前端知识点

HTML

教程:https://www.runoob.com/html/html-tutorial.html

基础页面形式

```
<!DOCTYPE html>
<html lang="zh-cn">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>页面标题名称</title>
</head>
<body>

</body>
</html>
```

页面元素标签

1. `<html>`：这个标签是HTML文档的根标签，它描述了整个文档的结构和内容。
2. `<head>`：这个标签用于定义文档的头部信息，通常用来包含文档元数据，如标题、关键字等。
3. `<title>`：这个标签用于描述文档的标题，一般用来向浏览器提供文档的标题信息。
4. `<body>`：这个标签用于包含文档的主体内容。它包含文档中所有可见的内容，如文本、图片、视频等。

功能性布局标签

1. `<script>`：用来创建 JavaScript 代码。
2. `<style>`：用来创建 Css代码。

常用布局标签

3. `<div>` 和 `<span>`：这些标签用于创建文档的区域或子元素。`<div>` 标签通常用来创建文档容器，而 `<span>` 标签通常用来创建文本的局部区域。

4. `<h1>` 到 `<h6>`：这些标签用于创建标题，数字表示标题的级别。H1是最高级别的标题，H6是最低级别的标题。
5. `<p>`：这个标签用于创建段落。段落是文本的基本构成单位，通常用来划分页面中不同的区域。
6. `<ul>` 和 `<ol>`：这些标签用于创建无序列表（ul）和有序列表（ol）。无序列表通常用来展示多个项目，而有序列表通常用来展示一组有序的项目。
7. `<li>` 标签用于创建列表项，每个列表项都包含在一个 `<li>` 标签内。
8. `<a>` 标签用于创建超链接。它通常用来链接到其他文档或页面。
9. `<header>` 和 `<footer>`：这些标签用于创建页面的头部和底部区域。头部通常包含页面的标题、图标等信息，而底部通常包含版权信息、页面状态等信息。
10. `<nav>`：这个标签用于创建导航栏。导航栏通常包含多个链接，这些链接可以用来导航到其他页面或功能。
11. `<aside>`：这个标签用于创建侧边栏。侧边栏通常包含与页面主题相关的信息，如广告、分类、相关文章等。
12. `<section>`：这个标签用于创建页面的章节。章节通常包含与页面主题相关的不同部分，如标题、段落、图像等。
13.
14. `<figure>`：这个标签用于创建图像。通常img标签外层套一层figure标签
15. `<table>`：这个标签用于创建表格。表格通常包含多个行和列，这些行和列都包含在一个表格行或表格列内。
16. `<tbody>` 和 `<thead>`：这些标签用于创建表格的主体和表头。表头通常包含一个 th 标签，它指定表头的标题，而且通常还包含一个 scope 属性，它指示表头的位置。表主体通常包含一个 tbody 标签，它指示表格的主体内容。
17. `<td>`：这个标签用于创建单元格。单元格通常用于表格中的表格行。
18. `<th>`：这个标签用于创建表头单元格。表头单元格通常用于表格的表头。
19. `<tr>`：这个标签用于创建表格行。表格行通常包含在一个表格行内。
20. `<td>` 和 `<th>` 标签通常用于构建复杂的表格布局。您可以根据需要使用这些标签来创建具有复杂布局的表格。

表单标签

21. `<form>`：这个标签用于创建表单。表单可以包含多个输入字段、提交按钮等，以便用户能够向您的网站提交数据。
22. `<input>`：这个标签用于创建输入字段。`<input>` 标签通常包含一个 type 属性和一个 name 属性，它们用于指定输入字段的类型和名称。
23. `<textarea>`：这个标签用于创建多行文本输入字段。

24. `<select>`：这个标签用于创建下拉列表或单选按钮。它通常包含一个 name 属性和一个 options 属性，其中 options 属性包含多个值，这些值通常被映射到下拉列表的值框中。
25. `<option>`：这个标签用于表示下拉列表中的一个选项。它通常包含一个 value 属性，它指定值的值，而且通常还包含一个 selected 属性，它指定当前值是否被选中。
26. `<button>`：这个标签用于创建按钮。按钮通常包含一个 type 属性指定按钮的类型，而且通常还包含一个 name 属性指定按钮的名称。

媒体标签

27. `<img>` 标签用于创建图像。它通常用来嵌入图像文件或使用图像作为页面的背景。
28. `<audio>` 和 `<video>`：这些标签用于创建音频或视频。`<audio>` 标签通常用来创建音频文件，而 `<video>` 标签通常用来创建视频文件。

CSS

教程:https://www.runoob.com/css/css-tutorial.html

CSS基础知识

1. CSS 简介:CSS 指层叠样式表(Cascading Style Sheets),是一门用于渲染 HTML 元素样式的语言。通过 CSS 可以控制网页的颜色、布局、字体等外观效果。2. CSS 语法:CSS 规则由选择器、属性和值三部分构成。

```
选择器 {
    属性1: 值1;
    属性2: 值2;
}
```

2. CSS 选择器:用于选择页面上的 HTML 元素。常用选择器有: · 标签选择器:选择特定的 HTML 标签,如

```
/*  标签选择器 */
h1{
}
/*  类选择器 */
.h1{
}
/*  D 选择器 */
#h1{
}
/* 后代选择器  选择 h1 下的所有 span 元素*/
h1 span{
```

```
12 }
13 /*  通配符 选择所有元素*/
14 *{
15 }
```

3. CSS 层叠性:如果多个 CSS 规则作用在同一个元素上,会根据一定的层叠规则来判断这些规则哪个生效。
4. CSS 继承:子元素会继承父元素的某些 CSS 属性。如,给body设置字体,则所有子元素会继承该字体。
5. CSS 尺寸:通过 width、height、padding、margin、border等属性控制元素的尺寸。
6. CSS 定位:通过 position 属性设置元素的定位,并配合 offset 属性(top/right/bottom/left)来确定元素位置。
7. CSS 浮动:通过 float 属性实现元素的浮动,常用于实现文字环绕图片的效果。
8. CSS 盒子模型:每个 HTML 元素可以看作是一个矩形的框,CSS 盒子模型用于设置这个框的宽度、高度、边界、内边距和外边距。
9. CSS 显示模式:通过 display 属性设置元素的显示模式,常见的值有:
    - block:块级元素,独占一行,可以设置宽高。如
    - inline:行内元素,不独占一行,不能设置宽高。如
    - inline-block:行内块元素,既不独占一行,又可以设置宽高。
    - none:隐藏元素。
10. CSS 背景:通过 background 属性设置元素的背景,包括背景颜色、背景图片、背景重复等。
11. CSS 文本格式化:用于设置文本的颜色、对齐方式、装饰、阴影等。常用属性有:
    - color:文本颜色。
    - text-align:文本对齐方式。
    - text-decoration:文本装饰,如下划线、删除线等。
    - text-shadow:文本阴影。
12. CSS 字体:通过 @font-face 规则加载自定义字体,并使用 font-family 属性设置元素字体。
13. CSS 链接:用于设置超链接的样式。常用属性有:
    - a:链接选择器
    - a未访问的链接
    - a:visited:已访问的链接
    - a:hover:鼠标悬停的链接
    - a:active:鼠标点击的链接

14. CSS 列表:用于设置列表的样式。常用属性有:· list-style-type:设置列表项标记的样式。
    - list-style-position:设置列表项标记的位置。
    - list-style-image:设置列表项标记的图像。
15. CSS 表格:用于设置表格的颜色、对齐、边框等样式。
16. CSS 伪类:用于向某些选择器添加特殊的效果。常见的伪类有:
    - :hover:鼠标悬停状态
    - :active:鼠标点击状态
    - :focus:获取焦点状态
    - first-child:第一个子元素
    - :last-child:最后一个子元素
    - :nth-child():选择特定的子元素
    - :before:在元素内容前添加内容
    - :after:在元素内容后添加内容
17. CSS 伪元素:用于选择和标记 XML/HTML 中的一些位置。常见的伪元素有:
    - ::before:在元素之前插入内容
    - ::after:在元素之后插入内容
    - ::first-line:选择第一行文本
    - ::first-letter:选择首字母
    - ::selection:选择高亮区域
18. CSS 长度单位:用于定义元素尺寸的单位。常用单位有:· %:百分比,相对于父元素
    - px:像素单位,固定尺寸
    - em:相对长度单位,相对于父元素字体大小
    - rem:相对长度单位,相对于根元素字体大小
    - vw/vh:相对于视口尺寸的单位,vw=视口宽度,vh=视口高度
19. CSS 颜色:用于设置页面元素的颜色。三种表示颜色的方式:
    - 颜色名:如 red、blue 等
    - RGB:如 rgb(0,0,255)
    - HEX:16进制表示法,如 #0000ff21.
20. CSS 转场:用于设置元素变化的过渡效果,常用的属性有:· transition:用于设置过渡效果
    - transition-property:设置过渡属性
    - transition-duration:设置过渡时间

- transition-timing-function:设置过渡方式,如线性、缓冲等
- transition-delay:设置过渡延迟时间

常用基础使用

1. 字体:- font-family:设置字体,如 Arial、Verdana
   - font-size:设置字体大小,如 12px、1.2em
   - font-style:设置字体样式,如 normal、italic
   - font-weight:设置字体粗细,如 normal、bold
2. 文本: - color:设置文本颜色
   - text-align:设置文本对齐方式,左对齐、右对齐、居中
   - text-decoration:设置文本装饰,如下划线、删除线
   - text-indent:设置首行缩进
3. 背景:- background-color:设置背景颜色
   - background-image:设置背景图片
   - background-repeat:设置背景是否及如何重复
   - background-position:设置背景图片的位置
4. 宽高:- width:设置元素宽度
   - height:设置元素高度
   - line-height:设置行高
5. 边框:- border:设置元素的边框
   - border-width:设置边框宽度
   - border-style:设置边框样式,实线、 dashed 等
   - border-color:设置边框颜色
6. 间距:- padding:设置内边距
   - margin:设置外边距
7. 显示模式:- display:设置元素显示模式,如 block、inline、inline-block、none
8. 定位:- position:设置定位类型,static、relative、absolute、fixed
   - top:设置上外边距
   - right:设置右外边距
   - bottom:设置下外边距
   - left:设置左外边距
9. 浮动:- float:设置浮动方向,left、right、none
   - clear:清除浮动,left、right、both
10. 尺寸: - min-width:设置最小宽度
    - max-width:设置最大宽度
    - min-height:设置最小高度
    - max-height:设置最大高度

11. 盒子模型: - box-sizing:设置盒模型,content-box、border-box
    - overflow:设置内容溢出 Container 的方式,visible、hidden、scroll
12. 链接:- a:链接选择器
    - a未访问链接
    - a:visited:已访问链接
    - a:hover:鼠标悬停链接
    - a:active:鼠标点击链接
13. 列表: - list-style-type:设置列表项标记的样式,disc、circle、square、none
    - list-style-position:设置列表项标记的位置,inside、outside
14. Table 表格:- border-collapse:设置是否合并单元格边框,collapse、separate
    - border-spacing:设置相邻单元格的边框间隔
    - empty-cells:设置是否显示表格中的空单元格所占的空间
15. 转换: - transition:设置元素状态变化的过渡效果
    - transition-delay:设置元素过渡效果的延迟时间
    - transition-duration:设置元素过渡效果的持续时间
    - transition-property:设置哪个属性会产生过渡效果
    - transition-timing-function:设置元素过渡效果的运动曲线,linear、ease、cubic-bezier
16. 动画:- @keyframes:定义动画
    - animation:设置元素动画效果
    - animation-name:指定 @keyframes 定义的动画名称
    - animation-duration:设置动画持续时间
    - animation-timing-function:设置动画的运动曲线
    - animation-delay:设置动画延迟时间
    - animation-iteration-count:设置动画播放次数
    - animation-direction:设置动画播放方向,normal、alternate、reverse、alternate-reverse
    - animation-play-state:设置动画播放状态,running、paused
17. flexbox 弹性布局:- display: flex | inline-flex:将元素定义为 flex 容器
    - flex-direction:设置主轴方向,row | row-reverse | column | column-reverse
    - flex-wrap:设置是否换行,nowrap | wrap | wrap-reverse
    - flex-flow:复合属性,设置 flex-direction 和 flex-wrap 属性
    - justify-content:设置主轴对齐方式,flex-start | flex-end | center | space-between | space-around
    - align-items:设置交叉轴对齐方式,flex-start | flex-end | center | baseline | stretch
    - flex-grow:设置项目放大比例
    - flex-shrink:设置项目缩小比例
    - flex-basis:设置项目占据的主轴空间
    - align-self:设置单个项目的对齐方式,auto | flex-start | flex-end | center | baseline | stretch

媒体查询

CSS 媒体查询用于针对不同的媒体类型定义样式。它通过允许您根据设备类型,设备宽度,设备高度,设备分辨率以及其他媒体功能来自定义内容的呈现方式。通过媒体查询,我们可以针对不同设备定制适配的样式,轻松实现响应式网站

语法格式为:

```
@media not|only mediatype and (media feature) {
    CSS-代码;
}
```

- mediatype:可选,指定要匹配的媒体类型,可以是 all、print、screen 等。
- not:可选,用于排除某个媒体类型。
- only:可选,用于指定某个特定的媒体类型,且设备必须只匹配这个媒体类型。
- and:可选,用于连接多个媒体功能查询。
- (media feature):必选,包含一个或多个媒体功能查询。常见的媒体类型有:- all:用于所有设备
- print:用于打印机和打印预览
- screen:用于电脑屏幕、平板电脑、智能手机等
- speech:用于屏幕阅读器等发声设备常见的媒体功能查询有:- width:可视区宽度
- height:可视区高度
- aspect-ratio:宽高比
- orientation:横屏或竖屏
- resolution:屏幕分辨率示例:

```
@media (min-width: 768px) and (max-width: 992px) {
  /* 针对屏幕宽度在 768px 到 992px 之间的设备的样式 */
}

@media only screen and (max-height: 600px) {
  /* 针对屏幕高度不超过 600px 的设备的样式 */
}

@media (orientation: portrait) {
  /* 针对竖屏设备的样式 */
}

@media print {
```

```
  /* 针对打印机和打印预览的样式 */
}

/* 当设备宽度在 600px 到 1200px 之间时,使用这组样式 */
@media (min-width: 600px) and (max-width: 1200px) {
  body {
    background: orange;
  }
}

/* 当设备宽度超过 1200px 时,使用这组样式 */
@media (min-width: 1200px) {
  body {
    background: blue;
  }
}

/* 当设备高度少于 500px 时,使用这组样式 */
@media (max-height: 500px) {
  body {
    background: red;
  }
}

/* 针对打印机和打印预览,使用这组样式 */
@media print {
  body {
    background: gray;
  }
}

/* 针对屏幕宽度超过 900px 的横屏设备,使用这组样式 */
@media (min-width: 900px) and (orientation: landscape) {
  body {
    background: green;
  }
}

/* 针对屏幕宽度在 600px 到 900px 之间且设备像素密度超过 2 的设备,使用这组样式*/
@media (min-width: 600px) and (max-width: 900px) and (min-resolution: 2dppx) {
  body {
    background: pink;
  }
}

/* 除了打印设备外的所有设备,使用这组样式 */
@media not print {
```

```
61    body {
62      background: purple;
63    }
64  }
```

JS

JS基础概要

1. JS 简介:JavaScript 是一门客户端脚本语言,运行在用户浏览器中,可以实现网页的动态效果和交互性。
2. JS 输出:使用 alert()、console.log() 和 document.write() 向 HTML 输出文本。
3. JS 变量:使用 var、let 和 const 来定义变量。let 和 const 是 ES6 中引入的。
4. JS 数据类型:包括 Number、String、Boolean、Null、Undefined、Object、Symbol 等。
5. JS 注释:单行注释 // 和多行注释 /* */。
6. JS 运算符:算数运算符、赋值运算符、比较运算符、逻辑运算符等。
7. JS 函数:使用 function 关键字定义函数,函数是 JS 中的基本单元。
8. JS 对象:对象是由属性和方法组成的数据结构。
9. JS 事件:onclick、onmouseover 等,用于执行点击、鼠标经过等操作时的功能。
10. JS 字符串:使用单引号或双引号定义,可以使用 + 进行连接,lenght 获取长度。
11. JS 数组:使用 [] 定义数组,可以存储不同数据类型,使用 length 获取长度。
12. JS 条件语句:if/else、switch,用于判断和执行代码块。
13. JS 循环语句:for、while、do/while,用于循环执行代码块。
14. JS Number:用于定义数字,可以使用 Number()、parseInt()、parseFloat() 转换。
15. JS Math:用于数学运算,如 Math.PI、Math.round()、Math.random() 等。
16. JS Date:用于处理日期和时间。可以使用 Date() 获取当前时间。
17. JS RegExp:用于定义正则表达式,常用于字符串的匹配、检索和替换。
18. JS 执行环境:JS 代码可以运行在浏览器环境和 Node.js 环境中。
19. JS 面向对象:使用关键字 class 定义类,使用 constructor 定义构造函数,使用 this 关键字定义属性和方法,使用 extends 继承。
20. JS 函数表达式:除了使用 function 关键字定义函数外,也可以使用变量名 = function() {} 的形式定义匿名函数。
21. JS 闭包:内部函数可以访问外部函数的变量和参数,内部函数的作用域链包含外部函数的作用域。

22. JS 原型与原型链:每个对象都有一个 prototype 属性,该属性指向其原型对象,原型对象的 prototype 指向其原型对象,此过程构成原型链。
23. JS 作用域:作用域决定了在代码中的哪部分可以访问某个变量。JS 有函数作用域和块作用域(ES6 引入)。
24. JS 回调函数:将一个函数作为参数传递给另一个函数,然后在稍后执行这个函数的函数。
25. JS 浏览器对象:包括 Window、Navigator、History、Location、Screen 等对象。
26. JS Web API:包括 DOM、BOM、Storage、Geolocation、Web Workers 等 API。
27. JS 事件循环:JS 是单线程语言,事件循环机制用于协调事件、回调和其它任务的执行。
28. JS 错误处理:使用 try/catch 语句捕获并处理异常错误。
29. JS 模块:使用关键字 export 向外导出,import 向内导入。ES6 引入模块的概念。

基础示例

```
// 1. 问候语:
var name = "John";
var msg = "Hello " + name + "!";
alert(msg);  // "Hello John!"

// 当前时间:
var date = new Date();
var hours = date.getHours();
var minutes = date.getMinutes();
var seconds = date.getSeconds();
var time = hours + ":" + minutes + ":" + seconds;
alert(time);  // 显示当前时间,如:16:30:25

// 简单计算器:
var num1 = 10;
var num2 = 20;
var sum = num1 + num2;
var sub = num1 - num2;
var mul = num1 * num2;
var div = num1 / num2;
alert(sum);  // 30
alert(sub);  // -10
alert(mul);  // 200
alert(div);  // 0.5

//掷骰子:
var num = Math.floor(Math.random() * 6 + 1);
alert(num);  // 随机显示1-6之间的数字

```

```
// BMI 计算:
var height = 1.75;  // 身高,米
var weight = 70;   // 体重,公斤
var bmi = weight / (height * height);
alert(bmi);  // 显示BMI指数,如22.9

// 数组操作:
var fruits = ["Apple", "Orange", "Banana"];
fruits.push("Pear");      // 向数组的末尾添加一个元素
fruits.pop();             // 从数组的末尾移除一个元素
fruits[1] = "Peach";      // 修改第二个元素
alert(fruits.length);   // 3
```

数组方法

```
// push():向数组的末尾添加一个或多个元素,返回数组的新长度。
let fruits = ["Apple", "Orange"];
fruits.push("Banana");  // ["Apple", "Orange", "Banana"]

// pop():从数组的末尾移除一个元素,返回移除的元素。
let fruits = ["Apple", "Orange", "Banana"];
fruits.pop();  // "Banana"   fruits: ["Apple", "Orange"]

// shift():从数组的开头移除一个元素,返回移除的元素,其他元素下标减 1。
let fruits = ["Apple", "Orange", "Banana"];
fruits.shift();  // "Apple"   fruits: ["Orange", "Banana"]

// unshift():向数组的开头添加一个或多个元素,返回数组的新长度,其他元素下标加 1。
let fruits = ["Orange", "Banana"];
fruits.unshift("Apple");  // 3   fruits: ["Apple", "Orange", "Banana"]

// splice():向数组中添加元素,从数组中删除元素,或者同时进行添加和删除元素。
fruits.splice(1, 0, "Peach");  // 在索引 1 位置添加 "Peach"
fruits.splice(0, 1);           // 删除索引 0 位置的元素


// join ()
let fruits = ['a','b','c']
fruits.join('-') // 输出字符串‘a-b-c’
```

JQuery

基础简介

1. jQuery 简介:jQuery 是一个 JavaScript 库,它使 HTML 文档遍历和操作、事件处理、动画和 Ajax 变得更简单。
2. jQuery 下载与引入:可以下载 jQuery 库文件,也可以通过 CDN 在线引入,一般在 HTML 页面头部引入。
3. jQuery 选择器:jQuery 使用 CSS 选择器来选取 HTML 元素,常用选择器有 #id、.class、element、* 等。
4. jQuery 事件:如 click、dblclick、mouseenter、mouseleave、keyup、keydown 等,用来触发 JavaScript 函数的执行。
5. jQuery 效果:如 hide()、show()、toggle()、fadeIn()、fadeOut()、slideUp()、slideDown() 等,用于控制 HTML 元素的显示和隐藏。
6. jQuery HTML 操作:如 text()、html()、append()、prepend()、after()、before()、remove()、empty() 等,用于创建、获取和操作 HTML 元素及内容。
7. jQuery CSS 操作:如 addClass()、removeClass()、toggleClass()、css()、height()、width() 等,用于创建、操作和获取 HTML 元素的 CSS。
8. jQuery 遍历:可以对选取的 HTML 元素进行遍历操作,常用遍历方法有 parent()、children()、siblings()、next()、prev() 等。
9. jQuery AJAX:jQuery 通过 $.ajax() 方法实现 AJAX 功能,可以使用 GET、POST 与服务器进行异步交互和通信。
10. jQuery 工具方法:如 $.each()、$.trim()、$.type()、$.isArray()、$.extend() 等,提供常用的工具功能。
11. jQuery 插件:jQuery 有大量开源插件,方便扩展功能,如 日期选择器、图片展示、树形菜单等,可以直接下载和使用。

基础示例

```
// 文档加载完成后执行:
$(document).ready(function() {
  // jQuery 代码
});

// 或者简写为:
$(function() {
  // jQuery 代码
});

// 选择器选取元素:
$("#id")        // id选择器
$(".class")     // class选择器
```

```
$("div")      // 元素选择器
$("*")        // 全选选择器

// 操作元素:
$("p").text("Hello");     // 获取/设置文本内容
$("p").html("<b>Hello</b>"); // 获取/设置 HTML 内容
$("p").val("John");       // 获取/设置元素的值
$("p").css("color","red"); // 获取/设置 CSS 属性
$("p").attr("title","Hi"); // 获取/设置元素属性

// 添加元素:
$("p").append("<b>Hello</b>");   // 向元素内追加内容
$("p").prepend("<b>Hello</b> "); // 向元素内前置内容
$("div").after("<p>Hello</p>");  // 向元素后添加内容
$("div").before("<p>Hello</p>"); // 向元素前添加内容

// 删除元素:
$("p").remove();         // 删除 p 元素
$("p").empty();          // 清空 p 元素
$("div").remove(".intro"); // 删除 class 为 intro 的 div 元素

// 事件绑定:
$("p").click(function(){
  $(this).hide();   // 绑定 click 事件后的回调函数
});

// jQuery 动画:
$("p").hide(1000);      // 1秒隐藏p元素
$("p").show(1000);      // 1秒显示p元素
$("p").fadeIn(1000);    // 1秒淡入p元素
$("p").fadeOut(1000);   // 1秒淡出p元素
```

Vue

教程：https://v2.cn.vuejs.org/v2/guide/index.html

基础教程

1. Vue 简介:Vue.js 是一套构建用户界面的渐进式框架,它核心库专注于视图层,采用组件化模式,底层使用 render 函数或模板创建虚拟 DOM。2. Vue 安装:可以通过 CDN 引入,也可以通过 npm 安装在项目中使用,引入方式为:

```
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.11/dist/vue.js"></script>
```

2. Vue 声明式渲染:使用 v- 开头指令渲染展示属于的复杂信息。

```
<span v-bind:title="message">鼠标悬停显示</span>
```

```
data: {
  message: '页面加载于 ' + new Date()
}
```

3. Vue 条件与循环:通过 v-if 和 v-for 渲染元素。

```
<ul v-if="todos.length > 0">
  <li v-for="todo in todos">
    {{ todo }}
  </li>
</ul>
```

4. Vue 事件处理:使用 v-on: 绑定事件监听,常简写为 @。

```
// 简写前
<button v-on:click="say('Hi')">Say Hi</button>

// 简写后
<button @click="say('Hi')">Say Hi</button>
```

```
methods: {
  say: function(msg) {
    alert(msg)
  }
}
```

5. Vue 双向绑定:使用 v-model 实现表单元素和数据的双向绑定。

```
<input v-model="message" placeholder="edit me">
<p>{{ message }}</p>
```

6. Vue 组件:Vue 组件用于构建可复用的 UI 元素。

```
Vue.component('todo-item', {
  template: '<li>这是个待办事项</li>'
})
```

```
<ul>
  <todo-item></todo-item>
</ul>
```

7. Vue 计算属性和侦听器:computed 和 watch 选项。9. Vue 生命周期:Vue 实例经历的生命周期钩子函数。

生命周期

Vue 实例有一个完整的生命周期,从开始创建、初始化数据、编译模板、挂载 DOM、更新、渲染、卸载等一系列过程。Vue 为生命周期中的每个状态都提供了钩子函数,允许我们在合适的时间执行自定义逻辑。Vue 生命周期分为以下几个阶段:

1. 初始化阶段:- beforeCreate:实例刚被创建,数据观察者和事件还未开始追踪。
- created:实例已经创建完成,数据观察者和事件配置已经初始化完成。

2. 编译阶段:- beforeMount:模板编译/挂载之前调用。
- mounted:模板编译/挂载之后调用。

3. 更新阶段:- beforeUpdate:组件更新之前调用。
- updated:组件更新之后调用。

4. 销毁阶段:- beforeDestroy:实例销毁之前调用。
- destroyed:实例销毁之后调用。 常用的生命周期钩子函数有:- created:发送ajax请求、订阅事件等初始操作。
- mounted:调用this.$nextTick触发渲染后工作。
- updated:调用this.$nextTick触发更新后工作。
- destroyed:移除事件监听器、定时器等。

基础示例

1. 实例化 Vue:

```
new Vue({
  // 选项
```

```
})
```

1. 元素绑定数据:

```
<!-- 简写前 -->
<span v-bind:title="message">这是内容</span>

<!-- 简写后 -->
<span :title="message">这是内容</span>
```

```
data: {
  message: '页面加载于 ' + new Date()
}
```

2. 条件渲染:

```
<span v-if="seen">现在你看到我了</span>
```

```
data: {
  seen: true
}
```

3. 循环渲染:

```
<ul>
  <li v-for="todo in todos">
    {{ todo.text }}
  </li>
</ul>
```

```
data: {
  todos: [
    { text: '学习 JavaScript' },
```

```
    { text: '学习 Vue' },
    { text: '整个牛项目' }
  ]
}
```

4. 事件处理:

```
<button v-on:click="say('hi')">Say hi</button>
```

```
methods: {
  say: function (message) {
    alert(message)
  }
}
```

5. 双向数据绑定:

```
<input v-model="message" placeholder="edit me">
<p>{{ message }}</p>
```

```
data: {
  message: 'Hello!'
}
```

6. 组件:

```
Vue.component('todo-item', {
  template: '<li>这是个待办事项</li>'
})
```

```
<ul>
  <todo-item></todo-item>
</ul>
```

Element-UI

教程：https://element.eleme.io/#/zh-CN/component/button

1. 基本用法:

```
<el-button>默认按钮</el-button>
<el-button type="primary">主要按钮</el-button>
<el-button type="success">成功按钮</el-button>
<el-button type="info">信息按钮</el-button>
<el-button type="warning">警告按钮</el-button>
<el-button type="danger">危险按钮</el-button>
```

2. 禁用状态:

```
<el-button disabled>默认按钮</el-button>
<el-button type="primary" disabled>主要按钮</el-button>
```

3. 图标按钮:

```
<el-button type="primary" icon="el-icon-search">搜索</el-button>
<el-button type="primary" icon="el-icon-edit"></el-button>
```

4. 按钮组:

```
<el-button-group>
  <el-button type="primary" icon="el-icon-arrow-left">上一页</el-button>
  <el-button type="primary">下一页<i class="el-icon-arrow-right el-icon--right"><
</el-button-group>
```

5. 加载中状态:

```
<el-button type="primary" :loading="true">加载中</el-button>
```

6. 按钮尺寸:

```
<el-button type="primary" size="small">小型按钮</el-button>
```

```
<el-button type="primary">默认按钮</el-button>
<el-button type="primary" size="medium">中等按钮</el-button>
<el-button type="primary" size="large">大型按钮</el-button>
```

ECharts

教程：https://echarts.apache.org/examples/zh/index.html

ECharts 是百度推出的一款开源的数据可视化工具,这里是 ECharts 的基础使用:1. 引入 ECharts:可以通过 CDN 引入,也可以通过 npm 安装在项目中使用,CDN 引入方式如下:

html

```
<script src="https://cdn.bootcdn.net/ajax/libs/echarts/4.8.0/echarts.min.js"></script>
```

1. 准备一个具备大小(宽高)的 DOM 容器:

html

```
<div id="main" style="width: 600px;height:400px;"></div>
```

2. 基于准备好的 DOM 元素初始化 ECharts 实例:

js

```
var myChart = echarts.init(document.getElementById('main'));
```

3. 指定图表的配置项和数据:

js

```
var option = {
 title: {
  text: '折线图'
 },
 tooltip: {},
 legend: {
  data:['销量']
 },
 xAxis: {
  data: ["衬衫","羊毛衫","雪纺衫","裤子","高跟鞋","袜子"]
 },
 yAxis: {},
 series: [{
  name: '销量',
  type: 'line',
  data: [5, 20, 36, 10, 10, 20]
 }]
};
```

4. 使用刚指定的配置项和数据显示图表:

```
js
myChart.setOption(option);
```

5. 监听浏览器缩放,图表对象调用 resize() 方法:

```
js
window.onresize = function(){
  myChart.resize();
}
```

这是 ECharts 的基础使用,我们从引入 ECharts 开始,一步步实现了一个简单的折线图表,后续我们会继续深入学习 ECharts 各种图表的配置与高级用法,欢迎您提出任何疑问。我们一起学习,共同进步。谢谢您的支持与信任! （已编辑）