Voici d'autres questions pratiques et de résolution de problèmes pour tester votre réactivité, votre approche méthodologique et votre capacité à gérer des situations réelles dans le cadre d'un poste d'intégrateur web avec SASS :

### **Questions pratiques et résolution de problèmes**

1. **Vous êtes en train de travailler sur un projet avec plusieurs collaborateurs, comment vous assurez-vous de la cohérence et de l'uniformité du style à travers tout le projet ?**
   * **Réponse attendue :** Pour garantir la cohérence, je mettrais en place des **normes de codage** partagées, comme une convention de nommage uniforme (par exemple, BEM), l'utilisation de variables communes pour les couleurs et les polices, et je m'assurerais que tout le monde suit la même structure de projet. J'utiliserais également un outil de linter (comme **stylelint**) pour appliquer des règles de style.
2. **Comment réagiriez-vous si un client demande des ajustements rapides sur un site déjà mis en ligne ?**
   * **Réponse attendue :** Dans ce cas, je m'assurerais de comprendre les changements exacts demandés, puis je les appliquerais sur un environnement de développement local ou de staging. Une fois testés et validés, je ferais les ajustements nécessaires dans les fichiers SASS, puis je recompilerais les styles et les déploierais sur l'environnement de production en respectant le processus de mise en production.

### **Questions pratiques et résolution de problèmes (suite)**

### **1. Problème de CSS non appliqué correctement sur un site en production**

**Question :** Lors de la mise en production d'un site, vous remarquez que certaines règles CSS ne sont pas appliquées correctement, même après avoir compilé les fichiers SASS. Comment allez-vous diagnostiquer et résoudre ce problème ?

**Réponse attendue :**

1. Vérifier la compilation des fichiers SASS : Je commencerais par m'assurer que les fichiers SASS sont bien compilés sans erreur. Si un problème est détecté durant la compilation, il pourrait empêcher l'application des règles CSS.
2. Vérification des fichiers liés : Il est essentiel de vérifier que les fichiers CSS générés par la compilation sont bien liés à la page HTML (via un <link> dans le <head>). Il peut y avoir un problème de cache dans le navigateur ou des liens incorrects.
3. Inspection via les outils de développement : J'utiliserais les **DevTools** de Chrome ou Firefox pour inspecter les éléments sur la page. Cela permet de vérifier si les styles sont bien chargés, s'il y a des conflits de spécificité ou des règles qui sont ignorées.
4. Vérification des priorités de sélecteurs : Un problème de spécificité dans les sélecteurs CSS peut entraîner l'ignorance des règles. Par exemple, des styles définis plus bas dans le fichier CSS ou des styles en ligne (inline styles) peuvent écraser les règles précédentes.
5. Vérifier les erreurs dans la console : La console des outils de développement pourrait donner des indices sur des erreurs liées à des variables non définies ou à des erreurs de syntaxe dans les fichiers SASS.

### **2. Gestion des conflits de styles entre différentes sections d'un site**

**Question :** Comment gérez-vous les conflits de styles entre différentes sections d'un site, notamment si deux sections utilisent des classes similaires mais des styles différents ?

**Réponse attendue :**

1. **Utiliser la méthodologie BEM :** En appliquant BEM correctement, on s'assure que chaque composant a un nom de classe unique et spécifique (par exemple, header\_\_title et footer\_\_title), ce qui permet d'éviter les conflits.
2. **Scope des styles :** Si des sections similaires partagent des classes avec des styles différents, je pourrais créer des classes spécifiques à chaque contexte en ajoutant des préfixes ou des modificateurs (par exemple, section\_\_title--dark et section\_\_title--light).
3. **CSS Modules (si applicable) :** Dans le cas où je travaille avec des frameworks modernes (React, Vue, etc.), j’utiliserais **CSS Modules** pour encapsuler les styles au niveau des composants, ce qui réduit les risques de conflits.
4. **SASS Partiels :** En utilisant des fichiers SASS modularisés (partiels), chaque composant ou section du site a ses propres styles définis dans un fichier séparé. Cela rend plus facile la gestion et la mise à jour des styles sans provoquer de conflits.

### **3. Problème de performance dû à des styles CSS inutiles**

**Question :** Le site commence à être lent à charger en raison de styles CSS non utilisés qui sont inclus dans le fichier final. Comment traiteriez-vous ce problème ?

**Réponse attendue :**

1. **Utilisation d'outils comme PurifyCSS ou UnCSS :** Ces outils permettent de supprimer les classes et les styles inutilisés dans le fichier CSS final. J'utiliserais ces outils pendant le processus de build pour m'assurer que seul le CSS nécessaire est inclus.
2. **Réorganiser le code SASS :** En SASS, je structurerais les fichiers pour qu'ils soient modulaires, en n'important que les fichiers nécessaires pour chaque page. Par exemple, un composant qui n'est utilisé que dans une page spécifique ne doit pas être inclus globalement.
3. **Lazy loading des styles :** Si le site est complexe et que certaines parties ne sont visibles que sur interaction (comme des onglets ou des sections d'un formulaire), je pourrais utiliser le **lazy loading** pour charger les styles de ces composants uniquement quand ils sont nécessaires.
4. **Code splitting avec Webpack :** Si le site utilise **Webpack**, je mettrais en place le **code splitting** pour séparer les styles en différents fichiers et les charger uniquement lorsque cela est nécessaire.

### **4. Résolution d'un bug sur la compatibilité mobile**

**Question :** Vous constatez qu'une section du site, qui fonctionne bien sur bureau, est cassée sur mobile. Quelle est votre démarche pour résoudre ce problème ?

**Réponse attendue :**

1. **Vérifier la responsivité des styles :** Je commencerais par inspecter les styles qui s'appliquent à la section à l’aide des **DevTools**. Il est possible que des media queries manquent ou qu’un élément ait une largeur fixe ou un padding trop grand pour les écrans plus petits.
2. **Tester les styles via les media queries :** J’ajusterais les règles CSS dans les media queries pour m'assurer que les éléments sont redimensionnés correctement pour les appareils mobiles. Par exemple, en réduisant la taille des polices ou en ajustant les marges/paddings.
3. **Vérifier les propriétés de mise en page (flexbox, grid) :** Si la section utilise **Flexbox** ou **CSS Grid**, je vérifierais les propriétés comme flex-wrap, justify-content, align-items ou grid-template-columns pour m'assurer qu'elles s'adaptent correctement aux petites résolutions.
4. **Tester dans plusieurs tailles d'écran :** Je testerais la page sur différents appareils et tailles d'écran pour m'assurer que la mise en page se comporte correctement dans toutes les résolutions cibles.
5. **Valider avec l'équipe de conception :** Si nécessaire, je vérifierais les maquettes avec l'équipe de design pour m'assurer qu'il n'y a pas d'attentes contradictoires en matière de mise en page mobile.

### **5. Problème d’alignement ou de centrage d’un élément**

**Question :** Vous avez un élément (par exemple, un bouton) qui ne se centre pas correctement dans un conteneur. Comment résolvez-vous ce problème ?

**Réponse attendue :**

1. **Vérification du modèle de boîte :** Je commencerais par m'assurer que le **box model** (marges, paddings, bordures) de l'élément est correctement défini. Un élément peut ne pas se centrer si ses marges sont incorrectes.

**Utilisation de Flexbox ou Grid :** Le **Flexbox** est une solution simple et moderne pour centrer un élément horizontalement et verticalement. Par exemple :  
 .container {

display: flex;

justify-content: center;

align-items: center;

}

Si le conteneur est une grille, je pourrais utiliser CSS Grid pour centrer l'élément avec :  
 .container {

display: grid;

place-items: center;

}

1. **Vérification des propriétés de positionnement :** Si j'utilise des techniques comme position: absolute, je m'assurerais que le parent a une position relative pour que l'élément soit correctement centré par rapport à son parent.
2. **Vérifier la taille de l'élément :** Si l'élément a une largeur ou hauteur fixe, cela peut interférer avec le centrage. Je m'assurerais que l'élément est flexible (par exemple, en utilisant width: auto ou height: auto) pour qu'il s'adapte correctement.

### **6. Mise à jour de styles en production sans interruption du service**

**Question :** Vous devez déployer une mise à jour des styles CSS en production sans interrompre l'expérience utilisateur. Comment vous y prenez-vous ?

**Réponse attendue :**

1. **Utiliser un environnement de staging :** Avant de déployer en production, je testerais les mises à jour dans un environnement de staging pour m'assurer que tout fonctionne correctement.
2. **Mise en cache des fichiers CSS :** Je m'assurerais que les fichiers CSS sont bien versionnés, en ajoutant un hash unique dans le nom du fichier (par exemple, style.abc123.css). Cela permet de garantir que le navigateur recharge les nouveaux styles sans utiliser les anciens en cache.
3. **Déploiement progressif (Canary releases) :** Pour minimiser les risques, j’utiliserais un déploiement progressif, où je déploie d'abord la mise à jour sur une petite portion d'utilisateurs (canary release) avant de la propager à tous.
4. **Monitoring après déploiement :** Après la mise à jour, je surveillerais les performances et les rapports d’erreurs pour m'assurer qu'aucune régression n'est survenue.

Ces questions pratiques visent à évaluer non seulement vos connaissances techniques, mais aussi votre approche méthodique pour résoudre des problèmes courants dans un environnement de production. Elles montrent également votre capacité à travailler sous pression et à maintenir des normes de qualité élevées.