Docs > registry-item.json

registry-item.json

Specification for registry items.

The `registry-item.json` schema is used to define your custom registry items.

registry-item.json

```
{
  "$schema": "https://ui.shadcn.com/schema/registry-item.json",
  "name": "hello-world",
  "type": "registry:block",
  "title": "Hello World",
  "description": "A simple hello world component.",
  "files": [
    {
      "path": "registry/hello-world/hello-world.tsx",
      "type": "registry:component"
    },
    {
      "path": "registry/hello-world/use-hello-world.ts",
      "type": "registry:hook"
    }
  ]
}
```

Definitions

You can see the JSON Schema for `registry-item.json` here.

$schema

The `$schema` property is used to specify the schema for the `registry-item.json` file.

registry-item.json

```
{
  "$schema": "https://ui.shadcn.com/schema/registry-item.json"
}
```

name

The `name` property is used to specify the name of your registry item.

registry-item.json

```
{
  "name": "hello-world"
}
```

title

A human-readable title for your registry item. Keep it short and descriptive.

Search... ⌘ K

```
{
  "title": "Hello World"
}
```

description

A description of your registry item. This can be longer and more detailed than the `title`.

registry-item.json

```
{
  "description": "A simple hello world component."
```

```
}
```

type

The `type` property is used to specify the type of your registry item.

registry-item.json

```
{
  "type": "registry:block"
}
```

The following types are supported:

Type	Description
`registry:block`	Use for complex components with multiple files.
`registry:component`	Use for simple components.
`registry:lib`	Use for lib and utils.
`registry:hook`	Use for hooks.
`registry:ui`	Use for UI components and single-file primitives
`registry:page`	Use for page or file-based routes.
`registry:file`	Use for miscellaneous files.

author

The `author` property is used to specify the author of the registry item.

It can be unique to the registry item or the same as the author of the registry.

registry-item.json

```
{
  "author": "John Doe <john@doe.com>"
```

```
}
```

dependencies

The `dependencies` property is used to specify the dependencies of your registry item. This is for `npm` packages.

Use `@version` to specify the version of your registry item.

registry-item.json

```
{
  "dependencies": [
    "@radix-ui/react-accordion",
    "zod",
    "lucide-react",
    "name@1.0.2"
  ]
}
```

registryDependencies

Used for registry dependencies. Can be names or URLs.

- For `shadcn/ui` registry items such as `button`, `input`, `select`, etc use the name eg. `['button', 'input', 'select']`.
- For custom registry items use the URL of the registry item eg. `['https://example.com/r/hello-world.json']`.

registry-item.json

```
{
  "registryDependencies": [
    "button",
    "input",
    "select",
```

```
      "https://example.com/r/editor.json"
    ]
  }
```

Note: The CLI will automatically resolve remote registry dependencies.

files

The `files` property is used to specify the files of your registry item. Each file has a `path`, `type` and `target` (optional) property.

The `target` property is required for `registry:page` and `registry:file` types.

registry-item.json

```
{
  "files": [
    {
      "path": "registry/hello-world/page.tsx",
      "type": "registry:page",
      "target": "app/hello/page.tsx"
    },
    {
      "path": "registry/hello-world/hello-world.tsx",
      "type": "registry:component"
    },
    {
      "path": "registry/hello-world/use-hello-world.ts",
      "type": "registry:hook"
    },
    {
      "path": "registry/hello-world/.env",
      "type": "registry:file",
      "target": "~/.env"
    }
  ]
}
```

path

The `path` property is used to specify the path to the file in your registry. This path is used by the build script to parse, transform and build the registry JSON payload.

type

The `type` property is used to specify the type of the file. See the **type** section for more information.

target

The `target` property is used to indicate where the file should be placed in a project. This is optional and only required for `registry:page` and `registry:file` types.

By default, the `shadcn` cli will read a project's `components.json` file to determine the target path. For some files, such as routes or config you can specify the target path manually.

Use `~` to refer to the root of the project e.g `~/foo.config.js`.

tailwind

The `tailwind` property is used for tailwind configuration such as `theme`, `plugins` and `content`.

You can use the `tailwind.config` property to add colors, animations and plugins to your registry item.

registry-item.json

```
{
  "tailwind": {
    "config": {
      "theme": {
        "extend": {
```

```
            "colors": {
              "brand": "hsl(var(--brand))"
            },
            "keyframes": {
              "wiggle": {
                "0%, 100%": { "transform": "rotate(-3deg)" },
                "50%": { "transform": "rotate(3deg)" }
              }
            },
            "animation": {
              "wiggle": "wiggle 1s ease-in-out infinite"
            }
          }
        }
      }
    }
```

cssVars

Use to define CSS variables for your registry item.

registry-item.json

```
{
  "cssVars": {
    "light": {
      "brand": "20 14.3% 4.1%",
      "radius": "0.5rem"
    },
    "dark": {
      "brand": "20 14.3% 4.1%"
    }
  }
}
```

Note: When adding colors, make sure to also add them to the `tailwind.config.theme.extend.colors` property.

docs

Use `docs` to show custom documentation or message when installing your registry item via the CLI.

registry-item.json

```
{
  "docs": "Remember to add the FOO_BAR environment variable to your .env
}
```

categories

Use `categories` to organize your registry item.

registry-item.json

```
{
  "categories": ["sidebar", "dashboard"]
}
```

meta

Use `meta` to add additional metadata to your registry item. You can add any key/value pair that you want to be available to the registry item.

registry-item.json

```
{
  "meta": { "foo": "bar" }
}
```

registry.json

Built by shadcn. The source code is available on GitHub.