एक व्यू.जे.एस. के साथ प्रभावी कोड रिव्यू प्लेटफॉर्म बनाना

आज के तेजी से चल रहे विकास दुनिया में, कोड की गुणवत्ता परम है। एक अच्छी तरह से संरचित कोड रिव्यू प्रक्रिया एक टीम की उत्पादन को बढ़ा सकती है और व्यक्तिगत कौशल को तेज कर सकती है। हाल ही में, मैंने एक रोचक प्रोजेक्ट का पता लगाया—एक कोड रिव्यू सेवा जो □□□.□□ के साथ बनाई गई है जो डेवलपर्स को विशेषज्ञ रिव्यूअर्स से जोड़ती है ताकि वे अपने कोडबेस को सुधार सकें। हम इस प्लेटफॉर्म के तकनीकी आधार पर गहरे उतरते हैं, इसके फ्रंट-एंड आर्किटेक्चर, कम्पोनेंट डिजाइन और स्टाइलिंग तकनीकों पर ध्यान केंद्रित करते हैं।

बड़ा चित्र: □□□.□□ के रूप में आधार

प्लेटफॉर्म □□□.□□, एक प्रोग्रेसिव जावास्क्रिप्ट फ्रेमवर्क का उपयोग करता है, एक इंटरैक्टिव और मॉड्यूलर यूजर इंटरफ़ेस बनाने के लिए। मैंने जांचा कोडबेस एक सिंगल-पेज एप्लिकेशन (□□□) है, जिसमें एक साफ़ सफाई के साथ □□□□ टेम्पलेट्स के लिए संरचना, जावास्क्रिप्ट के लिए लॉजिक और स्टाइलस के लिए स्टाइलिंग है। यह त्रिकूट एक आधुनिक वेब डेवलपमेंट के लिए एक अच्छा केस स्टडी बनाता है।

इसके कोर में, ऐप में एक होमपेज है जिसमें सेक्शन जैसे कि एक हीरो बैनर, फीचर हाइलाइट्स, रिव्यूअर शोकेस और उदाहरण रिव्यू शामिल हैं। प्रत्येक सेक्शन को सोच-समझकर डिजाइन किया गया है ताकि उपयोगकर्ताओं को सेवा की वैल्यू प्रोपोजिशन के माध्यम से मार्गदर्शन किया जा सके, विशेषज्ञ रिव्यूअर्स को खोजने से लेकर वास्तविक दुनिया के कोड रिव्यू केसों को खोजने तक।

टेम्पलेट को खंडित करना: कम्पोनेंट्स और डायनामिक रेंडरिंग

□□□□ टेम्पलेट एक मिश्रण है स्थिर सामग्री और डायनामिक □□□ कम्पोनेंट्स। यहाँ एक हीरो सेक्शन का एक स्निपेट है:

```
<section class="slide">
  <div class="bg">
    <h1>              </h1>
    <h2>     ,                          </h2>
    <a href="./belief.html"><button class="help">2016,                    </button></a>
  </div>
</section>
```

यह सेक्शन सीधा है लेकिन एक बॉल्ड बैकग्राउंड इमेज और एक कॉल-टू-एक्शन (□□□) के साथ टोन सेट करता है। हालाँकि, वास्तविक जादू डायनामिक सेक्शन में होता है, जैसे कि “उदाहरण कोड रिव्यू”:

```
<section class="example">
  <div class="container">
    <h2>          </h2>
    <ul class="list">
      <div class="row">
        <li class="clo-1" @click="goDetail(reviews[0].reviewId)">
          <div class="info">
            <button class="author" v-for="author in reviews[0].authors">{{author.authorName}}</button>
            <img :src="reviews[0].coverUrl">
```

```
            <div class="text">
              <h6 class="title" v-html="reviews[0].title"></h6>
              <h6 class="tips">
                <span v-for="tag in reviews[0].tags">#{{tag.tagName}}</span>
              </h6>
            </div>
          </div>
        </li>
        <!--          -->
      </div>
    </ul>
  </div>
</section>
```

मुख्य विशेषताएँ:

1. **डायनामिक डेटा बाइंडिंग**: `:src` और `v-html` डायरेक्टिव्स `reviews` एर्रे (स्क्रिप्ट में परिभाषित) से डेटा को टेम्पलेट में बाइंड करते हैं। यह ऐप को फेटच या हार्डकोडेड डेटा के आधार पर सामग्री को डायनामिक रूप से रेंडर करने की अनुमति देता है।
2. **इवेंट हैंडलिंग**: `@click="goDetail(reviews[0].reviewId)"` डायरेक्टिव एक विधि को ट्रिगर करता है जो एक डिटेल्ड व्यू में रिव्यू के लिए नाविगेट करता है, □□□ के सीमलेस इवेंट सिस्टम का प्रदर्शन करता है।
3. **लूप्स के साथ** `v-for`: `v-for` डायरेक्टिव `authors` और `tags` जैसे एर्रे पर इटरेट करता है, कई तत्वों को दक्षता से रेंडर करता है। यह कई योगदानकर्ताओं या मेटाडेटा को प्रदर्शित करने के लिए हार्डकोडिंग के बिना आदर्श है।

`reviews` डेटा स्क्रिप्ट में परिभाषित है:

```
reviews: [
  {
    reviewId: 1,
    coverUrl: 'http://7xotd0.com1.z0.glb.clouddn.com/photo-1450849608880-6f787542c88a.jpeg',
    title: '    <br>  <br>          ',
    tags: [{tagName: 'XCode'}, {tagName: 'iOS'}],
    authors: [{authorName: '    '}]
  },
  //
]
```

इस एर्रे को आसानी से एक □□□ कॉल से बदल दिया जा सकता है, जिससे ऐप वास्तविक दुनिया के उपयोग के लिए स्केल करने में सक्षम हो जाता है।

कम्पोनेंट आर्किटेक्चर: पुन: उपयोग्यता और मॉड्यूलरिटी

ऐप □□□ कम्पोनेंट्स का भारी उपयोग करता है, जो स्क्रिप्ट के शीर्ष पर आयात किए जाते हैं:

```
import reviewerCard from '../components/reviewer-card.vue';
import Guide from '../components/guide.vue';
import Overlay from '../components/overlay.vue';
import Contactus from '../components/contactus.vue';
```

ये कम्पोनेंट्स टेम्पलेट में रजिस्टर किए जाते हैं और उपयोग किए जाते हैं, जैसे कि `<reviewer :reviewers="reviewers"></reviewer>` और `<guide></guide>`। यह मॉड्यूलर दृष्टिकोण: - **पुनरावृत्ति को कम करता है**: सामान्य □□ तत्व (जैसे कि रिव्यूअर कार्ड) पेजों के माध्यम से पुन: उपयोग किए जाते हैं। - **बढ़ा देता है**: प्रत्येक कम्पोनेंट अपने स्वयं के लॉजिक और स्टाइल्स को एनकैप्सुलेट करता है।

उदाहरण के लिए, `Overlay` कम्पोनेंट डायनामिक सामग्री को लपेटता है:

```
<overlay :overlay.sync="overlayStatus">
  <component :is="currentView"></component>
</overlay>
```

यहाँ, `:overlay.sync` ओवरले के दृश्यता को `overlayStatus` डेटा प्रॉपर्टी के साथ सिंक करता है, जबकि `:is` `currentView` कम्पोनेंट (जैसे कि `Contactus`) को डायनामिक रूप से रेंडर करता है। यह एक शक्तिशाली तरीका है मोडल या पॉपअप्स को संभालने के लिए मुख्य टेम्पलेट को क्लटर करने के बिना।

डेटा फेटचिंग: □□□□ रिक्वेस्ट्स और इनिशियलाइजेशन

`created` लाइफसाइकल हुक पेज को डेटा फेटच करके इनिशियलाइज करता है:

```
created() {
  this.$http.get(serviceUrl.reviewers, { page: "home" }).then((resp) => {
    if (util.filterError(this, resp)) {
      this.reviewers = resp.data.result;
    }
  }, util.httpErrorFn(this));
  this.$http.get(serviceUrl.reviewsGet, { limit: 6 }).then((resp) => {
    if (util.filterError(this, resp)) {
      var reviews = resp.data.result;
      //
    }
  }, util.httpErrorFn(this));
  this.checkSessionToken();
}
```

- **एसिंक्रोनस डेटा लोडिंग**: ऐप □□□ के `$http` (शायद □□□ □□□□□□□□ या □□□□□) का उपयोग करता है रिव्यूअर और रिव्यू डेटा को एक बैकएंड □□□ से फेटच करने के लिए।
- **एरर हैंडलिंग**: `util.filterError` यूटिलिटी □□ को स्थिर रखने के लिए एक मजबूत एरर प्रबंधन सुनिश्चित करता है।
- **सेशन प्रबंधन**: `checkSessionToken` विधि उपयोगकर्ता प्रमाणन को क्वेरी पैरामीटर के माध्यम से संभालती है, कुकीज सेट करती है और आवश्यकता के अनुसार रिडायरेक्ट करती है।

स्टाइलिंग के साथ स्टाइलस: प्रतिक्रिया और शानदार

स्टाइलिंग, □□□□□□ में लिखी गई है, लचीलापन और सौंदर्य को संयोजित करती है। `.example` सेक्शन को देखें:

```
.example
  margin 0 auto
  padding-top 5px
  background #FDFFFF
  .list
    clearfix()
    .row
      clearfix()
      li:first-child
        margin-left 0
    li
      height 354px
      margin-left 48px
      pull-left()
      margin-bottom 48px
      .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
```

हाइलाइट्स:

- □ **हॉवर प्रभाव**: `&:hover` पसंदीदा-तत्व इमेज को स्केल करता है और पाठ को शिफ्ट करता है, एक सुलभ, इंटरैक्टिव अनुभव बनाने के लिए।
- □ **लचीलापन**: `clearfix()` मिक्सिन और `pull-left()` यूटिलिटी एक प्रतिक्रियाशील ग्रिड लेआउट सुनिश्चित करते हैं।
- □ **दृश्य चमक**: छायाएँ और ट्रांजिशंस (जैसे कि `transition: all 0.35s ease 0s`) गहराई और फ्लुइडिटी जोड़ते हैं।

`variables.styl` से चर (जैसे कि रंग `#1CB2EF`) का उपयोग करने से ऐप के माध्यम से एकसंगता सुनिश्चित होती है।

अगले प्रोजेक्ट के लिए लेक्चर

यह कोड रिव्यू प्लेटफॉर्म मूल्यवान पाठ प्रदान करता है: 1. □□□ **के प्रतिक्रियाशीलता का उपयोग करें**: डेटा को डायनामिक रूप से बाइंड करें और कम्पोनेंट्स का उपयोग करें ताकि आपका ऐप मॉड्यूलर रहे। 2. **स्केलेबिलिटी के लिए योजना बनाएं**: हार्डकोडेड डेटा को □□□ कॉल से बदलें जैसे कि आपका ऐप बढ़ता है। 3. **स्मार्ट स्टाइल करें**: □□□□□□ जैसे प्रीप्रोसेसर का उपयोग करें, जो बनाए रखने योग्य, पुन: उपयोग करने योग्य स्टाइल्स के लिए हैं। 4. □□ **पर ध्यान केंद्रित करें**: सुलभ ट्रांजिशंस और स्पष्ट □□□□ उपयोगकर्ता एंगेजमेंट को बढ़ाते हैं।

चाहे आप एक कोड रिव्यू टूल बनाएं या एक अलग वेब ऐप, ये सिद्धांत आपकी डेवलपमेंट प्रक्रिया को सुलभ बना सकते हैं और आपके उपयोगकर्ताओं को खुशी दे सकते हैं। आपका अगला प्रोजेक्ट क्या है? हम कोड गुणवत्ता के वार्तालाप को आगे बढ़ाते हैं!