node >=18.0.0

A powerful collection of scripts to generate and manage design tokens for your design system. Each wizard specializes in creating specific types of tokens, ensuring consistency and efficiency across your projects.

🧙 Getting Started

1. **Install Node.js**
   Download and install Node.js on your machine.

2. **Install VS Code**
   Download and install Visual Studio Code for an enhanced development experience.

3. **Open Terminal**

   - **VS Code:** Press `Ctrl + ` ` (Windows/Linux) or `Cmd + ` ` (Mac)
   - **System Terminal:**

     - Windows: `Windows + R`, type `cmd`
     - Mac: `Command + Space`, type `terminal`
     - Linux: `Ctrl + Alt + T`

4. **Download/Clone the Repository**

   Download ZIP

   or

   Clone Repo git clone https://github.com/fulviabuonanno/design-tokens-wizards.git cd design-tokens-wizards

5. **Install Dependencies**

   ```
   npm install
   ```

6. **Run the Scripts**
   Choose from the following wizards:

Token Wizard	Script Name	Run Command	Description	Version
🟡 COLOR WIZ	`color-wiz.js`	`npm run color`	Generate and manage color tokens	2.6.0 🔥
🔴 TYPOGRAPHY WIZ	`typo_wiz.js`	`npm run typo`	Generate and manage typography tokens	1.2.1 ✅
🟣 SPACE WIZ	`space_wiz.js`	`npm run space`	Generate and manage spacing tokens	1.7.0 🔥
🔵 SIZE WIZ	`size_wiz.js`	`npm run size`	Generate and manage size tokens	1.7.0 🔥
🟢 BORDER RADIUS WIZ	`radii_wiz.js`	`npm run radii`	Generate and manage border radius tokens	1.7.0 🔥

Spell	Script Name	Run Command	Description	Version
MERGE SPELL	`merge_spell.js`	`npm run merge`	Combine all token files into a single unified file	1.2.0
CLEAR SPELL	`clear_spell.js`	`npm run clear`	Remove all generated output files in one swift command	1.2.1 ✅

Legend:
✅ Patch // 🔥 Minor Change // 🆙 Major Change

🗂️ Project Structure

```
src/
  wizards/         # All wizard scripts (color, typo, space, size, radii)
  spells/          # Utility scripts (merge, clear)
  config/          # Config and helper scripts
output_files/      # Where generated tokens are saved
  tokens/
    color/
    typography/
    space/
    size/
    border-radius/
  css/
  scss/
  final/
```

🎨 Color Tokens Wizard

Version 2.6.0

Managing color tokens can sometimes feel as magical as mastering alchemy, but with the Color Tokens Wizard 🧙 , your journey to conjuring a luminous palette is a breeze. Begin with a base hue that will set the spellbinding tone, and the wizard will guide you through creating a dazzling spectrum of tokens. Whether you're defining a signature brand shade or curating an entire color system, let this wizard transform your creative vision into vibrant reality.

1. **Invoke the Spell**
   Begin your journey by running the Color Tokens Wizard script. Open your terminal and cast the spell with:

   ```
   npm run color
   ```

2. **Select Token Type**
   Choose between global and semantic colors:

   - **Global Colors:** Base colors that form the foundation of your design system
   - **Semantic Colors:** (Coming soon) Colors based on their meaning and purpose

   Note: The wizard currently supports **global colors**. Semantic colors are under development and will be available in a future update.

3. **Define Global Category** (Optional)
   Organize your colors into logical groups:

   - primitives
   - foundation
   - core

- basics
- essentials
- global
- roots
- custom

4. **Set Naming Level** (Optional)
   Provide context about how the color should be used:

   - color
   - colour
   - palette
   - scheme
   - custom

5. **Name Your Color**
   Enter a name for your color (e.g., `blue`, `yellow`, `red`).

6. **Select Base Color**
   Provide a HEX color code (e.g., `#FABADA`) to use as your base color.

7. **Choose Scale Type**
   Select how you want to generate your color stops:

   - **Incremental:** Generate stops using an incremental step
     - 10 in 10 (e.g., 10, 20, 30, 40)
     - 25 in 25 (e.g., 25, 50, 75, 100)
     - 50 in 50 (e.g., 50, 100, 150, 200)
     - 100 in 100 (e.g., 100, 200, 300, 400)
   - **Ordinal:** Create a sequence
     - Padded (e.g., 01, 02, 03, 04)
     - Unpadded (e.g., 1, 2, 3, 4)
   - **Alphabetical:** Use letters

        - Uppercase (e.g., A, B, C, D)
        - Lowercase (e.g., a, b, c, d)
    - **Semantic Stops:** Generate semantic stops

        - 1 stop: base
        - 2 stops: dark, base, light
        - 4 stops: darker, dark, base, light, lighter
        - 6 stops: darkest, darker, dark, base, light, lighter, lightest
        - 8 stops: ultra-dark, darkest, darker, dark, base, light, lighter, lightest, ultra-light
        - 10 stops: ultra-dark, darkest, darker, dark, semi-dark, base, semi-light, light, lighter, lightest, ultra-light
8. **Set Number of Stops**
   Specify how many color stops you want to generate (1-20).

9. **Customize Color Ranges**
   Define how your colors mix with white and black:

    - **Default Range:** 10% to 90% mix
    - **Custom Range:** Define your own mix percentages (0-100%)

        - Lower values create more extreme light/dark variations
        - Higher values create more subtle variations
        - Values of 0% or 100% will result in pure white/black colors

10. **Preview the Tokens**
    The wizard will show your color's preview along with the stops table. The token name will be constructed based on your selections:

```
 Type: Global
Category: primitives (optional)
Level: color (optional)
Name: blue
┌──────────┬─────────────┬──────────┐
│ Scale    │ HEX         │ Sample   │
├──────────┼─────────────┼──────────┤
```

```
| base    | #FABADA   | [🪄]    |
| 01      | #F0E3D2   | [🪄]    |
| 02      | #E6D6BF   | [🪄]    |
| ...     | ...       | ...     |
└─────────┴───────────┴─────────┘
```

11. **Generate Your Artifacts**
    Once confirmed, the wizard will:
    - Export your tokens in Tokens Studio JSON format Stored in: `output_files/tokens/color` as `color_tokens_{format}.json`
    - Create CSS and SCSS files with your tokens as variables Stored in `output_files/tokens/css/` and `output_files/tokens/scss/` respectively as: `color_variables_{format}.css` and `color_variables_{format}.scss`

12. **Expand Your Palette**
    Add more colors to your palette by repeating the process.

13. **Finalize the Spell**
    Review the output files and integrate your design tokens into your system.

Note: The wizard currently supports global colors only. Semantic colors, categories, and naming levels are under development and will be available in a future update.

Let the art of token crafting infuse your project with endless creativity—and may your colors forever dazzle!

🔤 Typography Tokens Wizard

Version 1.2.0

The Typography Tokens Wizard 🧙‍♂️ is your enchanted companion for crafting a comprehensive typography system. Like a skilled alchemist mixing potent ingredients, this wizard helps you blend font families, sizes, weights, spacing and heights into a harmonious typographic elixir for your design system.

1. **Invoke the Spell**
   Begin your typographic journey by running the Typography Tokens Wizard script:

   ```
   npm run typo
   ```

2. **Choose Your Properties**
   Select which typography properties you wish to configure:

   - Font Families
   - Font Sizes
   - Font Weights
   - Letter Spacing
   - Line Heights

3. **For Each Property:**

   🔤 **Font Family**

   - Name your property (fontFamily, font-family, fonts, ff, or custom)
   - Define 1-3 font families with fallbacks
   - Choose naming convention:

     - Semantic (primary, secondary, tertiary)
     - Purpose-based (title, body, details)

        - Ordinal (1, 2, 3)
        - Alphabetical (a, b, c)

📏 **Font Size**

- Name your property (fontSize, font-size, size, fs, or custom)
- Select scale type:
    - 4-Point Grid
    - 8-Point Grid
    - Modular Scale
    - Custom Intervals
    - Fibonacci Scale
- Choose unit (px, rem, em)
- Define 1-12 sizes with naming convention:
    - T-shirt (xs, sm, md, lg, xl)
    - Incremental (10, 20, 30)
    - Ordinal (1, 2, 3)
    - Alphabetical (a, b, c...)

⚖️ **Font Weight**

- Name your property (fontWeight, font-weight, weight, fw, or custom)
- Select from standard weights (100-900)
- Choose naming convention:
    - T-shirt (xs to xl)
    - Semantic (thin to bold)
    - Ordinal (1 to 5)
    - Purpose-based (body, heading...)

↔️ **Letter Spacing**

- Name your property (letterSpacing, letter-spacing, tracking, ls, or custom)
- Choose scale type:

        - **Predetermined Scale:** (-1.25 to 6.25)
        - **Custom Values:** Define your own values
    - Select unit (em, rem, %)
    - Define 1-7 values with naming convention:

        - T-shirt (xs to xl)
        - Incremental (100, 200...)
        - Ordinal (01, 02... or 1, 2...)
        - Alphabetical (a, b, c...)

↕️ **Line Height**

- Name your property (lineHeight, line-height, leading, lh, or custom)
- Choose scale type:

    - **Predetermined Scale 1:** (1.1, 1.25, 1.5, 1.6, 1.75, 2.0)
    - **Predetermined Scale 2:** (1.0, 1.2, 1.5, 1.6, 2.0)
    - **Custom Values:** Define your own values

- Choose naming convention:

    - T-shirt (xs to xl)
    - Semantic (tight, normal, loose, relaxed, spacious)
    - Ordinal (1 to 5)
    - Purpose-based (body, heading, display, compact, expanded)
    - Incremental (100, 200...)
    - Alphabetical (a, b, c...)

4. **Preview the Tokens**
   For each property, you'll see a preview table showing your configured values:

```
Property: Font Size
┌──────────┬──────────┐
│ Scale    │ Value    │
├──────────┼──────────┤
│ xs       │ 12px     │
│ sm       │ 14px     │
```

```
| md       | 16px    |
| lg       | 18px    |
| xl       | 20px    |
└──────────┴─────────┘
```

5. **Generate Your Artifacts**
   Once confirmed, the wizard will:

   - Export your tokens in Tokens Studio JSON format Stored in: `output_files/tokens/typography/typography_tokens.json`
   - Create CSS and SCSS files with your tokens as variables Stored in `output_files/tokens/css/typography/typography_variables.css` and `output_files/tokens/scss/typography/typography_variables.scss`

6. **Finalize the Spell**
   Review the output files and integrate your typography tokens into your system.

⚠️ **IMPORTANT:** Each step includes accessibility guidelines and recommendations to ensure your typography system is both beautiful and functional. The wizard acts as your trusted advisor, suggesting optimal values while giving you the freedom to customize according to your needs.

■ Space Tokens Wizard

Version 1.7.0

Creating and managing space tokens can be a daunting task, but with the Space Tokens Wizard 🧙 , you can streamline the process and save precious time. Begin with a base space value, and the wizard will generate a variety of space tokens in different units, ready for integration into your design system and development projects.

1. **Invoke the Spell**
   Begin your journey by running the Space Tokens Wizard script:

   ```
   npm run space
   ```

2. **Define the Base Unit**
   The default base unit for space tokens is pixels (px).

3. **Name Your Space Tokens**
   Provide a name for your space tokens (e.g., space, spc).

4. **Select a Scale**
   Choose a predefined scale for your tokens:

   - **4-Point Grid System**
   - **8-Point Grid System**
   - **Modular Scale** (multiplier based)
   - **Custom Intervals**
   - **Fibonacci Scale**

5. **Define the Number of Values**
   Specify how many space values you want to generate (e.g., 6 values for a small-to-large scale).

6. **Choose Naming Criteria**
   Select a naming pattern for your space tokens:

   - **T-shirt Sizes**: (xs, sm, md, lg, xl)
   - **Incremental Numbers**: (100, 200, 300)
   - **Ordinal Numbers**: (1, 2, 3)
   - **Alphabetical**: (A, B, C or a, b, c)

7. **Preview the Tokens**
   The wizard will show your space tokens preview:

```
Name: Space
┌─────────┬─────────┐
│ Scale   │ Value   │
├─────────┼─────────┤
│ 01      │ 16px    │
│ 02      │ 24px    │
│ 03      │ 32px    │
│ 04      │ 40px    │
└─────────┴─────────┘
```

8. **Generate Your Artifacts**
   Once confirmed, the wizard will:

   - Export your tokens in Tokens Studio JSON format Stored in: `output_files/tokens/space/space_tokens_{unit}.json`
   - Create CSS and SCSS files with your tokens as variables Stored in `output_files/tokens/css/space/space_variables_{unit}.css` and `output_files/tokens/scss/space/space_variables_{unit}.scss`

9. **Finalize the Spell**
   Review the output files and integrate your space tokens into your system.

📏 Size Tokens Wizard

Version 1.7.0

Managing size tokens can be daunting, but with the Size Tokens Wizard 🧙 , you can streamline the process and save valuable time. Start with a base size, and let the wizard generate a range of size tokens in various units, ready for integration into your design system and development projects.

1. **Invoke the Spell**
   Begin your journey by running the Size Tokens Wizard script:

   ```
   npm run size
   ```

2. **Define the Base Unit**
   The default base unit for size tokens is pixels (px).

3. **Name Your Size Tokens**
   Provide a name for your size tokens (e.g., size, dimension).

4. **Select a Scale**
   Choose from the predefined scales:

   - **4-Point Grid System**
   - **8-Point Grid System**
   - **Modular Scale** (multiplier based)
   - **Custom Intervals**
   - **Fibonacci Scale**

5. **Define the Number of Values**
   Specify how many size values you want to generate (e.g., 6 values for a small-to-large scale).

6. **Choose Naming Criteria**
   Select a naming pattern for your size tokens:

   - **T-shirt Sizes**: (xs, sm, md, lg, xl)
   - **Incremental Numbers**: (100, 200, 300)
   - **Ordinal Numbers**: (1, 2, 3)
   - **Alphabetical**: (A, B, C or a, b, c)

7. **Preview the Tokens**
   The wizard will show your size tokens preview:

```
Name: Size
┌──────────┬──────────┐
│ Scale    │ Value    │
├──────────┼──────────┤
│ 01       │ 16px     │
│ 02       │ 24px     │
│ 03       │ 32px     │
│ 04       │ 40px     │
└──────────┴──────────┘
```

8. **Generate Your Artifacts**
   Once confirmed, the wizard will:

   - Export your tokens in Tokens Studio JSON format Stored in: `output_files/tokens/size/size_tokens_{unit}.json`
   - Create CSS and SCSS files with your tokens as variables Stored in `output_files/tokens/css/size/size_variables_{unit}.css` and `output_files/tokens/scss/size/size_variables_{unit}.scss`

9. **Finalize the Spell**
   Review the output files and integrate your size tokens into your system.

⭕ Border Radius Tokens Wizard

Version 1.7.0

Creating border-radius tokens is simplified with the Border Radius Tokens Wizard 🧙 . This wizard guides you through defining and generating border-radius tokens for your design system, ready to be used in various formats including JSON, CSS, and SCSS.

1. **Invoke the Spell**
   Begin your journey by running the Border Radius Tokens Wizard script:

   ```
   npm run radii
   ```

2. **Define the Token Name**
   Choose a name for your border-radius tokens (e.g., `border-radius`, `corner-radius`, `radius`).

3. **Select Scale Structure**
   Decide whether to include 'none' and 'full' values for the border radius.

4. **Define Intermediate Steps**
   Optionally, add intermediate steps for a more granular scale:

   - **T-shirt Sizes:** (xs, sm, md, lg, xl)
   - **Incremental Numbers:** (100, 200, 300, 400)
   - **Ordinal Numbers:** (01, 02, 03, 04 or 1, 2, 3, 4)
   - **Alphabetical:** (A, B, C, D or a, b, c, d)
   - **Semantic:** (subtle, soft, moderate, bold)

5. **Define Value Scale**
   Select a value scale based on either minimal or expressive bases:

   - **4-Point Grid System**

- **8-Point Grid System**
- **Modular Scale** (multiplier based)
- **Custom Intervals**
- **Fibonacci Scale**

6. **Preview the Tokens**

   The wizard will show your border-radius tokens preview:

```
Name: Border Radius
┌──────────┬──────────┐
│ Scale    │ Value    │
├──────────┼──────────┤
│ None     │ 0px      │
│ 01       │ 4px      │
│ 02       │ 8px      │
│ 03       │ 12px     │
│ Full     │ 9999px   │
└──────────┴──────────┘
```

7. **Generate Your Artifacts**

   Once confirmed, the wizard will:

   - Export your tokens in Tokens Studio JSON format Stored in: `output_files/tokens/border-radius/border_radius_tokens_{unit}.json`
   - Create CSS and SCSS files with your tokens as variables Stored in `output_files/tokens/css/border-radius/border_radius_variables_{unit}.css` and `output_files/tokens/scss/border-radius/border_radius_variables_{unit}.scss`

8. **Finalize the Spell**

   Review the output files and integrate your border-radius tokens into your system.

🧹 Clear Spell

Version 1.2.1

The Clear Spell is a simple yet powerful tool to keep your project directory clean and organized. It removes all generated files, ensuring you can start fresh without any clutter.

1. **Run the Clear Spell Script**
   Open your terminal and execute the following command:

   ```
   npm run clear
   ```

2. **What Happens Next**

   - All generated token files in the `output_files/tokens/` directory are deleted.
   - CSS and SCSS files in the `output_files/tokens/css/` and `output_files/tokens/scss/` directories are removed.
   - Final output files in the `output_files/final/` directory, including JSON, CSS, and SCSS files, will also be removed.
   - Your project directory is left clean, free from outdated or unnecessary files.

🪄 Merge Spell

Version 1.2.0

The Merge Spell is designed to simplify your workflow by combining multiple token files into a single, unified file. This makes managing and integrating design tokens across categories (e.g., color, size, spacing) much easier.

1. **Run the Merge Spell Script**
   Open your terminal and execute the following command:

```
npm run merge
```

2. **What Happens Next**

   - All token files in the `output_files/final/` directory are merged into a single JSON, CSS and SCSS file.
   - The resulting file is structured consistently, ensuring seamless integration into your design system or development workflow.
   - You can select which formats and options from your generated tokens you want to include in the final merged file.

Created with Love in Barcelona by Fulvia Buonanno 🪄❤️

Discover more about our wizards at: Design Tokens Wizards Website

If you're passionate about design systems and tokens, this tool is your perfect companion, allowing you to create tokens effortlessly. For RPG or JRPG fans, this tool will evoke a sense of nostalgia, blending classic gaming vibes with your design workflow. 🧩

Created with love by Fulvia Buonanno, a Design Systems Designer based in Barcelona, this tool aims to bridge the gap between design and development, making tokens more accessible, especially for newcomers to this magical world. 🧙‍♀️

📦 Dependencies

Below is a comprehensive list of all dependencies used in this project:

Dependency	Version	Description	Repository
chalk	^5.4.1	Terminal string styling done right	chalk/chalk
cli-table3	^0.6.5	Pretty unicode tables for the command line	cli-table3
inquirer	^12.4.2	A collection of common interactive command line user interfaces	SBoudrias/Inquirer.js
path	^0.12.7	Node.js path module	nodejs/node
tinycolor2	^1.6.0	Fast, small color manipulation and conversion	bgrins/TinyColor
puppeteer	^20.0.0	Headless Chrome Node.js API for automating web interactions	puppeteer/puppeteer

📝 License

This project is licensed under the MIT License. This means you are free to use, modify, and distribute the software as long as the original copyright notice and permission notice are included in all copies or substantial portions of the software.

For more details, you can read the full license text in the LICENSE file included in this repository or visit the Open Source Initiative for more information.

❓ Troubleshooting and FAQ

Q: How can I provide feedback or report issues?
A: We welcome your feedback! You can:

- Reach out to us on our website
- Fill out this form

Your feedback helps us improve the tool and make it better for everyone. We're particularly interested in:

- Bug reports
- Feature requests
- Documentation improvements
- User experience feedback
- Performance issues

Q: I get a permission error or "command not found"?
A: Make sure you have Node.js (v18+) installed and are running commands from the project root.

Q: Where are my generated files?
A: Check the `output_files/` directory.

Q: How do I reset/clean all generated files?
A: Run `npm run clear` to remove all generated output.

Q: Can I use these tokens with my design tool?
A: Yes! The tokens are exported in multiple formats (JSON, CSS, SCSS) that can be used with most design tools and development environments.

Q: How do I update the tokens after making changes?
A: Simply run the wizard again with your new values. The files will be updated automatically.

Q: Can I customize the naming convention for my tokens?
A: Yes! Each wizard allows you to choose from different naming conventions (T-shirt sizes, incremental numbers, ordinal numbers, etc.).

Q: What's the difference between the Merge Spell and Clear Spell?
A: The Merge Spell combines all your token files into a single unified file, while the Clear Spell removes all generated files to start fresh.

Q: How do I contribute to the project?
A: Check out our Contributing section for guidelines. We welcome all contributions!

Q: Can I use these tokens in my commercial project?
A: Yes! This project is licensed under MIT, which means you can use it freely in any project, including commercial ones.

Q: What color formats are supported?
A: The Color Tokens Wizard supports HEX, RGB, RGBA, and HSL formats. You can choose your preferred format during the generation process.

Q: Can I use custom fonts in the Typography Wizard?
A: Yes! You can specify any font family, including custom fonts. Just make sure to include proper fallbacks for better cross-platform compatibility.

Q: What units are supported for spacing and sizing?
A: The Space and Size Wizards support px, rem, and em units. You can choose your preferred unit during the generation process.

Q: How do I maintain consistency across different projects?
A: Use the Merge Spell to combine tokens from different projects, and consider creating a token library for shared components.

Q: What's the best way to organize my token files?
A: We recommend organizing tokens by category (color, typography, spacing, etc.) and using the Merge Spell to combine them when needed.

Q: Can I automate token generation in my CI/CD pipeline?
A: Yes! The wizards can be run from the command line, making them perfect for

automation in your development workflow.

Q: Can I use these tokens with my CSS framework?
A: Yes! The tokens are exported in standard formats (CSS, SCSS) that can be used with any CSS framework or vanilla CSS.

Q: Something else isn't working!
A: Please open an issue or contact me.

📫 Contact and Support

For questions, suggestions, or bug reports, please open an issue or contact me.

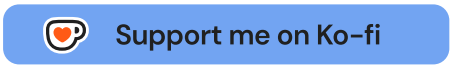
🤝 Contributing

Contributions, issues, and feature requests are welcome!
Feel free to check the issues page or submit a pull request.

Please see CONTRIBUTING.md for guidelines.

☕ Support the Project

If you find this tool helpful and want to show your appreciation, consider buying me a coffee! Your support helps me maintain and improve the Design Tokens Wizards, making it even more magical for everyone.



Every coffee helps me:

- Add new features and improvements
- Fix bugs and maintain the codebase
- Create more documentation and examples
- Keep the magic alive! ✨

Even a small contribution makes a big difference in keeping this project thriving.
Thank you for being part of our magical community! 🧙