Token Migration Playbook

Step-by-step guide for migrating to semantic tokens
By Petri Lahdelma - 2026 Edition

Why Migrate to Semantic Tokens?

- Enable dark mode and theming without touching component code
- Reduce inconsistency from raw color/spacing values
- Make design intent explicit in code
- Simplify brand changes and white-labeling

Migration Steps

Step 1: Audit Current Token Usage

- Run static analysis to find all color/spacing values
- grep for hex codes: #[0-9a-fA-F]{3,6}
- grep for rgb/rgba: rgb\(|rgba\(
- grep for px values: [0-9]+px
- Export to spreadsheet with file:line references

Step 2: Create Token Mapping

- Map each raw value to a semantic token
- `#3b82f6 -> color.primary.default`
- `#ef4444 -> color.danger.default`
- `16px -> spacing.4 or spacing.component.gap`
- Document edge cases and exceptions

Step 3: Set Up Enforcement

- Add ESLint rule to block core token imports
- Configure Stylelint to flag raw values
- Add pre-commit hooks for token validation
- Create CI check for token compliance

Step 4: Run Migration Codemods

- Write jscodeshift transforms for JS/TS
- Write postcss transforms for CSS
- Run in dry-run mode first
- Review diffs before committing
- Migrate one surface/feature at a time

Step 5: Visual Regression Testing

- Capture before/after screenshots
- Run Chromatic or Percy visual diffs
- Manual QA on high-traffic surfaces
- Sign off from design before merge

ESLint Rule Example

```
// .eslintrc.js
module.exports = {
  rules: {
    'no-restricted-imports': ['error', {
      patterns: [{
        group: ['**/tokens/core/*'],
        message: 'Use semantic tokens.'
      }]
    }]
  }
};
```