Ingénieur Frontend Entretien

Débuter avec HTML :

1. Balises sémantiques : Comprendre et utiliser `<article>`, `<section>`, `<header>`, `<footer>`, `<nav>`.
2. Formulaires : Mettre en œuvre la validation, gérer `<input>`, `<textarea>`, `<select>`, `<button>`.
3. Tables : Créer des tables accessibles avec `<table>`, `<thead>`, `<tbody>`, `<tfoot>`.
4. Métadonnées : Utiliser les balises `<meta>` pour le charset, la vueport et le SEO.
5. Liens et ancres : Comprendre les balises `<a>`, les attributs `href`, `target` et `download`.
6. Éléments multimédias : Utiliser correctement `<img>`, `<video>`, `<audio>` avec des attributs comme `src`, `alt`, `controls`.
7. Listes : Créer des listes ordonnées `<ol>` et non ordonnées `<ul>`, y compris les listes imbriquées.
8. En-têtes : Utiliser une hiérarchie d'en-têtes correcte `<h1>` à `<h6>`.
9. Intégration de contenu : Utiliser `<iframe>`, `<embed>`, et `<object>` pour intégrer du contenu externe.
10. API HTML5 : Familiarité avec la géolocalisation, le stockage web et l'API Fetch.

Maintenant, CSS :

11. Modèle de boîte : Comprendre la marge, le remplissage, la bordure et leur impact sur la mise en page.
12. Flexbox : Maîtriser l'alignement, l'enroulement et l'ordre avec les propriétés Flexbox.
13. Mise en page en grille : Créer des mises en page complexes à l'aide de CSS Grid.
14. Conception réactive : Utiliser les requêtes média, la balise méta viewport et les images réactives.
15. Préprocesseurs CSS : Connaissance de la syntaxe et des fonctionnalités de Sass, Less ou Stylus.
16. CSS-in-JS : Comprendre les frameworks comme styled-components ou emotion.
17. Animations et transitions : Mettre en œuvre des transitions fluides et des animations de clés.
18. Stylisation des formulaires : Personnaliser les éléments de formulaire et améliorer leur apparence.
19. Réinitialisation et normalisation CSS : Savoir quand et pourquoi les utiliser.
20. CSS Grid vs Flexbox : Comprendre les différences et choisir le bon outil pour le travail.

JavaScript :

21. Fonctionnalités ES6+ : Utiliser les fonctions fléchées, la déstructuration, les opérateurs spread/rest et les littéraux de modèle.

22. Manipulation du DOM : Sélectionner des éléments, modifier le DOM et gérer les événements.
23. JavaScript asynchrone : Comprendre les Promesses, async/await et l'API Fetch.
24. Boucle d'événements : Expliquer comment fonctionne la boucle d'événements en JavaScript.
25. Fermetures : Comprendre et utiliser les fermetures efficacement.
26. Héritage prototypal : Expliquer comment fonctionne l'héritage prototypal en JavaScript.
27. Modules : Utiliser les modules ES6 avec `import` et `export`.
28. Gestion des erreurs : Utiliser les blocs try/catch et comprendre les rejets de promesses non gérées.
29. Performance JavaScript : Optimiser le code pour de meilleures performances.
30. Console du navigateur : Utiliser les outils de développement du navigateur pour le débogage.

Frameworks :

31. React.js : Comprendre les composants, JSX, l'état, les props et les hooks.
32. Vue.js : Comprendre l'instance Vue, les directives, les composants et la réactivité.
33. Angular : Comprendre les composants, les services, l'injection de dépendances et le routage.
34. Gestion de l'état : Utiliser Redux, Vuex ou l'API Context pour la gestion de l'état.
35. Routage : Mettre en œuvre le routage côté client avec React Router, Vue Router, etc.
36. Architecture basée sur les composants : Comprendre et mettre en œuvre des composants réutilisables.
37. Méthodes de cycle de vie : Connaître les méthodes de cycle de vie de React ou les hooks de Vue.
38. Bibliothèques d'interface utilisateur : Utiliser des bibliothèques comme Bootstrap, Tailwind ou Material-UI.
39. Frameworks de test : Écrire des tests avec Jest, Jasmine ou Cypress.
40. Outils de construction : Utiliser Webpack, Babel ou Parcel pour construire des projets.

Outils et contrôle de version :

41. Git : Utiliser git pour le contrôle de version, y compris le branchement, la fusion et le rebasage.
42. npm/yarn : Gérer les dépendances et les scripts de projet.
43. package.json : Comprendre les scripts, les dépendances et les devDependencies.
44. Gestionnaires de tâches : Utiliser Gulp ou Grunt pour automatiser les tâches.
45. Linting : Utiliser ESLint ou Prettier pour la qualité du code.

46. Browsersync : Utiliser pour le rechargement en direct pendant le développement.
47. Figma/Adobe XD : Comprendre la remise de conception et collaborer avec les designers.
48. Intégration d'API : Récupérer des données à partir d'API RESTful ou GraphQL.
49. Variables d'environnement : Gérer les configurations spécifiques à l'environnement.
50. Intégration continue : Mettre en place des pipelines CI/CD avec GitHub Actions ou Jenkins.

Optimisation des performances :

51. Fractionnement du code : Mettre en œuvre le fractionnement du code avec Webpack ou les imports dynamiques.
52. Chargement différé : Charger paresseusement les images, les composants et les scripts.
53. Minification : Minifier les fichiers CSS, JavaScript et HTML.
54. Stratégies de mise en cache : Utiliser les en-têtes de mise en cache HTTP et les workers de service.
55. Optimisation des images : Compresser et optimiser les images pour une utilisation web.
56. CSS critique : Intégrer le CSS critique pour des chargements de page plus rapides.
57. Métriques de performance web : Comprendre Lighthouse, GTmetrix et PageSpeed Insights.
58. Chargement des polices : Optimiser le chargement des polices avec WebFont Loader ou l'hébergement propre.
59. Éviter les ressources bloquant le rendu : S'assurer que les scripts et les styles ne bloquent pas le rendu.
60. Budgets de performance : Définir et respecter les budgets de performance.

Accessibilité :

61. Rôles ARIA : Utiliser les rôles, états et propriétés ARIA pour une meilleure accessibilité.
62. HTML sémantique : Choisir des éléments sémantiques pour améliorer l'accessibilité.
63. Texte alternatif pour les images : Fournir un texte alternatif significatif pour les images.
64. Navigation au clavier : S'assurer que le site est navigable uniquement avec le clavier.
65. Contraste des couleurs : Utiliser des outils pour vérifier et améliorer le contraste des couleurs.
66. Test avec les lecteurs d'écran : Tester avec des lecteurs d'écran comme NVDA ou VoiceOver.
67. Gestion de la mise au point : Assurer une gestion correcte de la mise au point sur les éléments interactifs.
68. Directives d'accessibilité : Suivre les directives WCAG 2.1.

69. Accessibilité des formulaires : Utiliser correctement les étiquettes, les espaces réservés et la validation.
70. EPub et conformité AODA : Comprendre les normes de conformité de base.

Bonnes pratiques :

71. Organisation du code : Maintenir des structures de code propres et modulaires.
72. Documentation : Rédiger une documentation claire pour les composants et les API.
73. Tests inter-navigateurs : Tester sur plusieurs navigateurs et appareils.
74. Amélioration progressive : Construire des sites qui fonctionnent pour tous les utilisateurs, indépendamment du support du navigateur.
75. Sécurité : Prévenir les attaques XSS, utiliser la politique de sécurité de contenu et sécuriser les API.
76. Meilleures pratiques SEO : Optimiser pour les moteurs de recherche avec les balises méta, les en-têtes et le texte alternatif.
77. Versioning : Utiliser le versioning sémantique pour les bibliothèques et les dépendances.
78. Outils de collaboration : Utiliser GitHub, GitLab ou Bitbucket pour la collaboration d'équipe.
79. Revues de code : Participer aux revues de code et fournir des retours constructifs.
80. Ressources d'apprentissage : Rester à jour avec MDN, les blogs et les cours en ligne.

Sujets avancés :

81. WebSockets : Mettre en œuvre une communication en temps réel avec WebSockets.
82. PWA (Progressive Web Apps) : Comprendre les workers de service, le support hors ligne et les notifications push.
83. Canvas et SVG : Créer des graphiques avec les éléments Canvas et SVG.
84. Mises en page CSS Grid et Flexbox : Mettre en œuvre des mises en page complexes avec CSS Grid et Flexbox.
85. Éléments personnalisés : Créer des éléments HTML personnalisés avec les Web Components.
86. Shadow DOM : Comprendre et utiliser le Shadow DOM pour l'encapsulation.
87. Variables CSS : Utiliser des propriétés personnalisées pour le thème et les styles dynamiques.
88. Modèles de conception JavaScript : Mettre en œuvre des modèles de conception comme Singleton, Observer et Factory.
89. Internationalisation (i18n) : Mettre en œuvre le support de la langue et la localisation.

90. Profilage des performances : Utiliser des outils comme Chrome DevTools pour profiler les performances JavaScript et DOM.

Compétences interdisciplinaires :

91. Expérience utilisateur (UX) : Comprendre les principes UX et collaborer avec les designers UX.
92. Interface utilisateur (UI) : Créer des interfaces visuellement attrayantes et conviviales.
93. Gestion de projet : Utiliser les méthodologies Agile, Scrum ou Kanban pour la gestion de projet.
94. Compétences en communication : Communiquer efficacement avec les membres de l'équipe et les parties prenantes.
95. Résolution de problèmes : Aborder les problèmes de manière méthodique et trouver des solutions optimales.
96. Adaptabilité : Apprendre et s'adapter rapidement aux nouvelles technologies et outils.
97. Collaboration d'équipe : Travailler bien en équipe, partager les connaissances et mentoriser les autres.
98. Gestion du temps : Prioriser les tâches et gérer le temps efficacement.
99. Créativité : Apporter des solutions créatives aux défis de conception et de codage.
100. Passion pour l'apprentissage : Rester curieux et améliorer continuellement vos compétences.