# フロントエンドエンジニア面接

HTML から始める：

1. 意味論的タグ：<article>、<section>、<header>、<footer>、<nav> を理解し、使用する。
2. フォーム：検証を実装し、<input>、<textarea>、<select>、<button> を処理する。
3. テーブル：<table>、<thead>、<tbody>、<tfoot> を使用してアクセス可能なテーブルを作成する。
4. メタデータ：charset、viewport、SEO のための <meta> タグを使用する。
5. リンクとアンカー：<a> タグ、href、target、download 属性を理解する。
6. メディア要素：src、alt、controls などの属性を使用して <img>、<video>、<audio> を正しく使用する。
7. リスト：順序付き <ol> と順序なし <ul> リストを作成し、入れ子になったリストを含める。
8. 見出し：適切な見出し階層 <h1> から <h6> を使用する。
9. コンテンツの埋め込み：外部コンテンツを埋め込むために <iframe>、<embed>、<object> を使用する。
10. HTML5 API：地理情報、Web ストレージ、Fetch API に親しむ。

次に、CSS：

11. ボックスモデル：マージン、パディング、ボーダーを理解し、レイアウトに与える影響を理解する。
12. フレックスボックス：フレックスボックスのプロパティを使用して配置、ラッピング、順序をマスターする。
13. グリッドレイアウト：CSS グリッドを使用して複雑なレイアウトを作成する。
14. レスポンシブデザイン：メディアクエリ、ビューポートメタタグ、レスポンシブ画像を使用する。
15. CSS プリプロセッサ：Sass、Less、Stylus の構文と機能を知る。
16. CSS-in-JS：styled-components や emotion などのフレームワークを理解する。
17. アニメーションとトランジション：スムーズなトランジションとキーフレームアニメーションを実装する。
18. フォームのスタイル：フォーム要素をカスタマイズし、見た目を向上させる。
19. CSS リセットと正規化：使用するタイミングと理由を知る。
20. CSS グリッド対フレックスボックス：違いを理解し、適切なツールを選ぶ。

JavaScript：

21. ES6+ 機能：アロー関数、分割代入、スプレッド/レスト演算子、テンプレートリテラルを使用する。

22. DOM 操作：要素を選択し、DOM を変更し、イベントを処理する。
23. 非同期 JavaScript：Promise、async/await、fetch API を理解する。
24. イベントループ：JavaScript のイベントループの動作を説明する。
25. クロージャー：効果的に使用するためにクロージャーを理解する。
26. プロトタイプ継承：JavaScript でのプロトタイプ継承の動作を説明する。
27. モジュール：`import` と `export` を使用して ES6 モジュールを使用する。
28. エラーハンドリング：try/catch ブロックを使用し、処理されていない Promise 拒否を理解する。
29. JavaScript パフォーマンス：パフォーマンスを向上させるためにコードを最適化する。
30. ブラウザコンソール：デバッグのためにブラウザ開発者ツールを使用する。

フレームワーク：

31. React.js：コンポーネント、JSX、状態、プロップ、フックを理解する。
32. Vue.js：Vue インスタンス、ディレクティブ、コンポーネント、リアクティブを理解する。
33. Angular：コンポーネント、サービス、依存性注入、ルーティングを理解する。
34. 状態管理：Redux、Vuex、Context API を使用して状態管理を行う。
35. ルーティング：React Router、Vue Router などを使用してクライアントサイドルーティングを実装する。
36. コンポーネントベースのアーキテクチャ：再利用可能なコンポーネントを理解し、実装する。
37. ライフサイクルメソッド：React のライフサイクルメソッドや Vue フックを知る。
38. UI ライブラリ：Bootstrap、Tailwind、Material-UI などのライブラリを使用する。
39. テストフレームワーク：Jest、Jasmine、Cypress を使用してテストを書く。
40. ビルドツール：Webpack、Babel、Parcel を使用してプロジェクトをビルドする。

ツールとバージョン管理：

41. Git：バージョン管理のために Git を使用し、ブランチ、マージ、リベースを行う。
42. npm/yarn：プロジェクトの依存関係とスクリプトを管理する。
43. package.json：スクリプト、依存関係、devDependencies を理解する。
44. タスクランナー：Gulp や Grunt を使用してタスクを自動化する。
45. リンティング：ESLint や Prettier を使用してコードの品質を保つ。
46. Browsersync：開発中にライブリロードを行うために使用する。

47. Figma/Adobe XD：デザインの引き継ぎを理解し、デザイナーと協力する。
48. API 統合：RESTful または GraphQL API からデータを取得する。
49. 環境変数：環境固有の設定を管理する。
50. 継続的インテグレーション：GitHub Actions または Jenkins を使用して CI/CD パイプラインを設定する。

パフォーマンス最適化：

51. コード分割：Webpack または動的インポートを使用してコード分割を実装する。
52. 遅延読み込み：画像、コンポーネント、スクリプトを遅延読み込みする。
53. ミニファイ：CSS、JavaScript、HTML ファイルをミニファイする。
54. キャッシュ戦略：HTTP キャッシュヘッダーとサービスワーカーを使用する。
55. 画像最適化：ウェブ用に画像を圧縮し、最適化する。
56. クリティカル CSS：ページの読み込みを速くするためにクリティカル CSS をインラインにする。
57. ウェブパフォーマンスメトリクス：Lighthouse、GTmetrix、PageSpeed Insights を理解する。
58. フォントの読み込み：WebFont Loader または自ホスティングを使用してフォントの読み込みを最適化する。
59. レンダリングブロックリソースの回避：スクリプトとスタイルがレンダリングをブロックしないようにする。
60. パフォーマンス予算：パフォーマンス予算を設定し、遵守する。

アクセシビリティ：

61. ARIA ロール：より良いアクセシビリティのために ARIA ロール、状態、プロパティを使用する。
62. 意味論的 HTML：アクセシビリティを向上させるために意味論的な要素を選択する。
63. 画像の代替テキスト：画像に意味のある代替テキストを提供する。
64. キーボードナビゲーション：キーボードのみでサイトがナビゲーション可能であることを確認する。
65. 色のコントラスト：ツールを使用して色のコントラストを確認し、向上させる。
66. スクリーンリーダーテスト：NVDA や VoiceOver などのスクリーンリーダーでテストする。
67. フォーカス管理：インタラクティブ要素の適切なフォーカス管理を確認する。
68. アクセシビリティガイドライン：WCAG 2.1 ガイドラインに従う。
69. フォームのアクセシビリティ：ラベル、プレースホルダー、検証を正しく使用する。
70. EPub と AODA 準拠：基本的な準拠基準を理解する。

ベストプラクティス：

71. コードの整理：クリーンでモジュール化されたコード構造を保つ。
72. ドキュメント：コンポーネントと API の明確なドキュメントを書く。
73. クロスブラウザテスト：複数のブラウザとデバイスでテストする。
74. プログレッシブエンハンスメント：ブラウザのサポートに関係なくすべてのユーザーに対応するサイトを構築する。
75. セキュリティ：XSS 攻撃を防ぎ、コンテンツセキュリティポリシーを使用し、API をセキュリティ保護する。
76. SEO ベストプラクティス：メタタグ、見出し、代替テキストを使用して検索エンジンに最適化する。
77. バージョニング：ライブラリと依存関係にセマンティックバージョニングを使用する。
78. コラボレーションツール：GitHub、GitLab、Bitbucket を使用してチームで協力する。
79. コードレビュー：コードレビューに参加し、建設的なフィードバックを提供する。
80. 学習リソース：MDN、ブログ、オンラインコースで最新情報を入手する。

高度なトピック：

81. WebSockets：WebSockets を使用してリアルタイム通信を実装する。
82. PWA（プログレッシブウェブアプリ）：サービスワーカー、オフラインサポート、プッシュ通知を理解する。
83. Canvas と SVG：Canvas と SVG 要素を使用してグラフィックスを作成する。
84. CSS グリッドとフレックスボックスレイアウト：CSS グリッドとフレックスボックスを使用して複雑なレイアウトを実装する。
85. カスタム要素：Web コンポーネントを使用してカスタム HTML 要素を作成する。
86. シャドウ DOM：カプセル化のためにシャドウ DOM を理解し、使用する。
87. CSS 変数：テーマと動的スタイルのためにカスタムプロパティを使用する。
88. JavaScript デザインパターン：Singleton、Observer、Factory などのデザインパターンを実装する。
89. 国際化（i18n）：言語サポートとローカライゼーションを実装する。
90. パフォーマンスプロファイリング：Chrome DevTools を使用して JavaScript と DOM パフォーマンスをプロファイリングする。

跨学際スキル：

91. ユーザーエクスペリエンス（UX）：UX 原則を理解し、UX デザイナーと協力する。
92. ユーザーインターフェース（UI）：視覚的に魅力的でユーザーフレンドリーなインターフェースを作成する。
93. プロジェクト管理：アジルメソッド、Scrum、Kanban を使用してプロジェクト管理を行う。
94. コミュニケーションスキル：チームメンバーや関係者と効果的にコミュニケーションを取る。
95. 問題解決：問題を体系的にアプローチし、最適な解決策を見つける。
96. 適応力：新しい技術やツールを迅速に学び、適応する。
97. チームコラボレーション：チームで良く働き、知識を共有し、他の人を指導する。
98. タイムマネジメント：タスクを優先順位付けし、時間を効果的に管理する。
99. クリエイティブ：デザインとコーディングの課題に対してクリエイティブな解決策を提供する。
100. 学びへの情熱：好奇心を持ち続け、スキルを継続的に向上させる。