Now Design System - File Processing Flow

Complete System Flow Overview

TOKEN SOURCE (JSON) → STYLE DICTIONARY → CUSTOM FORMATTER → FINAL OUTPUTS

1. TOKEN SOURCE (JSON FILES)

```
packages/tokens/
```

- **color/brand.json** - Brand colors and values
- **scale/scale.json** - Scale tokens (spacing, sizing)
- **scale/responsive-spacing.json** - Responsive spacing tokens
- **typography/font-families.json** - Font family definitions
- **typography/font-weights.json** - Font weight values

```
{ "brand": { "blue": { "600": { "value": "#1579BE", "type":
"color" } } } }
```

▼

2. STYLE DICTIONARY CONFIGURATION

```
style-dictionary.config.js
```

This file tells Style Dictionary:

- Where to find token files: `packages/tokens/**/*.json`
- How to process them: `transforms: ['attribute/cti', 'name/cti/kebab']`
- Where to output: `buildPath: 'packages/tokens/dist/'`
- What formats to generate: CSS, SCSS, JS

▼

3. BUILD SCRIPT EXECUTION

```
build-tokens.js
```

The execution script that:

- Imports Style Dictionary
- Registers custom formatter
- Runs the build process
- Generates all outputs

▼

4. STYLE DICTIONARY PROCESSING

Internal processing steps:

1. **SCAN:** Reads all JSON files from packages/tokens/**/*.json
2. **PARSE:** Converts JSON to Style Dictionary objects
3. **RESOLVE:** Processes references like {scale.300.value} → 12
4. **TRANSFORM:** Applies name transforms (brandBlue600 → brand-blue-600)
5. **GROUP:** Organizes tokens by platform (css, scss, js)

▼

5. CUSTOM FORMATTER PROCESSING

```
style-dictionary/formats/responsive-css.js
```

Custom formatter that:

- Processes each token
- Handles responsive tokens (desktop/tablet/mobile)
- Generates CSS with media queries
- Creates semantic CSS variables

▼

6. FINAL OUTPUTS

`packages/tokens/dist/`

- **css/variables.css** - Generated CSS variables with media queries
- **scss/_variables.scss** - Generated SCSS variables
- **js/tokens.js** - Generated JavaScript objects

```
:root { --brand-blue-600: #1579BE; --gap-spacing-300-desktop: 12; --gap-
spacing-300-tablet: 6; --gap-spacing-300-mobile: 4; } @media (max-width:
1024px) { :root { --gap-spacing-300: 6; } }
```

▼

7. COMPONENT USAGE

`packages/atoms/src/Button/Button.css`

Components use the generated CSS variables:

```
.simple-token-button { padding: var(--gap-spacing-300-desktop, 12px)
var(--gap-spacing-600-desktop, 24px); background: var(--brand-blue-600,
#1579BE); color: var(--brand-neutral-100, #fff); font-family: var(--font-
family-body, 'Oxanium'); }
```

▼

8. CONSUMER APPLICATION

Consumer apps import and use components:

```
import { Button } from '@now-design/atoms'; import '@now-design/tokens/
dist/css/variables.css'; function MyApp() { return ( <Button>Click me</
Button> ); }
```

Key Insights

- **SINGLE SOURCE OF TRUTH:** All design values start in JSON files
- **RESPONSIVE BY DESIGN:** Responsive tokens have desktop/tablet/mobile values
- **AUTOMATED WORKFLOW:** Run `npm run build-tokens` to regenerate all outputs
- **PLATFORM AGNOSTIC:** Same tokens can generate CSS, SCSS, JS, Android, iOS, etc.

Developer Workflow

1. Edit JSON tokens in packages/tokens/
2. Run: `npm run build-tokens`
3. Generated files appear in packages/tokens/dist/
4. Components automatically use new token values
5. Test in Storybook: `npm run storybook`

File Dependencies Flow

JSON Tokens → Style Dictionary Config → Build Script → Custom Formatter → Generated Outputs → Components → Consumer Apps

Each step depends on the previous one, creating a reliable and predictable build process.