# 前端资料

## HTML

### 文档结构标签

<!DOCTYPE html> 声明 HTML5 文档类型 ✔️

<html> 根元素

<head> 文档头部（元信息）

<title> 页面标题

<body> 文档主体内容

<meta> 元数据（字符集、视口等）

<link> 引入外部资源（CSS、图标）

<style> 内联 CSS

<script> 引入或编写 JavaScript

<noscript> 脚本未支持时的备用内容

### 语义化布局标签

<header> 页眉或内容头部 ✔️

<footer> 页脚或内容尾部 ✔️

<nav> 导航栏 ✔️

<main> 文档主要内容 ✔️

<article> 独立内容（如博客文章） ✔️

<section> 内容区块 ✔️

<aside> 侧边栏或附加内容 ✔️

<figure> + <figcaption> 图文组合及标题 ✔️

<time> 时间/日期 ✔️

<mark> 高亮文本 ✔️

### 文本内容标签

<h1>-<h6> 标题

<p> 段落

<br> 换行

<hr> 水平分割线

<div> 通用容器

<span> 行内容器

<pre> 保留格式文本

<blockquote> 块级引用

<q> 行内引用

<cite> 引用来源

<code> 代码片段

<var> 变量

<kbd> 键盘输入

<samp> 程序输出示例

<sup> / <sub> 上标/下标

<small> 小号文本

<em> 强调（斜体）

<strong> 重要文本（加粗）

<i> 图标或斜体

<b> 无特殊意义的加粗

<u> 下划线文本

<s> 删除线文本

<del> / <ins> 删除/插入的文本

<wbr> 可选换行点

### 多媒体标签

<img> 图片

<audio> 音频播放器 ✔️

<video> 视频播放器 ✔️

<source> 多媒体资源（配合 <audio>/<video>） ✔️

<track> 字幕/音轨 ✔️

<canvas> 绘图画布 ✔️

<svg> 矢量图形

<picture> 响应式图片 ✔️

<map> + <area> 图片热区

### 表单相关标签

<form> 表单容器

<input> 输入框（类型扩展见下表）

<textarea> 多行文本输入

<button> 按钮

<select> + <option> 下拉选择框

<optgroup> 选项分组

<datalist> 输入框选项提示 ✔️

<label> 表单标签

<fieldset> + <legend> 表单分组及标题

<output> 计算结果输出 ✔️

<progress> 进度条 ✔️

<meter> 标量值（如磁盘用量）

### 表格标签

<table> 表格容器

<caption> 表格标题

<thead> / <tbody> / <tfoot> 表头/主体/页脚

<tr> 表格行

<th> 表头单元格

<td> 表格数据单元格

<colgroup> + <col> 列分组及样式控制

### 交互与动态内容标签

<details> + <summary> 折叠内容 ✔️

<dialog> 对话框 ✔️

<menu> 菜单（不常用）

<template> 内容模板（不可见）

### 其他标签

<iframe> 内嵌页面

<embed> 嵌入外部内容（如 Flash）

<object> 嵌入对象

<param> 为 <object> 传递参数

<base> 设置相对 URL 的基准路径

<bdi> 隔离双向文本（如阿拉伯语）

<ruby> + <rt> + <rp> 注音符号

## CSS

### 选择器

**(1) 基础选择器**

\*（通配符）

element（标签选择器，如 div）

.class（类选择器）

#id（ID 选择器）

selector1, selector2（分组选择器）

**(2) 组合选择器**

selector1 selector2（后代选择器）

selector1 > selector2（子选择器）

selector1 + selector2（相邻兄弟选择器）

selector1 ~ selector2（通用兄弟选择器）

**(3) 属性选择器（CSS3）**

[attr]（存在属性）

[attr="value"]（精确匹配）

[attr^="value"]（以 value 开头）

[attr$="value"]（以 value 结尾）

[attr\*="value"]（包含 value）

[attr~="value"]（属性包含单词 value）

**(4) 伪类选择器（Pseudo-classes）**

:hover（鼠标悬停）

:active（激活状态）

:focus（获得焦点）

:visited（已访问链接）

:first-child（第一个子元素）

:last-child（最后一个子元素）

:nth-child(n)（第 n 个子元素）

:not(selector)（排除选择器）

:checked（选中的表单元素）

:disabled / :enabled（禁用/启用状态）

**(5) 伪元素（Pseudo-elements）**

::before（在元素前插入内容）

::after（在元素后插入内容）

::first-line（首行样式）

::first-letter（首字母样式）

::selection（选中文本样式）

### 盒模型

**(1) 基本属性**

width / height（宽高）

margin（外边距）

padding（内边距）

border（边框）

box-sizing（盒模型计算方式）

content-box（默认，width 不包括 padding 和 border）

border-box（width 包括 padding 和 border）

**(2) 边框（Border）**

border-width（边框宽度）

border-style（边框样式，如 solid、dashed）

border-color（边框颜色）

border-radius（圆角，CSS3）

border-image（边框图片，CSS3）

**(3) 阴影（Box Shadow）**

box-shadow: h-shadow v-shadow blur spread color inset;（CSS3）

css

box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.3);

### 布局

**(1) 定位（Positioning）**

position

static（默认）

relative（相对定位）

absolute（绝对定位）

fixed（固定定位）

sticky（粘性定位，CSS3）

top / right / bottom / left（偏移）

z-index（层叠顺序）

**(2) 浮动（Float）**

float: left | right | none;

clear: left | right | both;（清除浮动）

**(3) Flexbox（弹性布局，CSS3）**

display: flex;（启用 Flex 布局）

flex-direction: row | column;（主轴方向）

justify-content（主轴对齐）

align-items（交叉轴对齐）

flex-wrap: nowrap | wrap;（是否换行）

flex-grow / flex-shrink / flex-basis（伸缩比例）

**(4) Grid（网格布局，CSS3）**

display: grid;（启用 Grid 布局）

grid-template-columns / grid-template-rows（定义网格）

grid-gap（网格间距）

grid-column / grid-row（项目位置）

**(5) 多列布局（Multi-column，CSS3）**

column-count（列数）

column-gap（列间距）

column-rule（列间分割线）

### 背景

background-color（背景色）

background-image（背景图）

background-repeat（是否重复）

background-position（背景位置）

background-size（背景大小，CSS3）

background-attachment（固定或滚动）

background-clip（背景裁剪）

background-origin（背景定位区域）

background-blend-mode（混合模式，CSS3）

### 文本与字体

**(1) 文本样式**

color（文字颜色）

font-family（字体）

font-size（字号）

font-weight（字重，如 bold）

font-style（斜体，如 italic）

text-align（对齐方式）

text-decoration（下划线、删除线）

text-transform（大小写转换）

line-height（行高）

letter-spacing（字符间距）

word-spacing（单词间距）

**(2) 高级文本效果（CSS3）**

text-shadow（文字阴影）

word-wrap: break-word;（允许长单词换行）

text-overflow: ellipsis;（溢出显示省略号）

white-space: nowrap;（禁止换行）

### 过渡与动画

**(1) 过渡（Transition）**

transition-property（过渡属性）

transition-duration（过渡时间）

transition-timing-function（速度曲线）

transition-delay（延迟时间）

简写：

css

transition: all 0.3s ease-in-out;

**(2) 动画（Animation）**

@keyframes（定义动画）

animation-name（动画名称）

animation-duration（持续时间）

animation-timing-function（速度曲线）

animation-delay（延迟）

animation-iteration-count（播放次数）

animation-direction（播放方向）

animation-fill-mode（动画结束状态）

简写：

css

animation: fadeIn 2s ease infinite;

### 变形

transform（变形）

translate(x, y)（移动）

rotate(deg)（旋转）

scale(x, y)（缩放）

skew(x, y)（倾斜）

matrix()（矩阵变换）

transform-origin（变形原点）

### 响应式设计

媒体查询（Media Queries）

css

@media (max-width: 768px) {

body { font-size: 14px; }

}

视口单位（Viewport Units）

vw（视口宽度百分比）

vh（视口高度百分比）

vmin / vmax（最小/最大视口比例）

### 其他重要属性

cursor（鼠标指针样式）

opacity（透明度）

visibility（可见性）

overflow（内容溢出处理）

clip-path（裁剪路径，CSS3）

filter（滤镜效果，CSS3）

css

filter: blur(5px) grayscale(50%);

## JavaScript

### **数据类型**

### **函数相关**

## DOM API

### 节点查询与选择

**（1）获取单个元素**

document.getElementById(id)

document.querySelector(selector)

**（2）获取多个元素**

document.getElementsByClassName(className)

document.getElementsByTagName(tagName)

document.querySelectorAll(selector)

**（3）特殊元素**

document.documentElement（<html>）

document.head

document.body

document.forms（获取所有 <form>）

document.images（获取所有 <img>）

document.links（获取所有 <a>）

### 节点操作

**（1）创建节点**

document.createElement(tagName)

document.createTextNode(text)

document.createDocumentFragment()（优化批量插入）

**（2）插入/删除节点**

parent.appendChild(node)

parent.insertBefore(newNode, refNode)

parent.removeChild(node)

parent.replaceChild(newNode, oldNode)

node.remove()（直接删除自身）

**（3）克隆节点**

node.cloneNode(deep)（true 深拷贝，false 浅拷贝）

### 节点属性与内容操作

**（1）属性操作**

element.getAttribute(name)

element.setAttribute(name, value)

element.removeAttribute(name)

element.hasAttribute(name)

element.attributes（返回所有属性集合）

**（2）类名操作**

element.classList.add(class)

element.classList.remove(class)

element.classList.toggle(class)

element.classList.contains(class)

**（3）内容操作**

element.innerHTML（获取/设置 HTML 内容）

element.textContent（获取/设置纯文本）

element.innerText（类似 textContent，但受 CSS 影响）

### 样式操作

element.style.property = value（修改行内样式）

window.getComputedStyle(element)（获取最终计算样式）

element.className（获取/设置 class 字符串）

### 事件API

**（1）事件监听**

element.addEventListener(event, handler, options)

element.removeEventListener(event, handler)

**（2）事件对象（Event）**

event.target（触发事件的元素）

event.currentTarget（绑定事件的元素）

event.preventDefault()（阻止默认行为）

event.stopPropagation()（阻止事件冒泡）

event.stopImmediatePropagation()（阻止后续监听器执行）

**（3）常见事件**

鼠标事件：click, dblclick, mouseenter, mouseleave, mousemove

键盘事件：keydown, keyup, keypress

表单事件：submit, input, change, focus, blur

窗口事件：load, resize, scroll

### 节点遍历与关系

**（1）父级关系**

node.parentNode

node.parentElement

**（2）子级关系**

node.childNodes（包含文本节点）

node.children（仅元素节点）

node.firstChild / node.lastChild

node.firstElementChild / node.lastElementChild

**（3）兄弟关系**

node.previousSibling / node.nextSibling

node.previousElementSibling / node.nextElementSibling

### 表单操作

formElement.submit()（提交表单）

formElement.reset()（重置表单）

inputElement.focus() / inputElement.blur()

inputElement.value（获取/设置输入值）

inputElement.checked（复选框/单选框选中状态）

### 动态加载与AJAX

document.write()（不推荐）

window.onload / DOMContentLoaded（页面加载事件）

### 其他API

element.matches(selector)（检查是否匹配选择器）

element.closest(selector)（查找最近的匹配祖先元素）

element.scrollIntoView()（滚动到元素可见区域）

element.getBoundingClientRect()（获取元素位置和尺寸）

## BOM API

### Windows对象

**（1）窗口控制**

window.open(url, name, features)（打开新窗口）

window.close()（关闭当前窗口）

window.moveTo(x, y)（移动窗口）

window.resizeTo(width, height)（调整窗口大小）

**（2）窗口属性**

window.innerWidth / window.innerHeight（视口宽高）

window.outerWidth / window.outerHeight（浏览器窗口宽高）

window.screenX / window.screenY（窗口相对于屏幕的位置）

window.scrollX / window.scrollY（页面滚动距离）

**（3）全局方法**

window.alert(message)（警告框）

window.confirm(message)（确认框，返回 true/false）

window.prompt(message, defaultText)（输入框，返回用户输入）

window.setTimeout(callback, delay)（延时执行）

window.setInterval(callback, delay)（定时执行）

window.clearTimeout(id) / window.clearInterval(id)（清除定时器）

### Navigator对象

**（1）浏览器信息**

navigator.userAgent（浏览器 UA 字符串）

navigator.appName（浏览器名称）

navigator.appVersion（浏览器版本）

navigator.platform（操作系统平台）

**（2）网络状态**

navigator.onLine（是否在线）

navigator.connection（网络信息，如 effectiveType: "4g"）

**（3）硬件/权限**

navigator.geolocation（地理位置 API）

navigator.geolocation.getCurrentPosition(success, error)

navigator.clipboard（剪贴板 API）

navigator.clipboard.writeText(text)（写入剪贴板）

navigator.clipboard.readText()（读取剪贴板）

### Screen对象

screen.width / screen.height（屏幕分辨率）

screen.availWidth / screen.availHeight（可用屏幕区域）

screen.colorDepth（颜色深度，如 24）

screen.orientation（屏幕方向，如 { type: "landscape-primary" }）

### Location对象

**（1）URL 属性**

location.href（完整 URL）

location.protocol（协议，如 https:）

location.host（主机名 + 端口，如 example.com:8080）

location.hostname（主机名，如 example.com）

location.port（端口号）

location.pathname（路径，如 /path/page.html）

location.search（查询字符串，如 ?id=123）

location.hash（哈希值，如 #section1）

**（2）页面跳转**

location.assign(url)（跳转到新 URL，保留历史记录）

location.replace(url)（替换当前 URL，不保留历史记录）

location.reload(force)（刷新页面，force=true 强制忽略缓存）

### History对象

history.length（历史记录数量）

history.back()（后退一页）

history.forward()（前进一页）

history.go(n)（跳转到第 n 条记录，n=0 刷新当前页）

history.pushState(state, title, url)（添加历史记录，不刷新页面）

history.replaceState(state, title, url)（替换当前历史记录）

### Localstorage&sessionStorage

**（1）localStorage（永久存储）**

localStorage.setItem(key, value)

localStorage.getItem(key)

localStorage.removeItem(key)

localStorage.clear()

**（2）sessionStorage（会话存储，关闭标签页后失效）**

方法与 localStorage 相同。

### Performance对象

performance.now()（高精度时间戳，单位毫秒）

performance.memory（内存使用情况，Chrome 支持）

performance.getEntries()（资源加载耗时统计）

### Console对象

console.log(msg)（普通日志）

console.error(msg)（错误日志）

console.warn(msg)（警告日志）

console.table(data)（表格形式打印数据）

console.time(label) / console.timeEnd(label)（计算代码执行时间）

## Vue

### 全局API

**(1) 应用实例（App）**

createApp()：创建 Vue 应用

javascript

import { createApp } from 'vue';

const app = createApp(App);

app.mount()：挂载到 DOM

javascript

app.mount('#app');

app.use()：使用插件

javascript

app.use(router);

app.component()：注册全局组件

javascript

app.component('MyComponent', MyComponent);

**(2) 全局配置**

app.config.globalProperties：添加全局属性

javascript

app.config.globalProperties.$http = axios;

app.config.errorHandler：全局错误处理

javascript

app.config.errorHandler = (err) => { console.error(err); };

### Composition API

**(1) 响应式数据**

ref()：定义响应式数据（基本类型）

javascript

const count = ref(0);

reactive()：定义响应式对象

javascript

const state = reactive({ count: 0 });

toRef() / toRefs()：解构响应式对象

javascript

const { count } = toRefs(state);

computed()：计算属性

javascript

const double = computed(() => count.value \* 2);

**(2) 生命周期钩子**

onMounted()：组件挂载后

javascript

onMounted(() => { console.log('mounted'); });

onUpdated()：组件更新后

onUnmounted()：组件卸载后

onBeforeMount() / onBeforeUpdate() / onBeforeUnmount()

**(3) 依赖注入**

provide() / inject()：跨组件通信

javascript

provide('key', value);

const value = inject('key');

**(4) 副作用与监听**

watch()：监听响应式数据

javascript

watch(count, (newVal) => { console.log(newVal); });

watchEffect()：自动追踪依赖

javascript

watchEffect(() => { console.log(count.value); });

### 组件相关API

**(1) 组件 Props**

defineProps()：定义 Props（<script setup>）

javascript

const props = defineProps({ title: String });

withDefaults()：设置默认值

javascript

const props = withDefaults(defineProps(), { title: '默认标题' });

**(2) 组件事件**

defineEmits()：定义自定义事件

javascrpt

const emit = defineEmits(['update']);

emit('update', newValue);

**(3) 组件引用**

ref() + template ref：获取 DOM 或子组件

javascript

const inputRef = ref(null);

<input ref="inputRef" />

defineExpose()：暴露组件方法（<script setup>）

javascript

defineExpose({ focus: () => inputRef.value.focus() });

### 路由

**(1) 路由配置**

createRouter()：创建路由

javascript

const router = createRouter({ history: createWebHistory(), routes });

useRouter() / useRoute()：访问路由实例和当前路由

javascript

const router = useRouter();

const route = useRoute();

**(2) 导航守卫**

router.beforeEach()：全局前置守卫

javascript

router.beforeEach((to, from) => { /\* ... \*/ });

### 状态管理

**(1) Pinia（推荐）**

defineStore()：定义 Store

javascript

const useCounterStore = defineStore('counter', { state: () => ({ count: 0 }) });

useStore()：使用 Store

javascript

const store = useCounterStore();

store.count++; // 直接修改

**(2) Vuex 4**

createStore()：创建 Store

javascript

const store = createStore({ state: { count: 0 } });

useStore()：在组件中使用

javascript

const store = useStore();

### 其他实用API

**(1) 响应式工具**

isRef() / isReactive()：检查响应式类型

unref()：获取 .value（如果是 ref）

markRaw()：标记对象为非响应式

**(2) 动态组件**

<component :is="currentComponent" />

**(3) 异步组件**

defineAsyncComponent()：懒加载组件

javascript

const AsyncComp = defineAsyncComponent(() => import('./Component.vue'));

**(4) 自定义指令**

v-model 自定义修饰符

javascript

const vMyDirective = { mounted(el, binding) { /\* ... \*/ } };