

前端工程師面試

從 HTML 開始：

1. 语义标签：了解并使用 `<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>` 标签进行字符集、视口和 SEO。
5. 链接和锚点：了解 `<a>` 标签、`href`、`target` 和 `download` 属性。
6. 多媒体元素：正确使用 ``、`<video>`、`<audio>` 及其属性如 `src`、`alt`、`controls`。
7. 列表：创建有序 `` 和无序 `` 列表，包括嵌套列表。
8. 标题：使用适当的标题层次 `<h1>` 到 `<h6>`。
9. 内嵌内容：使用 `<iframe>`、`<embed>` 和 `<object>` 进行外部内容的嵌入。
10. HTML5 API：熟悉地理位置、Web 存储和 Fetch API。

接下来是 CSS：

11. 盒模型：了解边距、填充、边框及其对布局的影响。
12. Flexbox：掌握 Flexbox 属性的对齐、包装和排序。
13. 网格布局：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像。
15. CSS 预处理器：了解 Sass、Less 或 Stylus 的语法和功能。
16. CSS-in-JS：了解 styled-components 或 emotion 等框架。
17. 动画和过渡：实现平滑过渡和关键帧动画。
18. 样式表单：自定义表单元素并改善其外观。
19. CSS 重置和标准化：了解何时以及为何使用它们。
20. CSS 网格与 Flexbox：了解它们的区别并选择合适的工具。

JavaScript：

21. ES6+ 特性：使用箭头函数、解构、展开/剩余运算符和模板字面量。
22. DOM 操作：选择元素、修改 DOM 并处理事件。

23. 异步 JavaScript：了解 Promise、async/await 和 fetch API。
24. 事件循环：解释 JavaScript 中的事件循环工作原理。
25. 闭包：理解并有效使用闭包。
26. 原型继承：解释 JavaScript 中的原型继承。
27. 模块：使用 ES6 模块的 import 和 export。
28. 错误处理：使用 try/catch 块并了解未处理的 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：

31. React.js：了解组件、JSX、状态、属性和钩子。
32. Vue.js：了解 Vue 实例、指令、组件和反应性。
33. Angular：了解组件、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：了解并实现可重用组件。
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 进行项目构建。

工具和版本控制：

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 进行代码质量。
46. Browsersync：用于开发期间的实时重新加载。
47. Figma/Adobe XD：了解设计交接并与设计师合作。

48. API 集成：从 RESTful 或 GraphQL API 获取数据。
49. 环境变量：管理环境特定的配置。
50. 持续集成：使用 GitHub Actions 或 Jenkins 设置 CI/CD 管道。

性能优化：

51. 代码拆分：使用 Webpack 或动态导入实现代码拆分。
52. 懒加载：懒加载图像、组件和脚本。
53. 缩减：缩减 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：压缩和优化图像以供网页使用。
56. 关键 CSS：内联关键 CSS 以加快页面加载。
57. 网页性能指标：了解 Lighthouse、GTmetrix 和 PageSpeed Insights。
58. 字体加载：使用 WebFont Loader 或自托管优化字体加载。
59. 避免阻塞渲染资源：确保脚本和样式不阻塞渲染。
60. 性能预算：设置并遵守性能预算。

无障碍：

61. ARIA 角色：使用 ARIA 角色、状态和属性以提高无障碍性。
62. 语义 HTML：选择语义元素以提高无障碍性。
63. 图像的替代文本：为图像提供有意义的替代文本。
64. 键盘导航：确保网站可以仅使用键盘导航。
65. 颜色对比：使用工具检查和改善颜色对比。
66. 屏幕阅读器测试：使用 NVDA 或 VoiceOver 等屏幕阅读器进行测试。
67. 焦点管理：确保交互元素的正确焦点管理。
68. 无障碍指南：遵循 WCAG 2.1 指南。
69. 表单无障碍：正确使用标签、占位符和验证。
70. EPub 和 AODA 合规：了解基本合规标准。

最佳实践：

71. 代码组织：保持清晰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. 多浏览器测试：在多个浏览器和设备上进行测试。
74. 逐步增强：构建适用于所有用户的网站，无论浏览器支持情况如何。
75. 安全：防止 XSS 攻击，使用内容安全策略并保护 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：使用语义版本控制进行库和依赖的版本控制。
78. 协作工具：使用 GitHub、GitLab 或 Bitbucket 进行团队协作。
79. 代码审查：参与代码审查并提供建设性反馈。
80. 学习资源：通过 MDN、博客和在线课程保持更新。

高级主题：

81. WebSockets：使用 WebSockets 实现实时通信。
82. PWA（渐进式网页应用）：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 网格和 Flexbox 布局：使用 CSS 网格和 Flexbox 实现复杂布局。
85. 自定义元素：使用 Web 组件创建自定义 HTML 元素。
86. Shadow DOM：了解并使用 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 Singleton、Observer 和 Factory 等设计模式。
89. 国际化（i18n）：实现语言支持和本地化。
90. 性能分析：使用 Chrome DevTools 等工具分析 JavaScript 和 DOM 性能。

跨学科技能：

91. 用户体验（UX）：了解 UX 原则并与 UX 设计师合作。
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：使用敏捷方法、Scrum 或 Kanban 进行项目管理。
94. 交流技能：有效地与团队成员和利益相关者交流。
95. 问题解决：系统地解决问题并找到最佳解决方案。
96. 适应能力：快速学习并适应新技术和工具。

97. 团队协作：在团队中良好工作，分享知识并指导他人。
98. 时间管理：优先处理任务并有效管理时间。
99. 创造力：为设计和编码挑战带来创造性解决方案。
100. 学习热情：保持好奇心并不断提升技能。