前端工程师面试

HTML (20 点)

1. 语义化标签：使用 `<article>`, `<section>`, `<header>`, `<footer>`, `<nav>`。
2. 表单：实现验证，处理 `<input>`, `<textarea>`, `<select>`, `<button>`。
3. 表格：创建可访问的表格，使用 `<table>`, `<thead>`, `<tbody>`, `<tfoot>`。
4. 元数据：使用 `<meta>` 标签设置字符集、视口和 SEO。
5. 链接和锚点：理解 `<a>` 标签的 `href`, `target`, 和 `download` 属性。
6. 媒体元素：正确使用 `<img>`, `<video>`, `<audio>`，包括 `src`, `alt`, `controls` 属性。
7. 列表：创建有序 `<ol>` 和无序 `<ul>` 列表，包括嵌套列表。
8. 标题：使用适当的标题层次结构 `<h1>` 到 `<h6>`。
9. 嵌入内容：使用 `<iframe>`, `<embed>`, 和 `<object>` 嵌入外部内容。
10. HTML5 API：了解 Geolocation, Web Storage, 和 Fetch API。

CSS (20 点)

11. 盒模型：理解 margin, padding, border，以及它们如何影响布局。
12. Flexbox：掌握对齐、换行和排序的 Flexbox 属性。
13. Grid 布局：使用 CSS Grid 创建复杂布局。
14. 响应式设计：使用媒体查询、视口元标签和响应式图片。
15. CSS 预处理器：了解 Sass, Less, 或 Stylus 的语法和特性。
16. CSS-in-JS：了解框架如 styled-components 或 emotion。
17. 动画和过渡：实现平滑过渡和关键帧动画。
18. 表单样式：自定义表单元素，提升其外观。
19. CSS Reset 和 Normalize：知道何时以及为什么使用它们。
20. CSS Grid vs Flexbox：了解差异，并选择合适的工具。

JavaScript (20 点)

21. ES6+ 特性：使用箭头函数、解构、扩展/剩余运算符和模板字面量。
22. DOM 操作：选择元素，修改 DOM，处理事件。
23. 异步 JavaScript：了解 Promises, async/await, 和 fetch API。
24. 事件循环：解释 JavaScript 中事件循环的工作原理。
25. 闭包：理解和有效使用闭包。
26. 原型继承：解释 JavaScript 中原型继承的工作原理。
27. 模块：使用 ES6 模块进行 `import` 和 `export`。
28. 错误处理：使用 try/catch 块，理解未处理的 Promise 拒绝。
29. JavaScript 性能：优化代码以提高性能。

30. 浏览器控制台：使用浏览器开发者工具进行调试。

框架 (10 点)

31. React.js：理解组件、JSX、状态、属性和钩子。
32. Vue.js：理解 Vue 实例、指令、组件和反应性。
33. Angular：理解组件、服务、依赖注入和路由。
34. 状态管理：使用 Redux, Vuex, 或 Context API 进行状态管理。
35. 路由：使用 React Router, Vue Router 等实现客户端路由。
36. 组件化架构：理解和实现可重用组件。
37. 生命周期方法：了解 React 生命周期方法或 Vue 钩子。
38. UI 库：使用 Bootstrap, Tailwind, 或 Material-UI 等库。
39. 测试框架：使用 Jest, Jasmine, 或 Cypress 编写测试。
40. 构建工具：使用 Webpack, Babel, 或 Parcel 构建项目。

工具和版本控制 (10 点)

41. Git：使用 Git 进行版本控制，包括分支、合并和变基。
42. npm/yarn：管理项目依赖和脚本。
43. package.json：了解脚本、依赖和开发依赖。
44. 任务运行器：使用 Gulp 或 Grunt 自动化任务。
45. 代码检查：使用 ESLint 或 Prettier 确保代码质量。
46. Browsersync：开发期间使用实时重新加载。
47. Figma/Adobe XD：理解设计交接，与设计师协作。
48. API 集成：从 RESTful 或 GraphQL API 获取数据。
49. 环境变量：管理环境特定配置。
50. 持续集成：使用 GitHub Actions 或 Jenkins 设置 CI/CD 管道。

性能优化 (10 点)

51. 代码分割：使用 Webpack 或动态导入实现代码分割。
52. 延迟加载：延迟加载图片、组件和脚本。
53. 最小化：最小化 CSS、JavaScript 和 HTML 文件。
54. 缓存策略：使用 HTTP 缓存头和服务工作者。
55. 图像优化：压缩和优化图像以适应 web。
56. 关键 CSS：内联关键 CSS 以加快页面加载。
57. Web 性能指标：了解 Lighthouse, GTmetrix, 和 PageSpeed Insights。
58. 字体加载：优化字体加载，使用 WebFont Loader 或自托管。
59. 避免渲染阻塞资源：确保脚本和样式不会阻止渲染。

60. 性能预算：设定并遵守性能预算。

可访问性 (10 点)

61. ARIA 角色：使用 ARIA 角色、状态和属性提升可访问性。
62. 语义化 HTML：选择语义化元素提升可访问性。
63. 图像替代文本：为图像提供有意义的替代文本。
64. 键盘导航：确保网站仅使用键盘即可导航。
65. 颜色对比：使用工具检查和提高颜色对比度。
66. 屏幕阅读器测试：使用 NVDA 或 VoiceOver 进行测试。
67. 焦点管理：确保交互元素的焦点管理正确。
68. 可访问性指南：遵循 WCAG 2.1 指南。
69. 表单可访问性：正确使用标签、占位符和验证。
70. EPub 和 AODA 合规性：了解基本合规标准。

最佳实践 (10 点)

71. 代码组织：保持清洁和模块化的代码结构。
72. 文档：为组件和 API 编写清晰的文档。
73. 跨浏览器测试：在多个浏览器和设备上进行测试。
74. 渐进增强：构建适用于所有用户的网站，无论浏览器支持如何。
75. 安全性：防止 XSS 攻击，使用内容安全策略，保护 API。
76. SEO 最佳实践：使用元标签、标题和替代文本优化搜索引擎。
77. 版本控制：使用语义化版本控制管理库和依赖。
78. 协作工具：使用 GitHub, GitLab, 或 Bitbucket 进行团队协作。
79. 代码审查：参与代码审查，提供建设性反馈。
80. 学习资源：通过 MDN、博客和在线课程保持更新。

高级主题 (10 点)

81. WebSockets：使用 WebSockets 实现实时通信。
82. PWA (渐进式 Web 应用)：了解服务工作者、离线支持和推送通知。
83. Canvas 和 SVG：使用 Canvas 和 SVG 元素创建图形。
84. CSS Grid 和 Flexbox 布局：使用 CSS Grid 和 Flexbox 实现复杂布局。
85. 自定义元素：使用 Web Components 创建自定义 HTML 元素。
86. Shadow DOM：理解和使用 Shadow DOM 进行封装。
87. CSS 变量：使用自定义属性进行主题和动态样式。
88. JavaScript 设计模式：实现设计模式如单例、观察者和工厂。
89. 国际化 (i18n)：实现语言支持和本地化。

90. 性能分析：使用 Chrome DevTools 分析 JavaScript 和 DOM 性能。

跨学科技能 (10 点)

91. 用户体验 (UX)：理解 UX 原则，与 UX 设计师协作。
92. 用户界面 (UI)：创建视觉上吸引人且用户友好的界面。
93. 项目管理：使用 Agile 方法、Scrum 或 Kanban 进行项目管理。
94. 沟通技巧：与团队成员和利益相关者有效沟通。
95. 问题解决：系统地解决问题，找到最优解。
96. 适应能力：快速学习和适应新技术和工具。
97. 团队协作：在团队中工作良好，分享知识，指导他人。
98. 时间管理：优先处理任务，有效管理时间。
99. 创造力：为设计和编码挑战带来创意解决方案。
100. 学习热情：保持好奇心，不断改进技能。