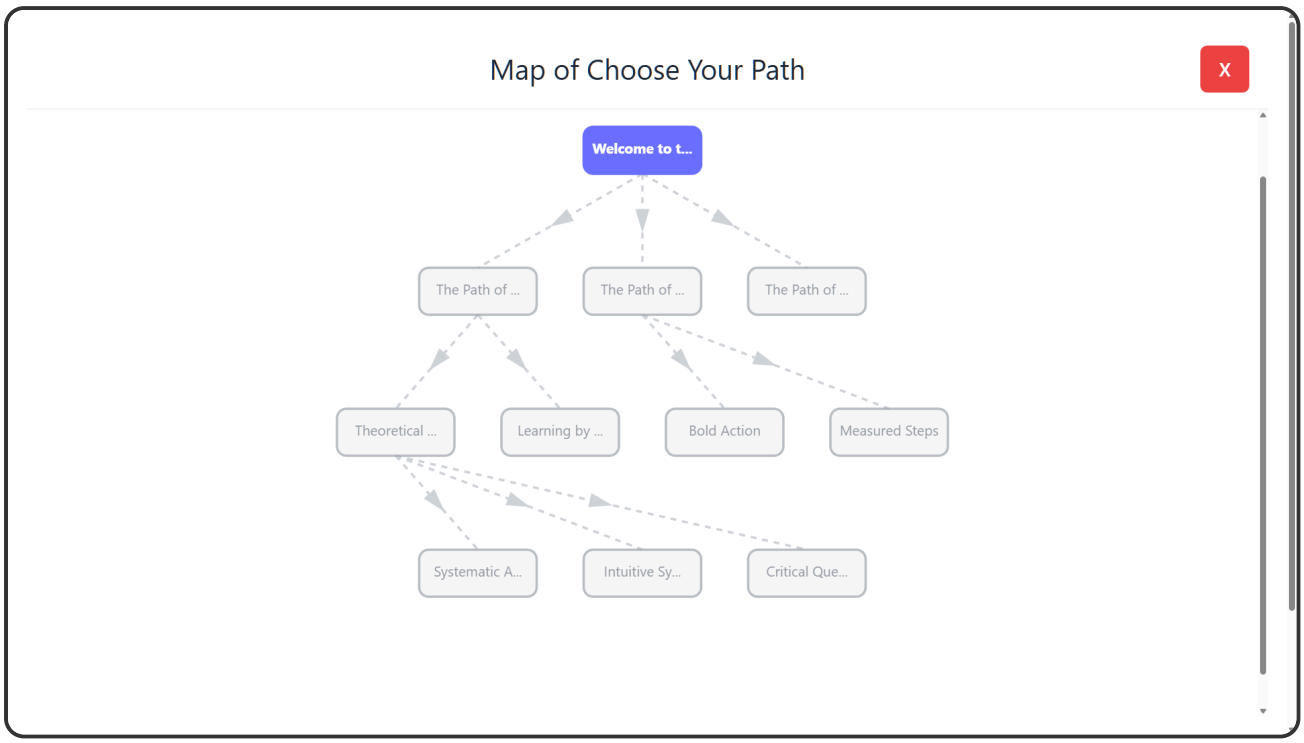
Navigation

Once you've started a path, a "**Back**" button appears at the top, allowing you to return to the previous card, retracing the path taken. It's useful for rereading information or reconsidering a choice.

The "**Restart**" button returns to the beginning of the scenario, resetting the path. Useful for exploring again with different choices.

The "**View Map**" button opens the tree visualization of the path, which can also be used to move through the structure, see the next section.

Journey map



(https://docs.ai-know.pro/choicemap-en/img/navigator-map.png)

The map shows the entire scenario as a tree of connected nodes. It's a visualization that allows seeing where you are relative to the overall structure.

Each node is represented by a rectangle with an abbreviated title. Nodes are colored differently based on their state.

The current node, where the user is located, is highlighted with the primary color. Nodes already visited in the current path have a color indicating "already seen". Nodes not yet reached are gray or with a dashed border.

The lines connecting nodes represent choices. Lines also change color: those traversed are solid and highlighted, those not yet explored are dashed. Arrows on the lines indicate from which node to which node the user moves, as connections are always unidirectional.

Clicking on a node in the map jumps directly to that card, if already visited.

The map closes with the close button at the top right.

Responsiveness

The interface automatically adapts to screen dimensions. On mobile devices, buttons become larger to facilitate touch, the map resizes to remain readable, text maintains proportions suitable for reading.

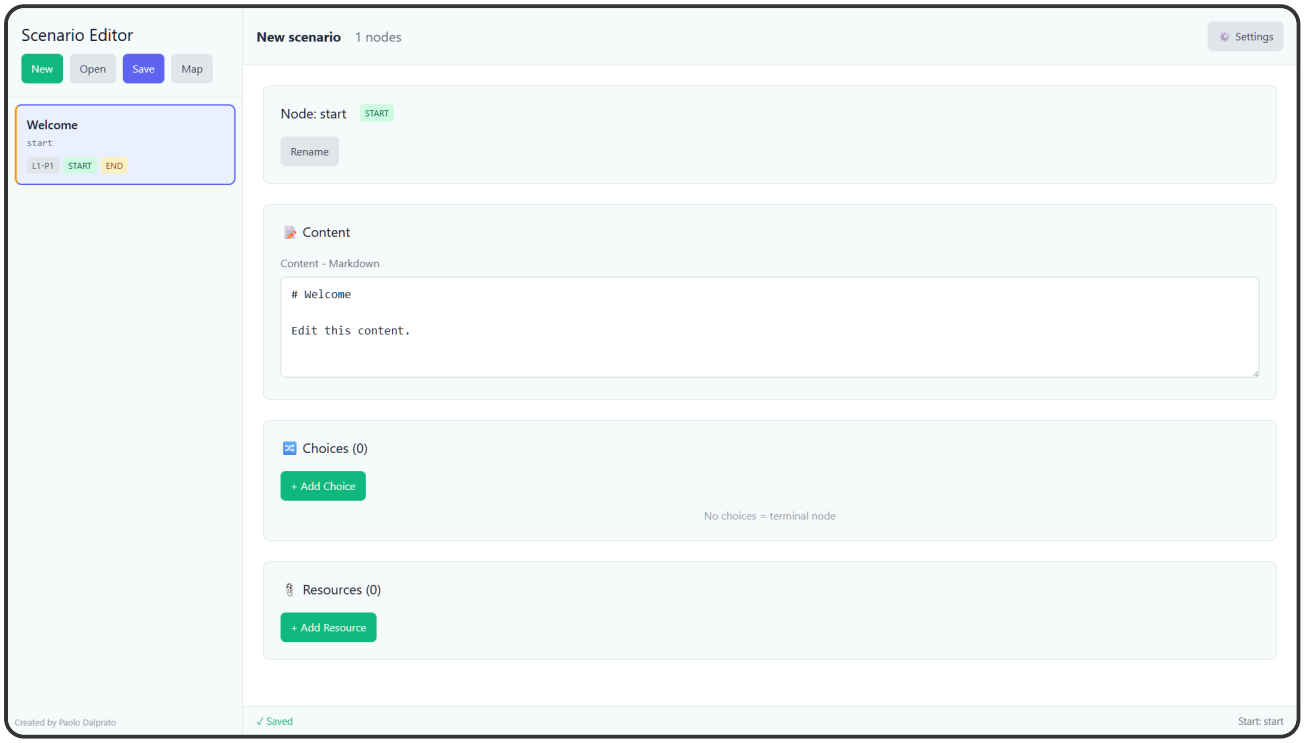
Navigation remains functional even on small screens: all controls remain accessible and content scrolls vertically when necessary.

Scenario Editor interface

The Scenario Editor is the tool for creating and modifying scenarios.

Layout

A scenario is the set of nodes (which the user sees as cards in the Navigator) and connections between nodes (which the user sees as choice buttons within the cards). Scenarios are saved as JSON files.

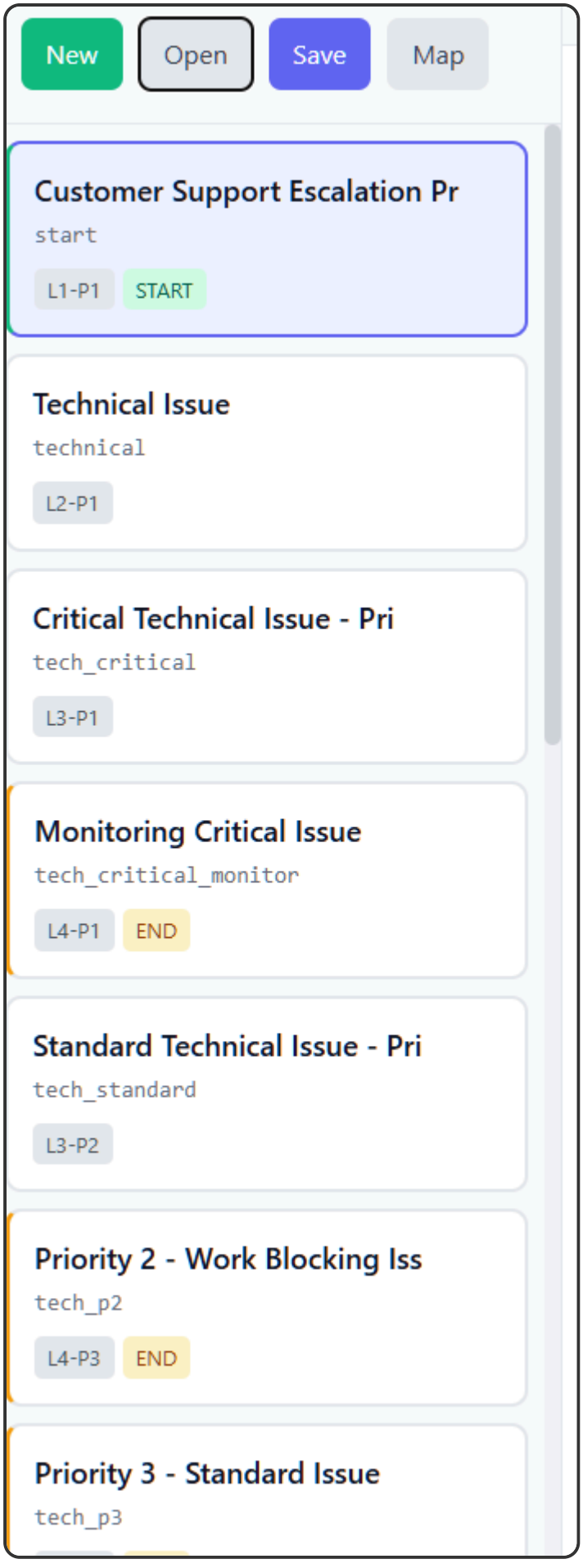


(https://docs.ai-know.pro/choicemap-en/img/scenario-edit-welcome.png)

The interface is divided into two parts.

Left column

The left column is dedicated to managing general operations and the open scenario.



At the top it shows a row with buttons for:

- **New**: create a new scenario
- **Open**: open an existing scenario
- **Save**: save the current scenario as a JSON file
- **Map**: activate the map visualization

Button names are in English by default, but can be customized through Settings.

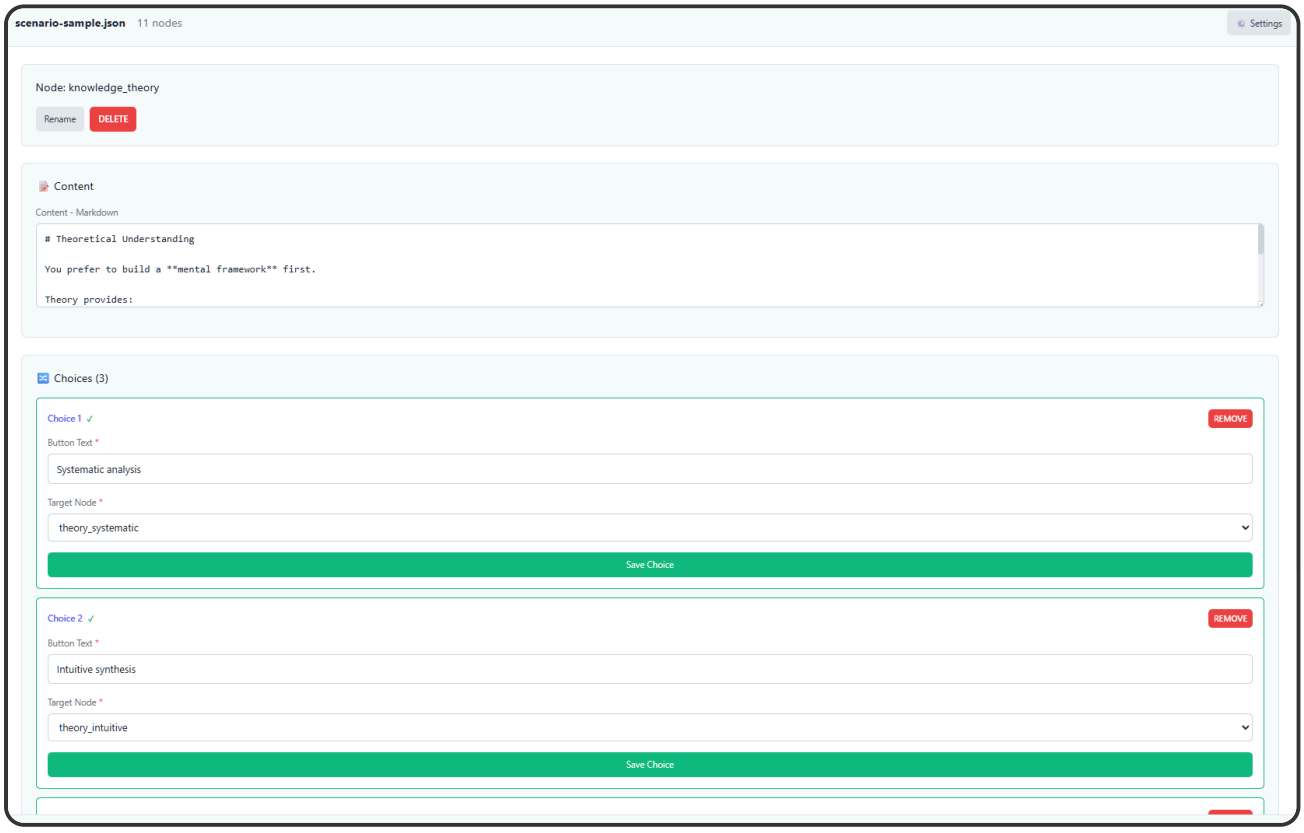
Below the button row appears the list of nodes that are part of the open scenario: only the Start node if starting with a new scenario, a complete list if you're already advanced in creation or if you open an existing scenario.

Each node in the list shows:

- the title (extracted from the content)
- the technical identifier
- informational badges: level and position (e.g., L2-P1), START for the initial node, END for terminal nodes

Editing area

The right part is dedicated to editing the selected node.



(https://docs.ai-know.pro/choicemap-en/img/scenario-edit-area.png)

The top row contains:

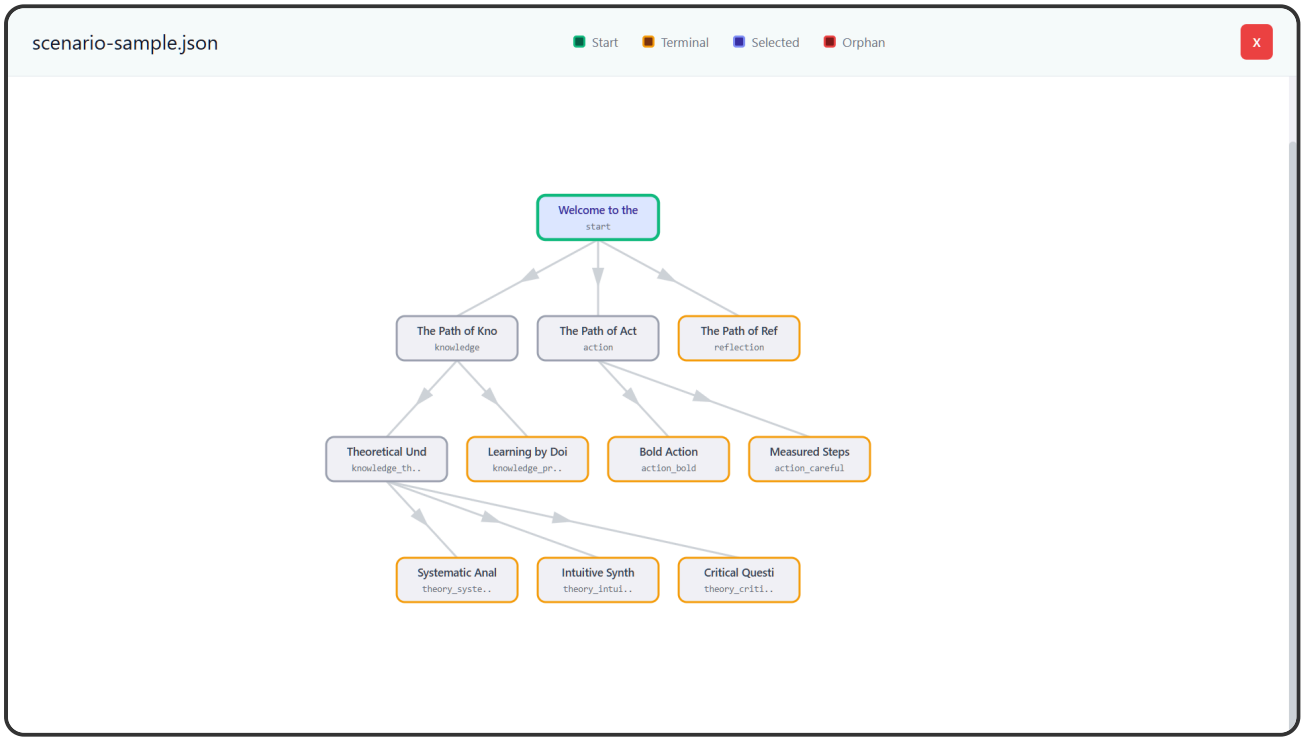
- on the left: the scenario file name and total nodes. For a new scenario "New scenario" appears, the name changes automatically after saving
- on the right: the **Settings** button for metadata and translations

Below the header row appear the blocks that define the node:

- **Node**: node identifier management
- **Content**: the text content
- **Choices**: the choices available to the user
- **Resources**: attachments and linked resources

Map view

The map view shows the scenario as a visual tree.



(https://docs.ai-know.pro/choicemap-en/img/scenario-map.png)

Each node shows its identifier and title. Connections between nodes represent available choices, arrows indicate the path direction. The automatic layout arranges nodes in levels, with the start node at the top and branches expanding downward.

Border color codes:

- **green**: start node (START)
- **orange**: terminal node (END)

- **red**: orphan node, unreachable
- **purple**: selected node

Backward connections (loops) are colored orange to distinguish them from forward connections.

Clicking on a node in the map opens it directly in the editing area. To close the map, click the X button at the top right.

Editing area blocks

Node

The Node block manages the node identifier. It contains:

- **Rename**: opens a field to modify the identifier. The ID must be unique in the entire scenario and can only contain letters, numbers, and underscores
- **DELETE**: deletes the entire node from the scenario. Not available for the START node. Before deleting, the system asks for confirmation

> ⚠ **Deletion warning**
>
> Deleting a node that has connected children makes those children orphans. It's good practice to delete nodes starting from leaves toward the root.

Content

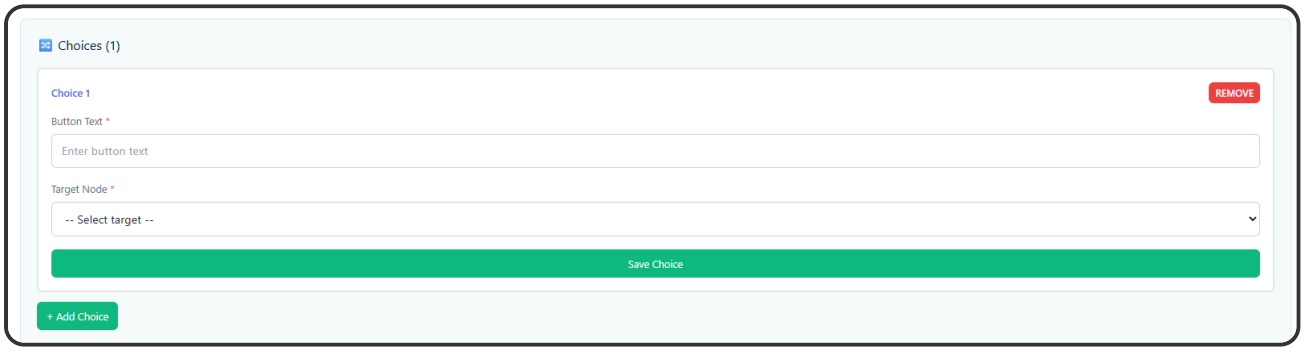
The Content block contains the text that the user will see in the Navigator card.

The content uses a simplified Markdown syntax:

Element	Syntax
Large heading	`# Text`
Heading	`## Text`
Bold	`**text**`
Italic	`*text*`
List item	`- text`
Quote	`> text`

Choices

The Choices block defines the choices available to the user, i.e., the buttons that allow proceeding in the path.

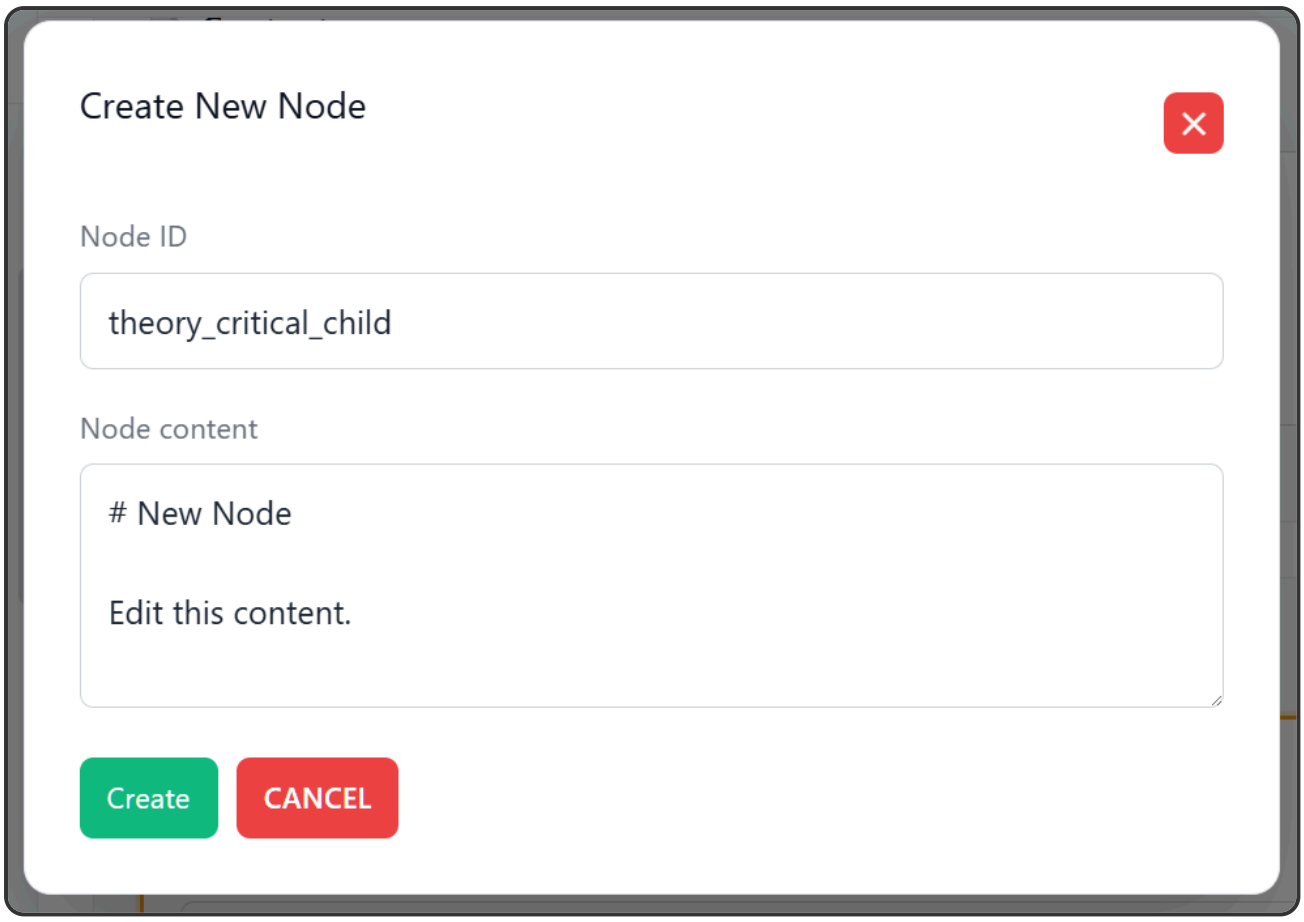


(https://docs.ai-know.pro/choicemap-en/img/scenario-add-choice.png)

Clicking **+ Add Choice** adds a new choice with two required fields:

- **Button Text**: the text that appears on the button
- **Target Node**: the destination node, selectable from a dropdown menu listing all existing nodes

The Target Node menu also includes the option **+ Create new node** which opens a popup to directly create a new destination node.



(https://docs.ai-know.pro/choicemap-en/img/scenario-new-node.png)

Visual status indicators:

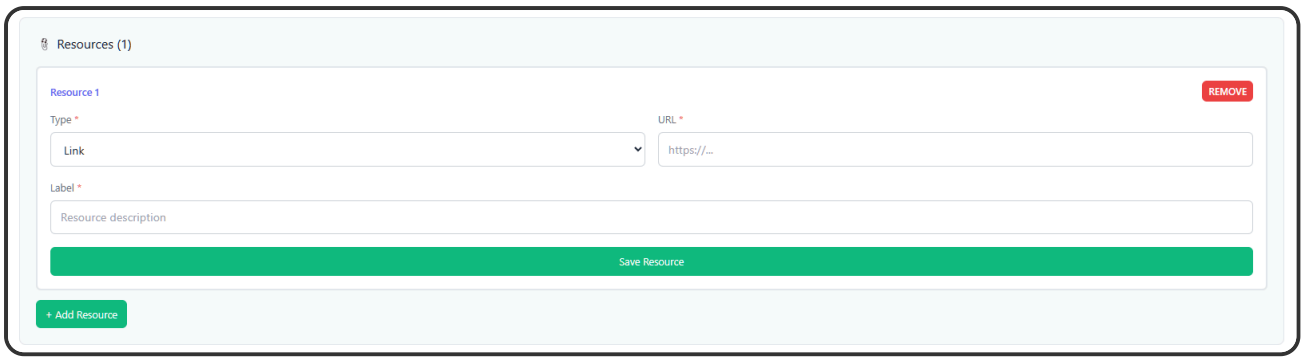
- gray border: newly created choice, not yet filled in
- amber border with "● unsaved": unsaved changes
- green border with "✓": complete and saved choice
- red border: validation errors

The **Save Choice** button saves the choice only if both fields are filled in. The **REMOVE** button deletes the choice.

A node can have multiple choices leading to different nodes. It's possible to create loops by connecting a node to one of its ancestors.

Resources

The Resources block allows attaching external resources to the node.



(https://docs.ai-know.pro/choicemap-en/img/scenario-new-resource.png)

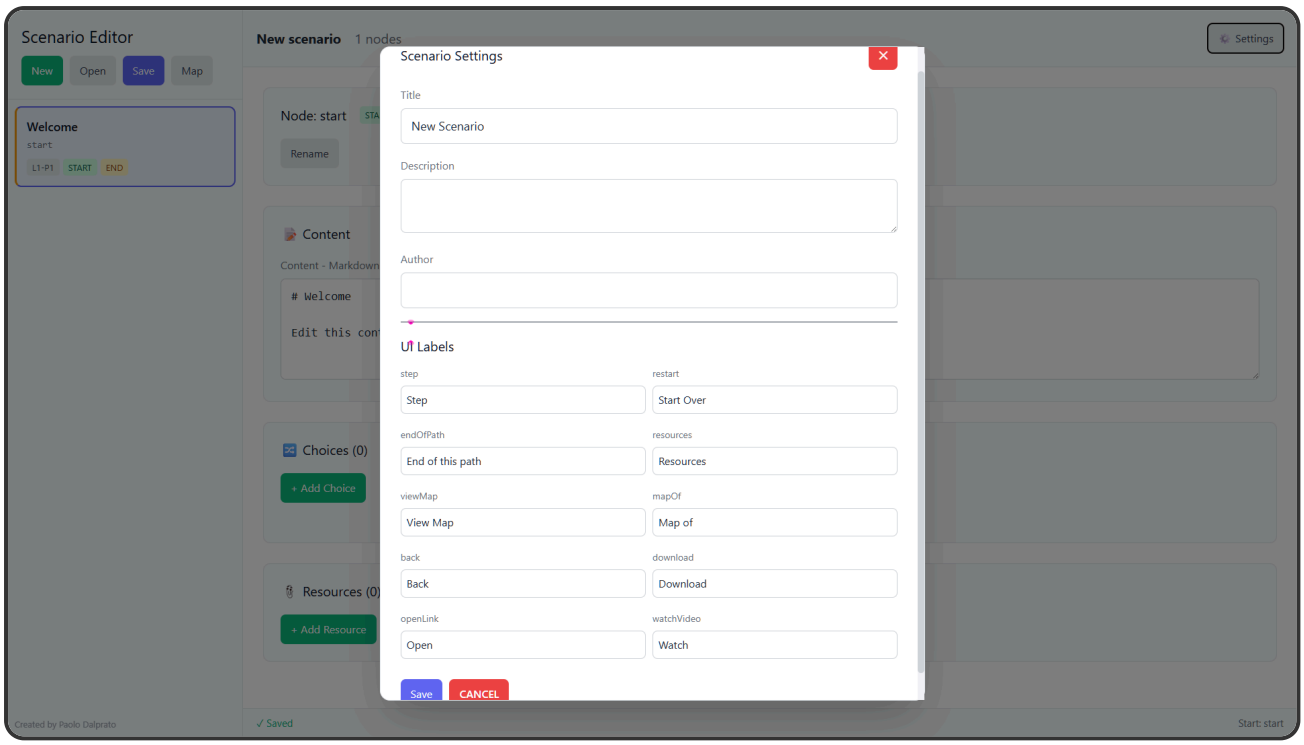
Clicking **+ Add Resource** adds a resource with three required fields:

- **Type**: the resource type
  - *Link*: link to a web page
  - *Download*: downloadable document (must be accessible via URL)
  - *Video*: YouTube video (use the share link)
- **URL**: the resource address, can be either a web link or a relative URL to a local resource. In this case, it's convenient to define a subfolder to collect elements, the URL could be "**docs/element1.pdf**"
- **Label**: the description that appears to the user in the Navigator

Visual status indicators work the same as for Choices. The **Save Resource** button saves only if all fields are filled in.

Settings: metadata and translations

The Settings button opens a panel to configure scenario metadata and interface labels.



(https://docs.ai-know.pro/choicemap-en/img/scenario-metadata.png)

Metadata:

- **Title**: the scenario name, visible in the Navigator header
- **Description**: brief description (optional)
- **Author**: author name (optional)

UI Labels: allows translating or customizing all user interface labels (step, restart, endOfPath, resources, viewMap, back, download, openLink, watchVideo).

The JSON file

A scenario is saved as a structured JSON file in three sections:

(https://docs.ai-know.pro/choicemap-en/img/scenario-json.png)

- **meta**: metadata (title, description, author)
- **translations**: customized labels
- **startNode**: the identifier of the start node
- **nodes**: the object containing all nodes with their content, choices, and resources

Saving

The Save button generates a JSON file. There is no auto-save: save regularly during work.

In the status bar at the bottom, the state is always visible:

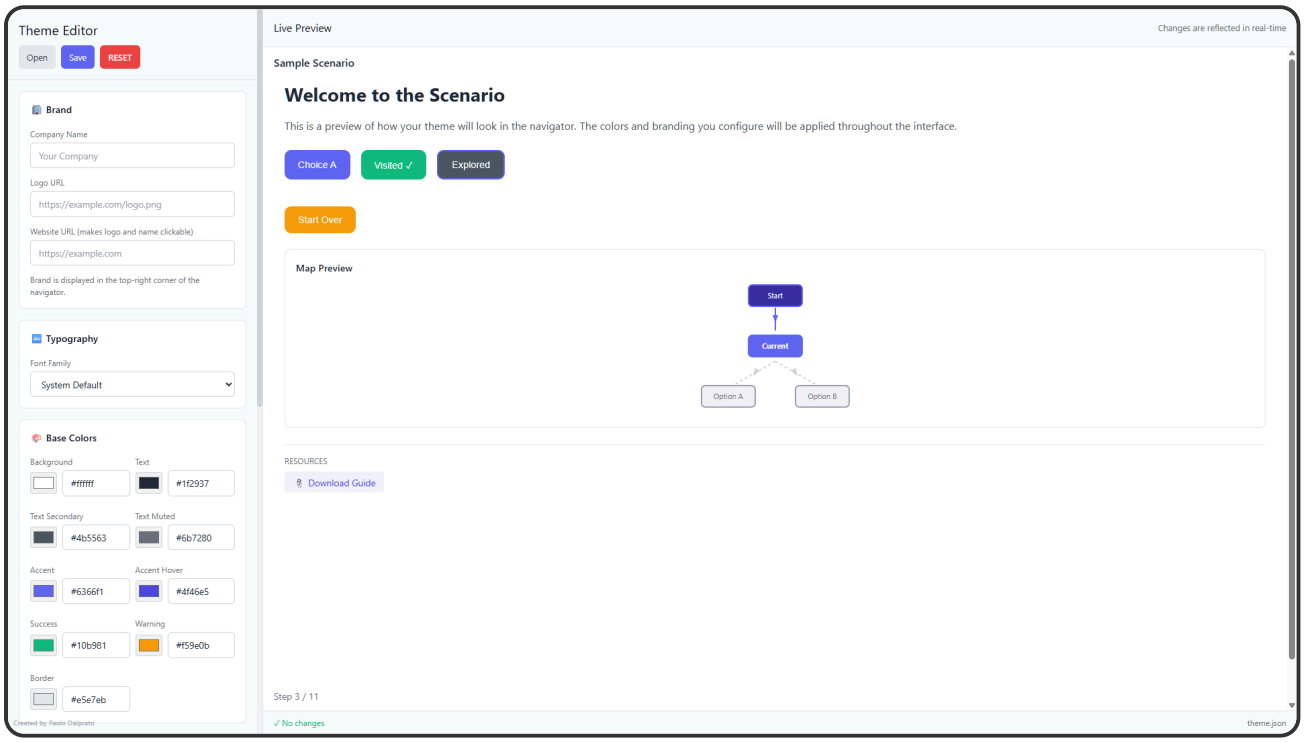
- "● Unsaved" (orange): there are unsaved changes
- "✓ Saved" (green): everything saved

Theme Editor interface

The Theme Editor allows customizing the Navigator's visual appearance without modifying code. Every change is immediately reflected in a preview, allowing you to see the final result while working. The theme is saved in a JSON file.

General layout

The interface is divided into two main side-by-side areas.



(https://docs.ai-know.pro/choicemap-en/img/theme-general.png)

Left column

In the left column, at the top there are buttons for general actions:

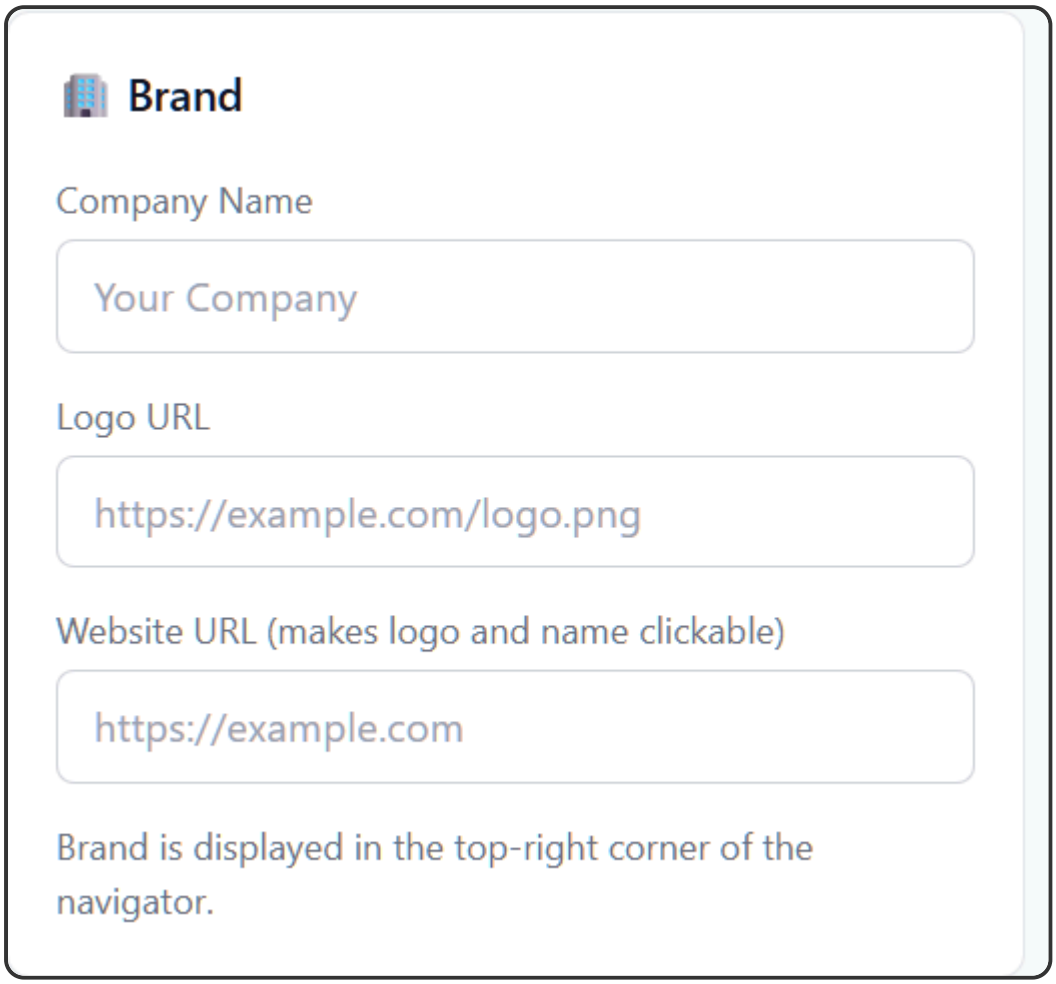
- **Open**: to open a theme file.
- **Save**: to save the theme to a file.
- **RESET**: to restore the theme to default values.

Below the buttons are control panels organized in expandable sections, each section groups related settings. The sections are:

- **Brand**: contains corporate branding elements.
- **Typography**: defines the font to use.
- **Colors**: defines colors for various graphic elements, subdivided into blocks by element type.

Brand section

The brand section controls the corporate identity elements that appear in the Navigator in the top right corner.

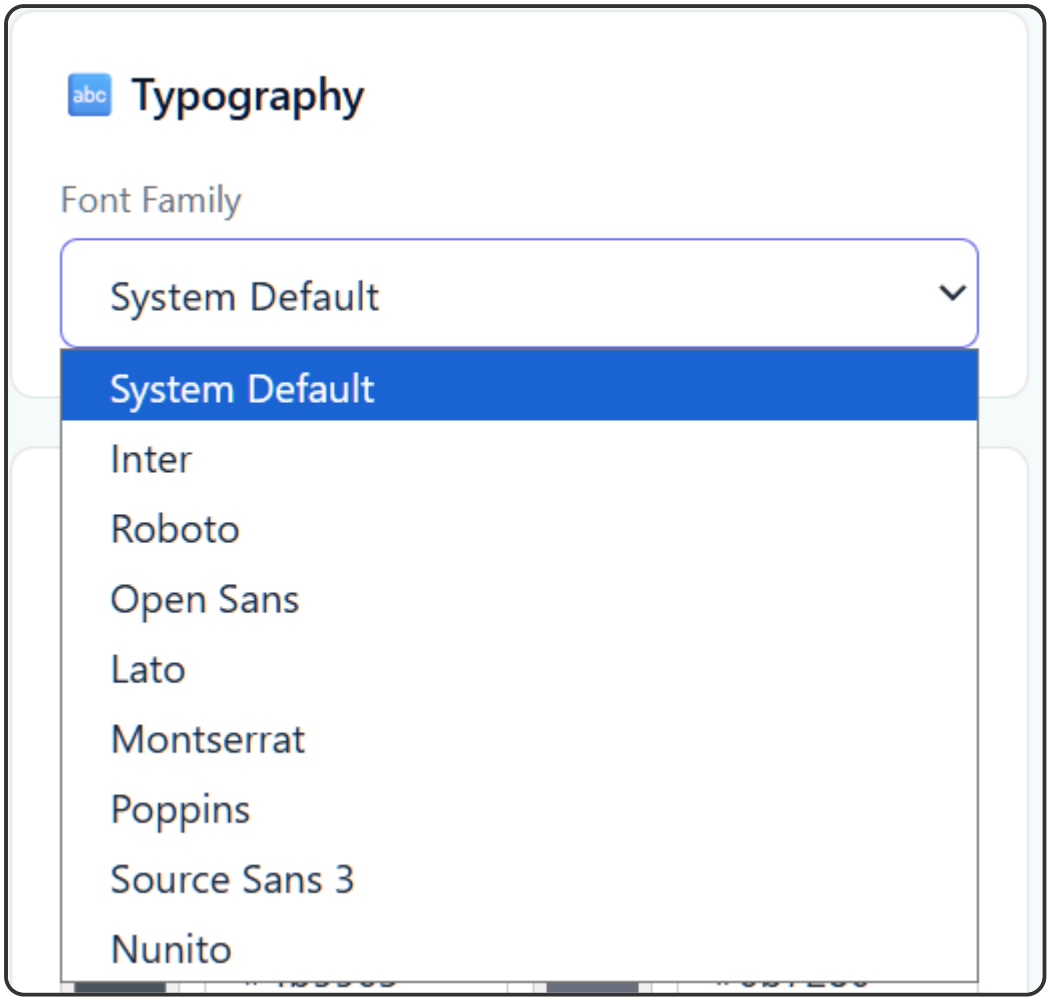


(https://docs.ai-know.pro/choicemap-en/img/theme-brand.png)

- **Company Name**: the field for the company name allows entering text that appears next to or below the logo. If you don't want to show the name, just leave the field empty.
- **Logo URL**: the logo field accepts a URL pointing to the logo image. It can be an external URL or a relative path to a file in the project folder or, better, to a subfolder. Dimensions are automatically adapted to not interfere with the content.
- **Website URL**: the website URL makes the logo and company name (if present) clickable. The user can click to visit the site in a new tab.

Typography section

The typography section controls the font used throughout the interface.



(https://docs.ai-know.pro/choicemap-en/img/theme-typography.png)

A dropdown menu presents the available options. The first option is "System Default", which uses the device's system font without loading external fonts.

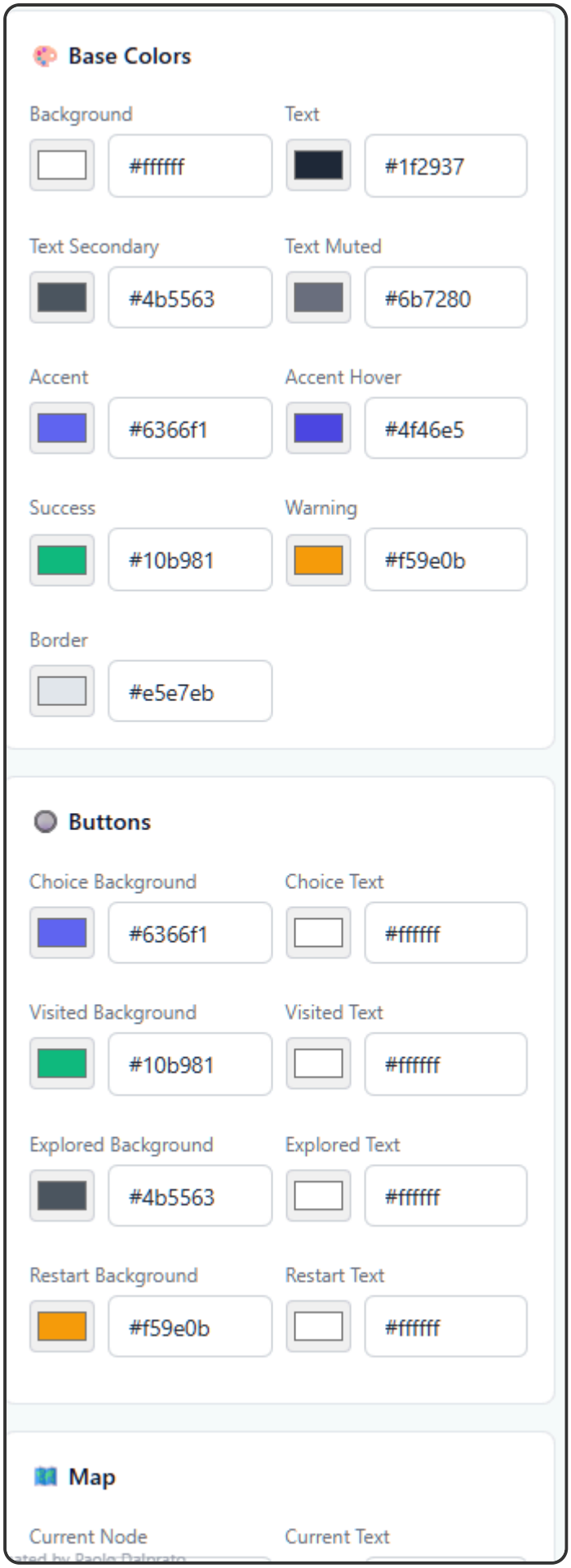
The other options are Google Fonts: - Inter - Roboto - Open Sans - Lato - Montserrat - Poppins - Source Sans 3 - Nunito

When selecting one of these, the font is automatically loaded from Google and applied to the interface.

The preview immediately shows the effect of the font change on headings, text, and buttons.

Colors section

The colors section is the most detailed, with controls for every interface element.



- **Base Colors**: general colors define the page background, the main text color, the secondary text color (used for less important information), the accent color (used to highlight interactive elements). Each control is a color picker that opens a panel to choose the desired shade. The hexadecimal value is visible and directly editable for those who prefer to enter precise color codes.
- **Buttons**: choice buttons have separate controls for each state: background color and text color for unexplored choices, for choices already made, for choices explored but not in the current path. This chromatic distinction helps the user understand what they've already seen when going back to explore alternative paths. The restart button has its own colors, typically different from choice buttons to visually distinguish it.
- **Map**: the map block controls the tree visualization colors: the current node, visited nodes, nodes not yet reached, connection lines. These colors should be consistent with button colors to maintain a consistent visual language.

Live preview

The preview area on the right shows a working Navigator with a sample scenario and map. Every change to the controls is instantly reflected in the preview.

The preview uses a simple predefined scenario. To see how the theme works with the real scenario, you need to save the theme and load it in the actual Navigator.

Typical workflow

The typical flow involves starting from an existing theme or default values, modifying settings while checking the effect in the preview, saving the result as a JSON file.

The saved theme file must then be specified in the project's `config.json` configuration file, associating it with the desired scenario.

It's possible to create multiple different themes for the same scenario, for example a light and dark version, or customized themes for different clients.

Creating a scenario

This guide walks through creating a complete scenario step by step, from planning to publication. For interface details, see Scenario Editor Interface (https://docs.ai-know.pro/choicemap-en/scenario-editor-interface/).

Plan before building

Before opening the editor, it's useful to have a clear picture of the path structure:

- What is the starting point?
- What are the key decision points?
- What are the possible endings?
- Are there paths that reconverge or loops?

A sketch on paper or a mind map helps visualize the structure. It doesn't need to be perfect: the scenario will evolve during creation, but starting with a clear idea prevents getting lost.

Recommended workflow

1. Configure metadata

Open `scenario-editor.html` in your browser. The editor presents a new empty scenario containing only the Start node.

Before working on nodes, set up metadata by clicking **Settings**:

- **Title**: the name that will appear in the Navigator header
- **Description**: a brief description (optional, useful for remembering the purpose)
- **Author**: the author's name (optional)

If the scenario will be in a language other than English, or you want to use different button texts, you must also customize the **UI Labels** to translate button texts (**Step**, **Start Over**, **End of this path**, etc.).

2. Build from the initial node

Start from the Start node and proceed depth-first:

1. Select the node in the left column

2. Write content in the **Content** block using Markdown syntax (https://docs.ai-know.pro/choicemap-en/scenario-editor-interface/#content)
3. Add choices in the **Choices** block
4. For each choice, select an existing node or create a new one with **+ Create new node**

Continue this way for all nodes, branching the structure according to the initial plan.

> **Identifier conventions**
>
> Use short, descriptive names without spaces: `intro`, `question_1`, `answer_ok`, `positive_ending`. Identifiers are not visible to users but help navigate the structure.

3. Verify with the map

Periodically, use the **Map** button to visualize the entire structure. The map shows:

- The general flow of paths
- Any orphan nodes (red border) to connect or delete
- Terminal nodes (orange border)
- Loops, highlighted with orange connections

For color code details, see Map view (https://docs.ai-know.pro/choicemap-en/scenario-editor-interface/#map-view).

4. Save regularly

There is no auto-save. Click **Save** frequently to avoid losing work. The button generates a JSON file to download.

The status bar at the bottom always shows if there are unsaved changes.

Managing terminal nodes

A node without choices is automatically a terminal node. In the Navigator, instead of buttons, the message "End of path" and a restart button appear.

- To create a terminal: remove all choices from the node
- To transform a terminal into an intermediate node: add at least one choice

Managing orphan nodes

An orphan node exists in the scenario but is not reachable from the start node through any path. The editor marks them with the "orphan" label in the list and red border in the map.

Orphan nodes are typically created when:

- A node that was the only connection to other nodes is deleted
- A choice that was the only path to a node is removed

To resolve: connect the node by adding a choice that points to it from some other node, or delete it if no longer needed.

Deleting nodes safely

To delete a node, select it and click **DELETE** in the Node block.

> **Warning**
>
> Deleting a node that has children can make those children orphans if they have no other connections.

Best practice: delete nodes starting from leaves (terminal nodes) and working back toward the root. This way you don't accidentally create orphans.

Creating loops

It's possible to create paths that go backward by connecting a node to one of its ancestors. In the map view, these connections are highlighted in orange.

Typical use cases:

- Allow users to review a section
- Create a main menu to return to
- Handle errors with "try again"
- Represent **repeat ... until** structures (or equivalents)

Adding resources

To attach documents, links, or videos to a node, use the **Resources** block. For details on available types and configuration, see Resources (https://docs.ai-know.pro/choicemap-en/scenario-editor-interface/#resources).

Resources appear in the Navigator as buttons below the node content.

> **Local resources**
>
> For local files (PDFs, documents), it's recommended to create a `docs/` subfolder and use relative paths like `docs/manual.pdf`.

Modifying an existing scenario

To resume work on a scenario:

1. Click **Open** and select the JSON file
2. The scenario loads with all nodes in the left column
3. Select a node to modify it
4. Save changes with **Save**

From creation to publication

Once the scenario is complete:

1. Save the JSON file with a descriptive name (e.g., `new-employee-onboarding.json`)
2. Copy the file to the Navigator folder
3. Modify `config.json` to point to the new scenario
4. Test by opening `choicemap.html`

For details on publication options (intranet, GitHub Pages, embedding), see Publication (https://docs.ai-know.pro/choicemap-en/publication/).

Content structure

This section describes the file architecture and JSON format that composes a scenario. Knowledge of this structure is not necessary for using the Scenario Editor, but is useful for advanced modifications, debugging, or generating scenarios programmatically.

File architecture

ChoiceMap clearly separates the framework (code) from content (data). HTML files contain the application and should not be modified; all customizations happen through JSON files.

HTML files (framework)

The three HTML files are complete applications that require no modification:

File	Function
`choicemap.html`	Displays scenarios for end users
`scenario-editor.html`	Editor for creating and modifying scenarios
`theme-editor.html`	Editor for customizing appearance

These files load React and Tailwind CSS from CDN and read content from JSON files. It's not necessary (nor recommended) to modify them.

config.json (the director)

The `config.json` file is the entry point that tells the Navigator what to load:

```
{
    "scenario": "scenario-my-project.json",
    "theme": "theme.json",
    "showCredits": true
}
```

Field	Function
`scenario`	Name of the JSON file containing the scenario to display
`theme`	Name of the JSON file containing the theme to apply
`showCredits`	Show/hide footer with author credits

By changing the `scenario` value and reloading the Navigator, you switch to a different scenario without modifying anything else. The same applies to `theme`.

defaults.json (default values)

Contains all default values for the theme: colors, fonts, branding configuration. It's the "base theme" from which all others start.

```
{
    "brand": {
        "name": "",
        "logo": "",
        "website": ""
    },
    "colors": {
        "background": "#ffffff",
        "text": "#1f2937",
        "accent": "#6366f1"
    }
}
```

This file should not be modified. To customize appearance, create a separate theme file.

theme.json (customizations)

The theme file contains only customizations relative to defaults. The Navigator reads `defaults.json` first, then overwrites with values present in `theme.json`.

If `theme.json` contains only the company logo, all other values (colors, fonts) are taken from `defaults.json`:

```
{
    "brand": {
        "name": "Company XYZ",
        "logo": "images/logo.png"
    }
}
```

This approach allows creating minimal themes that specify only what changes.

shared-styles.css (editors only)

Contains CSS styles shared between Scenario Editor and Theme Editor. It's not used by the Navigator and doesn't need to be included when publishing a scenario for end users.

The scenario file

Each scenario is a JSON file with a well-defined structure:

```
{
    "meta": {
        "title": "Scenario title",
        "description": "Optional description",
        "author": "Author name"
    },
    "translations": {
        "step": "Step",
        "restart": "Start Over",
        "endOfPath": "End of this path",
        "resources": "Resources",
        "viewMap": "View Map",
        "mapOf": "Map of",
        "back": "Back",
        "download": "Download",
        "openLink": "Open",
        "watchVideo": "Watch"
    },
    "startNode": "start",
    "nodes": {
        "start": {
            "content": "# Welcome\n\nContent in **Markdown**.",
            "choices": [
                { "text": "First option", "next": "node_a" },
                { "text": "Second option", "next": "node_b" }
            ],
            "resources": []
        }
    }
}
```

Meta section

The `meta` section contains the scenario's descriptive information.

Field	Required	Use
`title`	Yes	Appears in the Navigator header
`description`	No	Notes for scenario managers
`author`	No	Shown in footer if `showCredits` is active

Translations section

Defines interface labels, allowing scenario localization in any language.

Key	Use	Example
`step`	Progress indicator	Step
`restart`	Restart button	Start Over
`endOfPath`	End of path message	End of this path
`resources`	Resources section title	Resources
`viewMap`	Open map button	View Map
`mapOf`	Map title	Map of
`back`	Back button	Back
`download`	Download label	Download
`openLink`	Link label	Open
`watchVideo`	Video label	Watch

For a quiz, you could use "Question" instead of "Step" and "Try Again" instead of "Start Over".

startNode field

Indicates which node is the scenario's starting point. The value must correspond to a key present in the `nodes` object.

Typically `"start"` is used by convention, but it can be any valid identifier.

Nodes section

An object where each key is a node identifier and the value contains the node data.

Node identifier: each node's key is its unique ID, used internally for connections. Recommended conventions: - Only lowercase letters, numbers, and underscores - Descriptive but short names: `intro`, `question_1`, `answer_ok` - Avoid spaces and special characters

Node content: the `content` field contains text in Markdown format. For supported syntax, see Content (https://docs.ai-know.pro/choicemap-en/scenario-editor-interface/#content).

To create line breaks in JSON, use `\n`:

```
"content": "First line\n\nSecond line after a space"
```

Node choices: the `choices` field is an array of objects defining choice buttons:

```
"choices": [
    { "text": "Button text", "next": "destination_node_id" }
]
```

- `text`: what the user reads on the button
- `next`: the destination node identifier

A node without choices (empty array `[]`) is a terminal node.

Node resources: the `resources` field is optional and defines attachments:

```
"resources": [
    { "type": "download", "label": "Download PDF", "url": "docs/guide.pdf" },
    { "type": "link", "label": "Learn more", "url": "https://example.com" },
    { "type": "video", "label": "Video tutorial", "url": "https://youtube.com/watch?
v=..." }
]
```

Type	Behavior
`download`	Downloads the file
`link`	Opens in new tab
`video`	Opens in new tab

Validation

The Navigator shows errors if something doesn't work in JSON files. Common errors:

- Malformed JSON (missing commas, unclosed quotes)
- `startNode` pointing to a non-existent node
- Choices pointing to non-existent nodes
- Missing or incomplete `translations` section

The Scenario Editor always generates valid JSON, but manual modifications can introduce errors. In case of problems, verify syntax with an online JSON validator.

Customizing the theme

This guide walks through customizing the Navigator's visual appearance. For interface details, see Theme Editor Interface (https://docs.ai-know.pro/choicemap-en/theme-editor-interface/).

Recommended workflow

1. Open the Theme Editor (`http://localhost:8000/theme-editor.html`)
2. Start from the default theme or load an existing one with **Open**
3. Modify settings while checking the effect in the preview
4. Save with **Save** and copy the file to the project folder
5. Update `config.json` to point to the new theme
6. Verify in the actual Navigator with a complete scenario

The Theme Editor preview uses a simplified scenario. To see the final result with real content, you need to test in the Navigator.

Branding tips

Logo

Format	Recommended use
SVG	Optimal scalability, lightweight file
Transparent PNG	Good quality, universal compatibility
JPG	Only if necessary, avoid for logos

Avoid overly large images: the logo is automatically resized but heavy files slow down loading. Recommended size: maximum 200x60 pixels or equivalent.

The field accepts both external URLs (`https://company.com/logo.png`) and relative paths (`images/logo.png`). For distributable projects, prefer relative paths.

Typography tips

Context	Recommended fonts
Corporate/professional	Inter, Roboto, Source Sans 3
Creative/informal	Poppins, Nunito, Montserrat
Maximum compatibility	System Default

"System Default" doesn't load external fonts: it uses the device's font. It's the lightest choice and guarantees readability on any system.

Color tips

Contrast

Text must be readable on the background. Safe combinations:

Background	Main text	Secondary text
White (#ffffff)	Dark gray (#1f2937)	Medium gray (#4b5563)
Dark gray (#1f2937)	White (#ffffff)	Light gray (#9ca3af)

Accent color

The accent color defines the interface personality. Use the main brand color for visual consistency. Verify it's saturated enough to stand out as an interactive element.

Button states

Chromatic distinction between buttons helps users orient themselves:

- **Available choices**: accent color, invites action
- **Choices already made**: different color (green by default), visual confirmation
- **Explored choices**: neutral gray, indicates "already seen in another path"

Maintain consistency between button colors and map colors for a uniform visual language.

Saving and applying

The **Save** button generates a JSON file. Rename it with a descriptive name:

```
theme-company-2024.json
theme-safety-course.json
theme-dark.json
```

To apply the theme, update `config.json`:

```
{
    "scenario": "scenario-course.json",
    "theme": "theme-company-2024.json",
    "showCredits": false
}
```

Reload the Navigator (Ctrl+Shift+R) to see the applied theme.

Managing multiple themes

You can create multiple themes for different contexts:

- **Brightness variants**: light theme and dark theme
- **Different clients**: a theme for each client with their branding
- **Different contexts**: formal theme for compliance, vivid theme for onboarding

Changing the `theme` value in config.json switches between themes without modifying the scenario. The same content can have completely different appearances.

Modifying the theme via JSON

For those who prefer working directly on the file, the structure is:

```
{
    "brand": {
        "name": "Company Name",
        "logo": "images/logo.png",
        "website": "https://example.com"
    },
    "typography": {
        "fontFamily": "inter"
    },
    "colors": {
        "background": "#ffffff",
        "text": "#1f2937",
        "textSecondary": "#4b5563",
        "accent": "#6366f1"
    },
    "buttons": {
        "choiceBackground": "#6366f1",
        "choiceText": "#ffffff",
        "visitedBackground": "#10b981",
        "visitedText": "#ffffff",
        "exploredBackground": "#e5e7eb",
        "exploredText": "#374151",
        "restartBackground": "#f59e0b",
        "restartText": "#ffffff"
    },
    "map": {
        "nodeCurrent": "#6366f1",
        "nodeCurrentText": "#ffffff",
        "nodeVisited": "#3730a3",
        "nodeVisitedText": "#ffffff",
        "nodeUnvisited": "#f3f4f6",
        "nodeUnvisitedText": "#6b7280",
        "nodeUnvisitedBorder": "#d1d5db",
        "lineVisited": "#6366f1",
        "lineUnvisited": "#d1d5db"
    }
}
```

All fields are optional. Missing values are taken from `defaults.json`. This allows creating minimal themes that specify only what changes:

```
{
    "brand": {
        "name": "Company XYZ",
        "logo": "images/logo-xyz.png"
    },
    "colors": {
        "accent": "#dc2626"
    }
}
```

Available values for `fontFamily`: `system`, `inter`, `roboto`, `open-sans`, `lato`, `montserrat`, `poppins`, `source-sans-3`, `nunito`.

Publication

This section covers work organization, roles involved, and options for making scenarios accessible to end users.

Roles and responsibilities

In a corporate context, two distinct roles interact with ChoiceMap.

End user

The end user uses **only the Navigator** to consume scenarios. They have no access to editors and don't modify content.

Aspect	Detail
Uses	Only `choicemap.html`
Accesses	Publication folder with minimal files
Required skills	None, just open a link in browser
Visible files	Navigator + config + scenario + theme

Administrator

The administrator creates, modifies, and publishes scenarios. They have full access to all tools.

Aspect	Detail
Uses	Navigator + Scenario Editor + Theme Editor
Accesses	Complete working repository
Required skills	File management, understanding of structure
Managed files	All project files

In small contexts, the same person may cover both roles. In larger organizations, roles are typically separate.

Administrator workflow

Working repository

The administrator maintains a local repository (a folder on their computer or server) with all project files:

```
choicemap/
├── choicemap.html    # Navigator
├── scenario-editor.html        # Scenario editor
├── theme-editor.html           # Theme editor
├── config.json                 # Configuration
├── defaults.json               # Default values
├── shared-styles.css           # Shared styles
├── scenarios/                  # Scenarios in progress
│   ├── onboarding-v1.json
│   ├── onboarding-v2.json
│   └── compliance-draft.json
├── themes/                     # Custom themes
│   └── corporate-theme.json
└── resources/                  # Attached documents
    ├── manual.pdf
    └── checklist.docx
```

Typical work cycle

1. **Create or modify the scenario** using `scenario-editor.html`
2. **Customize the theme** using `theme-editor.html` (if needed)
3. **Configure** `config.json` to point to the correct scenario and theme
4. **Test** by opening `choicemap.html` in the browser
5. **Iterate** until achieving the desired result
6. **Publish** by copying only the necessary files to the destination

Versioning

It's good practice to maintain scenario versions:

- keep previous versions (e.g., `scenario-v1.json`, `scenario-v2.json`)
- annotate significant changes
- use a version control system (Git) for complex projects

Preparing for publication

Files needed for end users

To publish a scenario, **only** these files are needed:

File	Required	Notes
`choicemap.html`	Yes	The Navigator
`config.json`	Yes	Points to scenario and theme
`defaults.json`	Yes	Default values
Scenario JSON file	Yes	E.g., `scenario-onboarding.json`
Theme JSON file	Only if customized	E.g., `corporate-theme.json`
Resources folder	Only if used	PDFs, documents, etc.

Do not include in publication:

- `scenario-editor.html` (not needed for end users)
- `theme-editor.html` (not needed for end users)
- `shared-styles.css` (used only by editors)
- Draft scenarios or test versions
- Administrator working files

Configure config.json

Before publishing, verify that `config.json` points to the correct files:

```
{
    "scenario": "scenario-onboarding.json",
    "theme": "corporate-theme.json",
    "showCredits": false
}
```

The `showCredits: false` option removes the footer with credits, for a cleaner appearance.

Publication destinations

Corporate intranet

For internal use, copy files to a shared folder on the corporate file server.

```
\\server\share\training\
├── choicemap.html
├── config.json
├── defaults.json
└── scenario-onboarding.json
```

Users access by directly opening the HTML file or via a link on the intranet.

> **Local files vs web server**
>
> The Navigator also works when opened directly as a local file, without needing a web server, as long as all JSON files are in the same folder.

Public website

For web publication, upload files to any static hosting:

- **GitHub Pages** (free)
- **Netlify**, **Vercel** (free with limitations)
- **Amazon S3**, **Google Cloud Storage**
- Traditional hosting

No PHP, database, or other server-side component support is needed.

Corporate LMS

To integrate into a Learning Management System:

- some LMSs support uploading HTML packages
- others allow embedding via iframe
- verify the specific LMS security policies

GitHub Pages

GitHub Pages is the simplest solution for publishing to the web for free.

Procedure

1. Create a repository on GitHub
2. Upload necessary files (only those for end users)
3. In Settings → Pages, enable publishing from main branch
4. Wait for deploy (1-2 minutes)

Resulting URL

```
https://username.github.io/repository-name/choicemap.html
```

Example: if the repository is `training-onboarding` by user `company-xyz`:

```
https://company-xyz.github.io/training-onboarding/choicemap.html
```

Updates

To update a published scenario:

1. Modify files locally
2. Upload new versions to GitHub
3. Wait for automatic rebuild

If changes don't appear immediately, force refresh (Ctrl+Shift+R) or wait a few minutes for caching.

Managing multiple scenarios

An organization may have different scenarios: onboarding, compliance, technical training.

Option 1: Separate folders

Each scenario has its own folder with all necessary files:

```
training/
├── onboarding/
│   ├── choicemap.html
│   ├── config.json
│   ├── defaults.json
│   └── scenario-onboarding.json
├── compliance/
│   ├── choicemap.html
│   ├── config.json
│   ├── defaults.json
│   └── scenario-compliance.json
```

Advantages: each scenario is independent, easy to manage and update separately.

Distinct URLs: - `.../training/onboarding/choicemap.html` - `.../training/compliance/choicemap.html`

Option 2: Shared installation

A single Navigator installation, multiple selectable scenarios:

```
training/
├── choicemap.html
├── defaults.json
├── config-onboarding.json
├── config-compliance.json
├── scenario-onboarding.json
└── scenario-compliance.json
```

Requires renaming the appropriate config to `config.json` to change scenarios, or modifying the Navigator to accept a URL parameter (advanced customization).

Embedding in other sites

The Navigator can be embedded in existing pages via iframe.

Basic code

```
<iframe
    src="https://yourdomain.com/choicemap.html"
    width="100%"
    height="600"
    frameborder="0">
</iframe>
```

Responsive version

```
<div style="position: relative; width: 100%; height: 100vh;">
    <iframe
        src="https://yourdomain.com/choicemap.html"
        style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; border:
none;"
        frameborder="0"
        allowfullscreen>
    </iframe>
</div>
```

Considerations

- Verify any CORS restrictions if Navigator and page are on different domains
- External links open in the iframe unless using `target="_blank"`
- Keyboard navigation may behave differently inside an iframe

Security and access

ChoiceMap has no built-in authentication. Anyone with the URL can access it.

To restrict access:

- Private repository on GitHub (requires Pro account for GitHub Pages)
- Hosting on corporate server behind VPN
- Web server level authentication (basic auth, corporate SSO)
- Non-indexed URL (security through obscurity, not recommended for sensitive content)

For scenarios with confidential information, carefully evaluate infrastructure and corporate policies.

Quick reference

Operational checklist for those already familiar with the manual who want a reminder of essential steps.

Start local server

Windows (double-click batch scripts):

- `start-navigator.bat` → opens the Navigator
- `start-scenario-editor.bat` → opens the Scenario Editor
- `start-theme-editor.bat` → opens the Theme Editor

Manual (Mac/Linux or without scripts):

```
python -m http.server 8000
```

Addresses:

- Navigator: `http://localhost:8000/choicemap.html`
- Scenario Editor: `http://localhost:8000/scenario-editor.html`
- Theme Editor: `http://localhost:8000/theme-editor.html`

Create a scenario

1. **New** → creates empty scenario
2. **Settings** → title, description, author
3. Select node → write **Content**
4. **+ Add Choice** → button text + destination node
5. Repeat for all nodes
6. **Map** → verify structure (no red borders = ok)
7. **Save** → downloads JSON file

Modify an existing scenario

1. **Open** → select JSON file
2. Modify nodes and choices

3. **Save**

Customize the theme

1. Open Theme Editor
2. **Brand** → logo, company name, website
3. **Typography** → choose font
4. **Colors** → background, text, buttons, map
5. Verify in preview
6. **Save** → downloads JSON file

Configure config.json

```
{
    "scenario": "scenario-name.json",
    "theme": "theme-name.json",
    "showCredits": false
}
```

Test locally

1. Update config.json with correct file names
2. Reload the Navigator (Ctrl+Shift+R)
3. Go through the entire scenario verifying paths

Publish

Required files (only these):

- `choicemap.html`
- `config.json`
- `defaults.json`
- Scenario JSON file
- Theme JSON file (if customized)
- Resources folder (if used)

Do not include: scenario-editor.html, theme-editor.html, shared-styles.css

GitHub Pages:

1. Create repository
2. Upload required files
3. Settings → Pages → enable from main branch
4. URL: `https://username.github.io/repo-name/choicemap.html`

Map color codes (Scenario Editor)

Border color	Meaning
Green	Start node (START)
Orange	Terminal node (END)
Red	Orphan node (problem!)
Purple	Selected node

Orange connections = loops (go backward).

Markdown in content

```
# Large heading
## Subheading
**bold**
*italic*
- list item
> quote
```

Quick troubleshooting

Problem	Solution
Blank page when opening HTML	Start local server
Changes not visible	Ctrl+Shift+R to force refresh
Port 8000 busy	Use another port: `python -m http.server 8080`
Orphan nodes (red border)	Connect or delete them