Styling a Code Review Platform with Stylus

When building a modern web application, styling is more than just making things look good—it's about creating an intuitive, responsive, and engaging user experience. I recently explored the Stylus-based stylesheet of a Vue.js-powered code review platform, and its CSS architecture is a treasure trove of techniques worth unpacking. Let's dive into how this app uses Stylus to craft its polished UI, from layout structuring to hover effects, all while keeping the code maintainable and scalable.

Why Stylus? A Quick Primer

Stylus is a CSS preprocessor that strips away the verbosity of traditional CSS (no curly braces or semicolons required) and adds powerful features like variables, mixins, and nesting. The provided code imports variables from `variables.styl` and a base stylesheet from `base.styl`, setting the stage for consistent and reusable styles. For example, the primary color `#1CB2EF` is likely defined in `variables.styl` and reused across buttons and backgrounds.

Structuring the Layout: Sections and Containers

The app's homepage is divided into distinct sections—`.slide`, `.feature`, `.reviewer`, `.example`, and `.contact`—each with its own styling strategy. Here's how the `.slide` (hero) section is styled:

```
.slide
  height 800px
  position relative
  color #fff
  width 100%
  overflow hidden
  .bg
    background url("../img/home/hero.jpg") no-repeat
    background-size cover
    background-position-y 40%
    position 200% 200%
    width 100%
    height 100%
    padding-top 280px
```

Key Techniques:

- **Full-Screen Hero**: The `height 800px` and `width 100%` create a bold, full-width banner. `overflow hidden` ensures no content spills out.

- **Background Image**: The `.bg` class uses `background-size cover` to scale the hero image proportionally, while `background-position-y 40%` fine-tunes its vertical alignment for visual impact.
- **Nesting**: Stylus's nesting keeps related styles grouped, improving readability compared to flat CSS.

Responsive Grids with Flexbox and clearfix

The `.feature` section showcases a three-column layout:

```
.feature
  height 450px
  padding 125px 0
  background white
  .list
    width 1160px
    margin 0 auto
    display flex
    flex-direction row
    li
      height 200px
      padding-left 50px
      flex-grow 1
      &:first-child
        padding-left 0
      .short
        width 235px
        height 200px
        margin 0 auto
```

Highlights:

- **Flexbox**: `display flex` and `flex-direction row` align the list items horizontally, while `flex-grow 1` ensures they expand evenly to fill the container.
- **Centering**: `width 1160px` paired with `margin 0 auto` centers the content, a classic technique for fixed-width layouts.
- **Pseudo-Class Magic**: The `&:first-child` selector removes padding from the first item, preventing awkward spacing.

The `.example` section takes this further with a grid of review cards, using the `clearfix()` mixin:

```
.example
  .list
```

```
    clearfix()
    .row
      clearfix()
      li:first-child
        margin-left 0
    li
      height 354px
      margin-left 48px
      pull-left()
      margin-bottom 48px
```

- **Clearfix**: This mixin (likely defined in `base.styl`) handles float clearing, ensuring rows stack properly in older browsers or custom layouts.
- **Float-Based Grid**: `pull-left()` (another utility mixin) floats items left, with `margin-left 48px` adding gutters. This approach complements Flexbox for broader compatibility.

Interactive Styling: Hover Effects and Transitions

The review cards in `.example` shine with smooth hover interactions:

```
li
  .info
    position relative
    height 354px
    width 100%
    color white
    box-shadow 0 4px 4px 1px rgba(135,135,135,.1)
    overflow hidden
    cursor pointer
    &:hover
      img
        transform scale(1.2,1.2)
        -webkit-filter brightness(0.6)
      .title
        -webkit-transform translate(0, -20px)
        opacity 1.0
      .tips
        -webkit-transform translate(0, -10px)
        opacity 0.8
    img
```

```
    height 100%
    -webkit-filter brightness(0.4)
    transition all 0.35s ease 0s
```

Breakdown:

- **Hover Effects**: On hover, the image scales up (`transform scale(1.2,1.2)`) and brightens (`-webkit-filter brightness(0.6)`), while text elements shift upward with `translate` and adjust opacity.
- **Transitions**: The `transition all 0.35s ease 0s` ensures smooth animations for all properties, with a 350ms duration and easing curve.
- **Layering**: `position absolute` on `.text` positions it over the image, with `z-index 2` ensuring visibility.

The `.author` button also reacts:

```
.author
  position absolute
  background black
  margin-left 30px
  margin-top 30px
  height 30px
  padding-left 20px
  padding-right 20px
  transition all 0.35s ease 0s
  &:hover
    background #1cb2ef
```

A simple color swap from black to the brand color `#1CB2EF` on hover adds a delightful touch.

Visual Polish: Shadows, Buttons, and Icons

Shadows enhance depth, as seen in `.info`'s `box-shadow 0 4px 4px 1px rgba(135,135,135,.1)`. Buttons, like in `.contact`, are styled with care:

```
.contact
  .rightbtn
    .more
      width 127px
      height 50px
      color #1CB2EF
      background white
```

```
    border-radius 3px
    border 1px solid #00A3E6
    -webkit-box-shadow 0px 1px 0px rgba(255,255,255,0.15) inset, 0px 1px 2px rgba(0,0,0,0.15)
```

- **Inset Shadow**: The subtle inner shadow (`inset`) paired with an outer drop shadow creates a pressed-button effect.
- **Consistency**: The border color `#00A3E6` ties into the brand palette.

Icons, like `.icon_crown`, use background images:

```
.icon_crown
  background url("../img/icon/crown@2x.png") no-repeat
  background-size contain
  width 49px
  height 52px
```

The `@2x` suffix suggests retina-ready assets, with `background-size contain` ensuring proper scaling.

Best Practices and Takeaways

This Stylus implementation offers lessons for any CSS project: 1. **Use Preprocessors**: Stylus's nesting and mixins (e.g., `clearfix()`) streamline complex layouts. 2. **Balance Layouts**: Combine Flexbox for modern browsers with float-based fallbacks for robustness. 3. **Enhance UX**: Smooth transitions and hover effects make the UI feel alive. 4. **Keep It Maintainable**: Leverage variables and imports for consistency across large codebases.

Whether you're styling a code review platform or a personal portfolio, these techniques can elevate your CSS game. Next time you write a stylesheet, consider how nesting, transitions, and a touch of shadow can transform your design!