



讲师：李振良（阿良）

今天课题：《Web前端开发基础》

学院官网：[www.aliangedu.cn](http://www.aliangedu.cn)



阿良个人微信



DevOps技术栈公众号

# Web前端开发基础

- HTML
- CSS
- JavaScript
- JavaScript库: jQuery

# 网页组成

例如一个index.html页面：

- HTML标记语言：组成网页架构的元素组件
- CSS 样式语言：美化网页的样式
- JavaScript 程式语言：控制网页的动态效果
- JQuery 程式语言：协助及加强JavaScript的实现



例如：大家所熟知的百度首页，右击->查看网页源代码即可看到HTML内容

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>首页</title>
  <meta name=""Keywords content="关键字">
  <meta name="Description" content="简介、描述">
  <link rel="stylesheet" href="./css/main.css">
  <style>
    /* css代码 */
  </style>
  <script type="text/javascript" src="./js/main.js"></script>
</head>
<body>
  <!-- 内容 -->
  <script type="text/javascript">
    // js代码
  </script>
</body>
</html>
```

网页代码结构

# HTML 标记语言

- HTML介绍
- 文本格式化标签
- 列表标签
- 超链接标签
- 图片标签
- 表格标签
- 表单标签
- 列表标签
- 按钮标签
- <div> 标签

# | HTML介绍

**HTML：** 是一种用于创建网页的标记语言，可以使用HTML创建网页，用浏览器打开会自动解析。

HTML是由标签和内容构成。

# HTML介绍

```
<html>
<head>
  <title>文档的标题</title>
</head>
<body>
  文档的内容...
</body>
</html>
```

HTML代码结构

# HTML：文本格式化标签

标签	描述
 	换行
<h1>~</h1>	标题，定义标题字体大小，1最大，6最小
<p>...</p>	段落
<i>...</i>	斜体
<cite> </cite>	引用
<b>...</b>	加粗
<strong>...</strong>	强调加粗
<del> </del>	删除线



# HTML：列表标签

标签	描述	参数
<ul>	无序列表	<ul style="list-style-type: none"><li>• type=disc 默认实心圆</li><li>• square 实心方块</li><li>• circle 空心圆</li></ul>
<ol>	有序列表	type=1 默认数字，其他值：A/a/I/i/1
<li>	列表项目	在有序列表和无序列表中用

# HTML：超链接标签

超链接标签：<a href="网址" ></a>

属性	描述
href	指定链接跳转地址
target	链接打开方式，常用值：_blank 打开新窗口
title	文字提示属性
name	定义锚点

# HTML： 图片标签

<img src= “图片文件路径” alt= “图片提示” >

属性	描述
alt	图片加载失败时的提示信息
title	文字提示属性

# HTML: 表格标签

```
<table border="1">
  <thead>
    <tr>
      <th>主机名</th>
      <th>IP</th>
      <th>操作系统</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>www.aliangedu.cn</td>
      <td>192.168.1.10</td>
      <td>CentOS7</td>
    </tr>
  </tbody>
</table>
```

- table 表格标签
- thead 表格标题
- tr 行标签
- th 列名
- tbody 表格内容
- td 列内容

# HTML：表单标签

表单标签：<form> </form>

属性	描述
action	提交的目标地址(URL)
method	提交方式：get(默认)和post
enctype	编码类型 <ul style="list-style-type: none"><li>application/x-www-form-urlencoded 默认值，编码字符</li><li>multipart/form-data 传输数据为二进制类型，如提交文件</li><li>text/plain 纯文本的传输</li></ul>

表单项标签：<input>

属性	描述
type	<ul style="list-style-type: none"><li>text：单行文本框</li><li>password：密码输入框</li><li>checkbox：多选框</li><li>radio：单选框</li><li>file：文件上传选择框</li><li>button：普通按钮</li><li>submit：提交按钮</li><li>reset：重置按钮</li></ul>
name	表单项名，用于存储内容值
value	表单项的默认值
disabled	禁用该元素
checked	默认被选中，值也是checked

# HTML：列表标签

下拉列表标签：<select> </select>

属性	描述
name	下拉列表的名称，用于存储下拉值
disabled	禁用该元素
multiple	设置可以选择多个项目
size	指定下拉列表中的可见行数

下拉列表选项标签：<option> </option>

属性	描述
value	选项值
selected	默认下拉项

# HTML: 按钮标签

按钮标签: `<button type= "submit" ></botton>`

type可选值:

- button: 普通
- submit: 提交
- reset: 重置

# HTML: <div>标签



<div> 标签用于在HTML文档中定义一个区块。常用于将标签集中起来，然后用样式对它们进行统一排版。





# CSS 样式语言

- CSS介绍
- 使用方法
- 选择器
- 常用属性

**CSS：**是一种用于修饰网页的文本样式语言，还可以配合Javascript脚本语言动态对网页各元素操作。

# CSS使用方法

## 1、内联方式（行内样式）

```
<p style="color:red">在HTML中如何使用css样式</p>
```

## 2、内部方式（内嵌样式），在head标签中使用

```
<style type="text/css">
  p {
    color:red;
  }
</style>
```

## 3、外部导入方式（推荐），在head标签中使用

```
<link href="main.css" type="text/css" rel="stylesheet"/>
```

# 选择器

**选择器：** 需要改变样式的HTML元素

**格式：** 选择器{属性:值;属性:值;属性:值;....}

**常见选择器：** 标签选择器、类选择器、ID选择器、派生选择器

# 选择器：元素

**元素选择器：**使用html标签作为选择器，为指定标签设置样式。

示例1：h1元素设置样式

```
h1 {  
  color: red;  
  font-size: 14;  
}
```

示例2：多个元素设置样式

```
h1,h2,h3,h4,h5,h6 {  
  color: green;  
}
```

示例3：子元素会继承最高级元素所有属性

```
body {  
  color: #000;  
  font-family: Verdana, serif; /*字体*/  
}
```

# 选择器：ID

**id选择器：**使用 “id” 作为选择器，为指定id设置样式。

使用格式：#id名{样式...}

**特点：**

- 每个id名称只能在HTML文档中出现一次
- 在实际开发中，id一般预留JavaScript使用

第一步：给标签指定id

```
<p id="t">...</p>
```

第二步：针对id设置样式

```
#t {  
  color: red;  
}
```

# 选择器：类

**类选择器：**使用“类名”作为选择器，为指定类设置样式。

使用格式：`.类名{样式...}`

第一步：给标签指定类

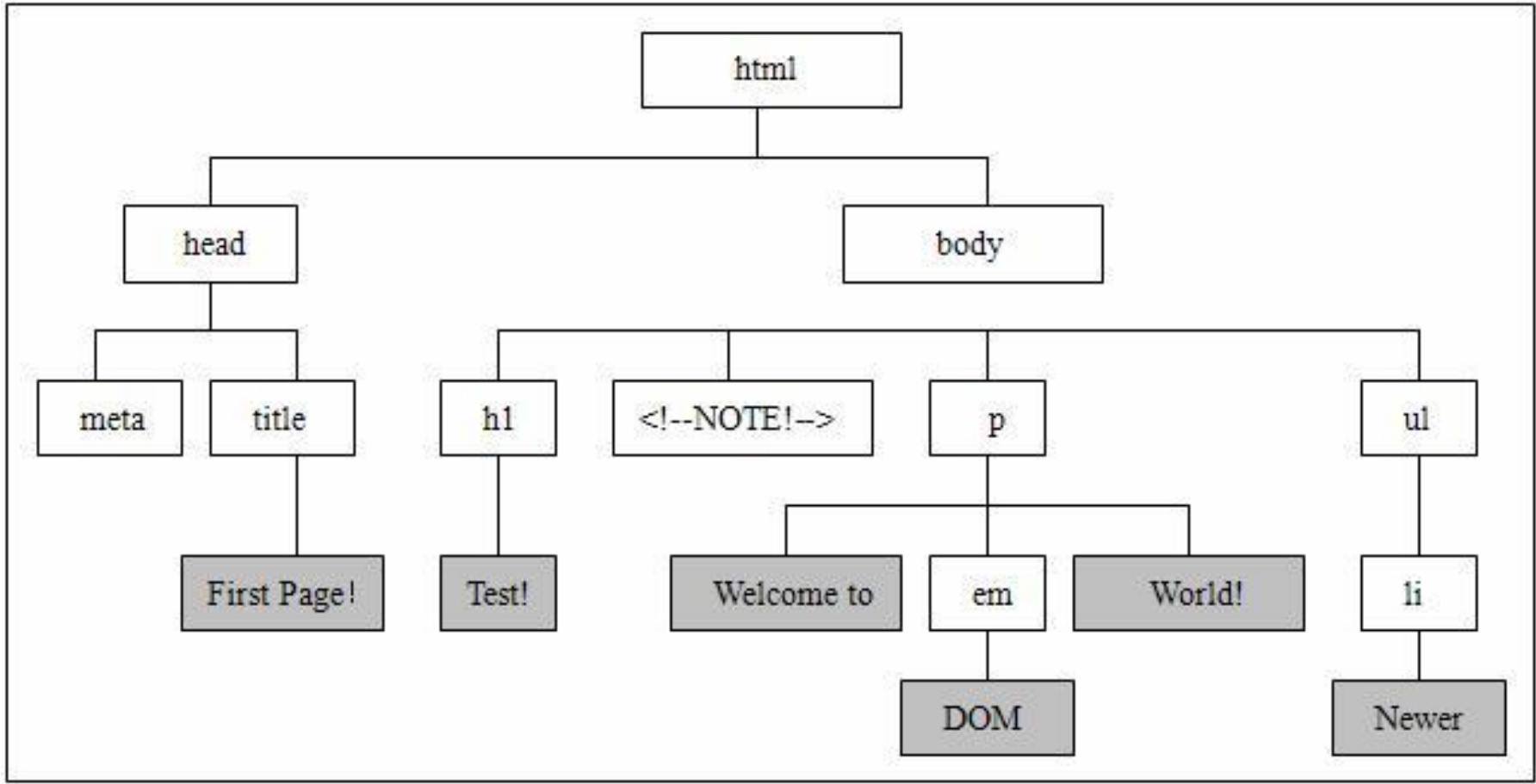
```
<p class="c">...</p>
```

第二步：针对类设置样式

```
.c {  
  color: red;  
}
```

# 选择器：派生

**派生选择器：** 依据元素在其位置的上下文关系来定义样式。



示例：

```
<style type="text/css">
  .c p {
    color: red;
  }
</style>

<div class="c">
  <h1>一号标题</h1>
  <p>这是一个段落</p>
</div>
```



## CSS常用属性：内边距和外边距



**padding (内边距)**：钻戒到盒子内边框的距离

**margin (外边距)**：钻戒盒子距离桌子边缘的距离

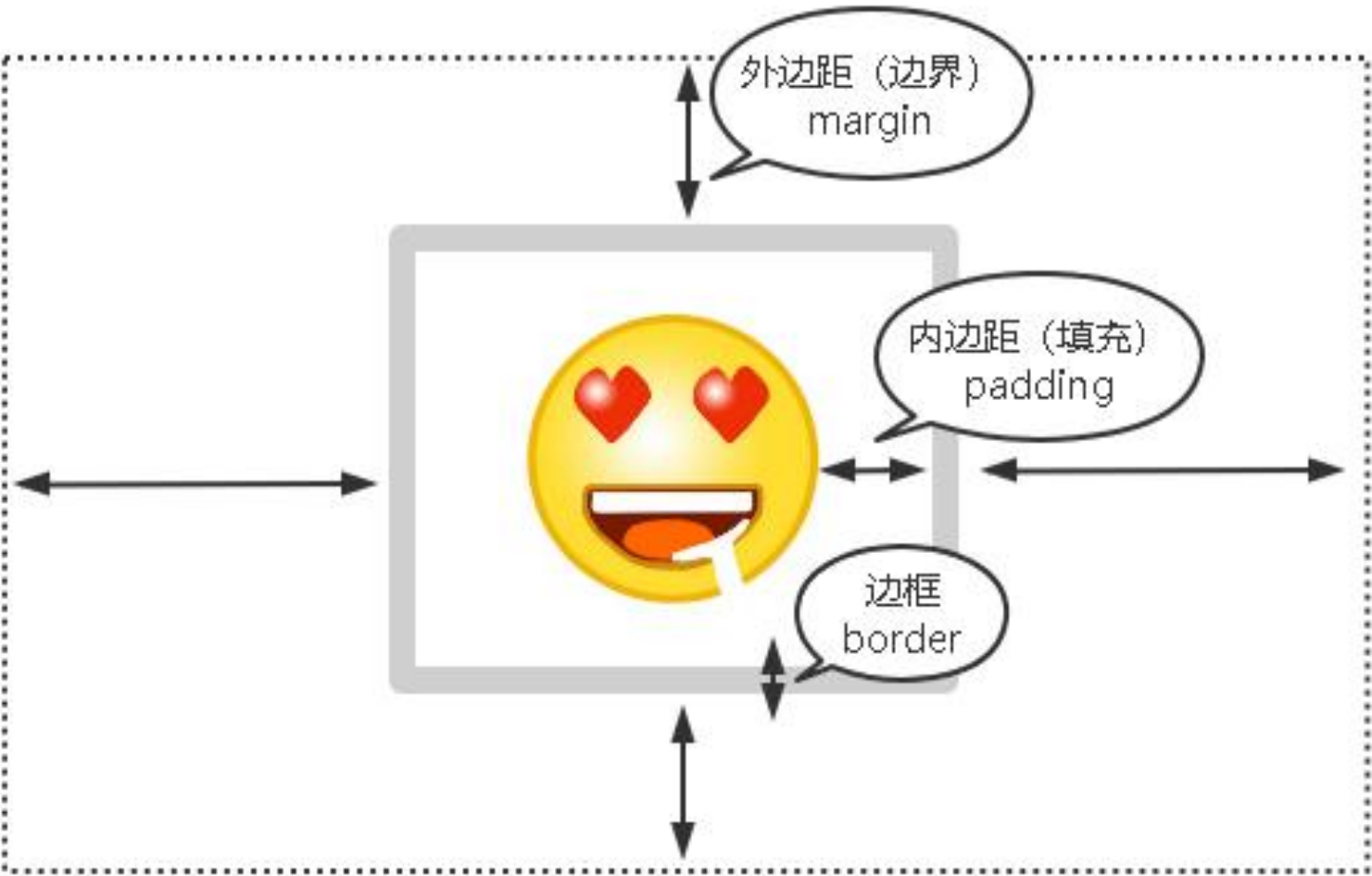
**border**：钻戒盒子边框宽度

# CSS常用属性：内边距和外边距

属性	描述
padding	设置四边的内边距
padding-top	上内边距
padding-right	右内边距
padding-bottom	下内边距
padding-left	左内边距

属性	描述
margin	设置四边的外边距，使用方法同padding
margin-top	上外边距
margin-right	右外边距
margin-bottom	下外边距
margin-left	左外边距

示例：  
padding: 10px 5px 15px 20px # 上右下左  
padding: 10px 5px 15px # 上右下  
padding: 10px 5px # 上右  
padding: 10px # 四边都是10px



示例

# CSS常用属性：字体 font-\*

属性	描述	值
font-size	设置字体的尺寸	<ul style="list-style-type: none"><li>xx-small、x-small、small、medium、large、x-large、xx-large, 从小到大, 默认值 medium</li><li>length 固定长度, 例如12px</li></ul>
font-family	字体系列。可以写多个如果第一个不支持, 使用下一个	Microsoft YaHei
font-weight	设置字体的粗细	<ul style="list-style-type: none"><li>normal 默认值</li><li>bold 粗体</li><li>bolder 更粗</li><li>lighter 更细</li></ul>
font-style	字体样式	<ul style="list-style-type: none"><li>normal 正常</li><li>italic 斜体</li><li>oblique 倾斜的字体</li></ul>

# CSS常用属性：文本

属性	描述	值
color	字体颜色	<ul style="list-style-type: none"><li>• 颜色名称，例如red</li><li>• 十六进制值，例如#ff0000</li><li>• rgb 代码，例如rgb(255,0,0)</li></ul>
text-align	文本对齐方式	<ul style="list-style-type: none"><li>• left左边</li><li>• right 右边</li><li>• center 中间</li><li>• justify 两端对齐文本效果</li></ul>
text-decoration	文本修饰	<ul style="list-style-type: none"><li>• none 默认，定义标准的文本，例如去掉超链接下划线</li><li>• line-through 删除线</li><li>• underline 文本下加一条线</li></ul>
text-overflow	文本溢出后显示效果	<ul style="list-style-type: none"><li>• clip 修剪文本</li><li>• ellipsis 显示省略号来代表被修剪的文本</li><li>• string 使用给定的字符串来代表被修剪的文本</li></ul>
letter-spacing	字符间的距离	<ul style="list-style-type: none"><li>• normal 默认</li><li>• length 自定义间距</li></ul>
line-height	行间的距离（行高）	<ul style="list-style-type: none"><li>• normal 默认</li><li>• length 设置固定值</li></ul>

# CSS常用属性：边框 border-\*

属性	描述	值
border	所有边框样式的缩写	示例：border: 1px solid blue; 宽度 样式 颜色
border-radius	圆角边框	直接写像素
box-shadow	给元素添加阴影	<p>格式：box-shadow: h-shadow v-shadow blur spread color inset;</p> <ul style="list-style-type: none"><li>• h-shadow 必选，水平阴影的位置</li><li>• v-shadow 必选，垂直阴影的位置</li><li>• blur 可选，模糊程度</li><li>• spread 可选，阴影的大小</li><li>• color 可选，阴影的颜色</li><li>• inset 可选，从外层的阴影（开始时）改变阴影内侧阴影</li></ul> <p>示例1：box-shadow: 1px 2px 3px 1px #c2c2c2;</p> <p>示例2：box-shadow: 0 5px 20px 0 #e8e8e8;</p>

# CSS常用属性：背景 background-\*

属性	描述	值
background-color	背景颜色	<ul style="list-style-type: none"><li>• 颜色名称，例如red</li><li>• 十六进制值，例如#ff0000</li><li>• rgb 代码，例如rgb(255,0,0)</li></ul>
background-image	背景图片	<ul style="list-style-type: none"><li>• url('URL') 图片路径</li><li>• none 不显示背景图片</li></ul>
background-repeat	设置是否及如何重复背景图像	<ul style="list-style-type: none"><li>• repeat 默认。背景图像将在垂直方向和水平方向重复</li><li>• repeat-x 背景图像将在水平方向重复</li><li>• repeat-y 背景图像将在垂直方向重复</li><li>• no-repeat 背景图像将仅显示一次</li></ul>
background-position	背景图片的位置	<ul style="list-style-type: none"><li>• left、top、top right、center left、center center、center right、bottom left、bottom center、bottom right</li><li>• x% y% 水平位置和垂直位置</li></ul>
background-size	背景图片的尺寸	<ul style="list-style-type: none"><li>• length 背景的高度和宽度，例如80px 60px</li><li>• percentage 以父元素的百分比设置背景图像的高度和宽度，例如50% 50%</li></ul>

# CSS常用属性：Flex弹性布局

在之前要控制HTML元素的布局，会用到padding、margin、postion、float等方法，经过反反复复调试才能实现效果。

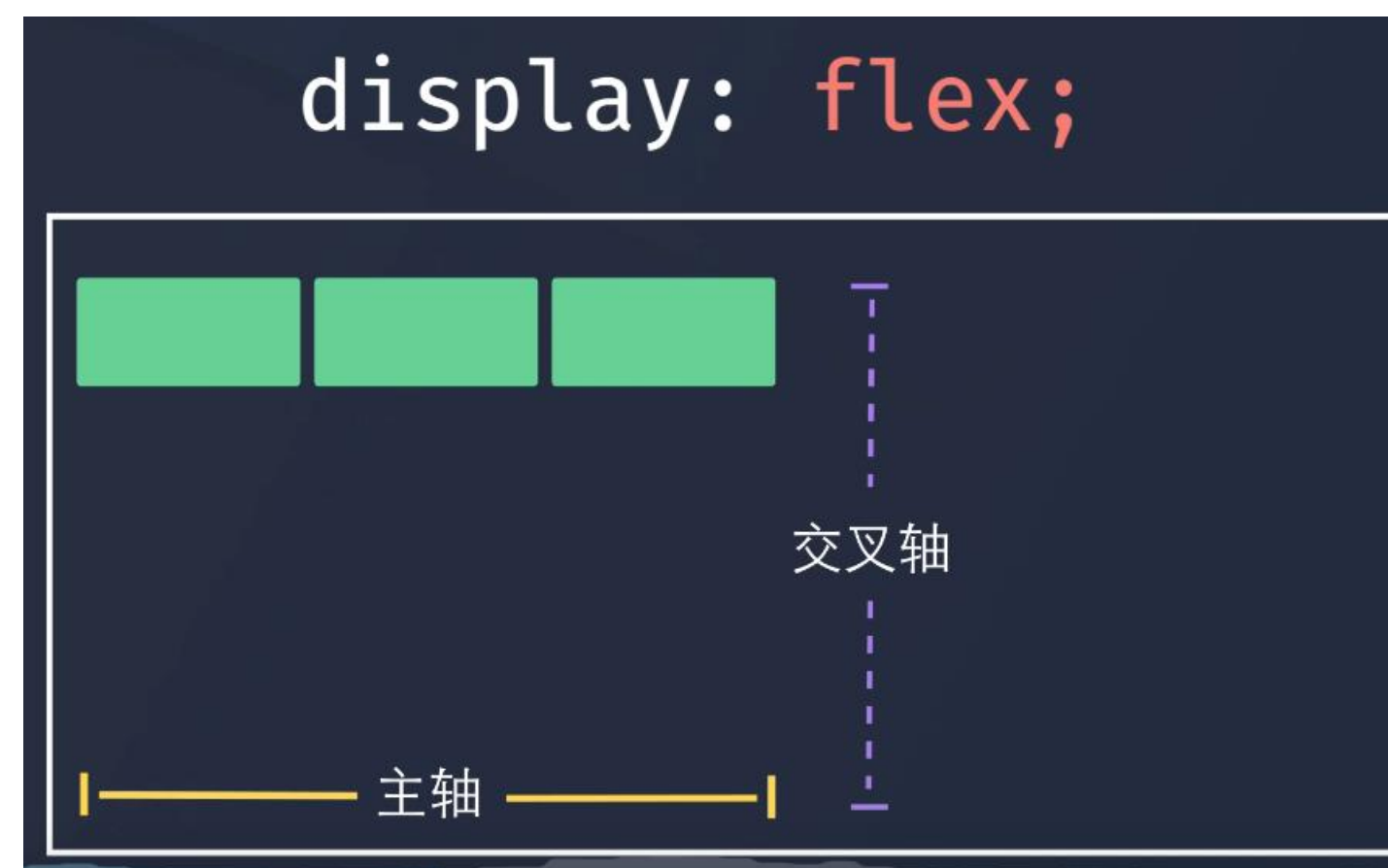
自从Flex弹性布局出现，一切似乎豁然开朗！

启用Flex布局，只需要在外部元素设置display: flex属性。



# CSS常用属性：Flex弹性布局

Flex布局有一个隐式的坐标空间，水平方向有一条主轴，垂直方向有一条交叉轴：





# CSS常用属性：Flex弹性布局

## 改变主轴（横向）的布局：

justify-content:

- flex-end: 右对齐
- center: 居中对齐
- space-evenly: 平分空间
- space-between: 两端对齐

## 改变交叉轴（竖向）的布局：

align-items

- flex-end: 靠下对齐
- center: 居中对齐

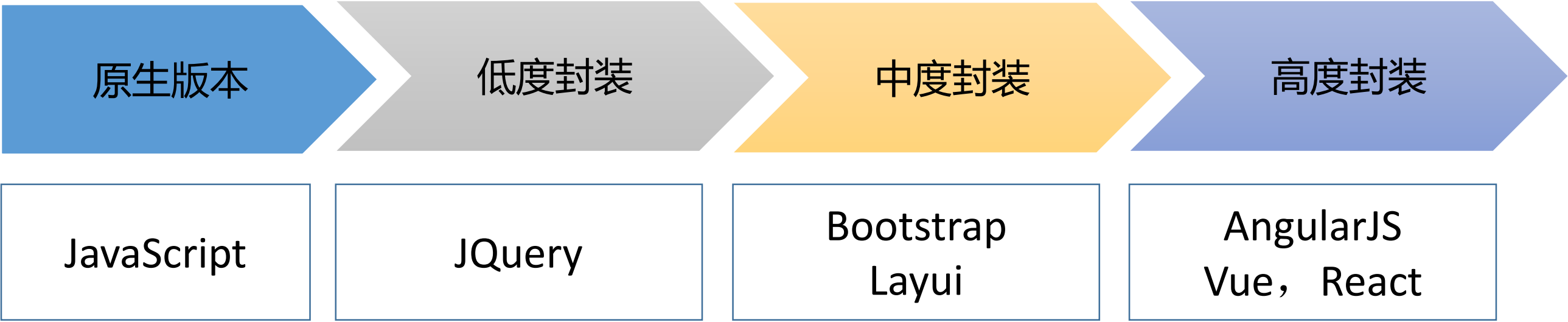
## 调整空间占比（子元素宽度）：

例如第一个元素三分之一，第二个元素占三分之二，第三个元素占三分之一：flex:1;flex2;flex1

# JavaScript 客户端脚本语言

- JavaScript发展史
- JavaScript介绍
- 基本使用
- 事件
- 选择器
- JS操作HTML
- 数据类型：字符串、数组、对象
- 操作符、流程控制
- 函数

# JavaScript发展史



# JavaScript介绍

**JavaScript（简称JS）：** 是一种轻量级客户端脚本语言，通常被直接嵌入 HTML 页面，在浏览器上执行。

## **JavaScript的主要用途：**

- 使网页具有交互性，例如响应用户点击，给用户提供更好的体验
- 处理表单，检验用户输入，并及时反馈提醒
- 浏览器与服务端进行数据通信，主要使用Ajax异步传输
- 在网页中添加标签，添加样式，改变标签属性等

# JavaScript基本使用

## 1、内部方式（内嵌样式），在body标签中使用

```
<script type="text/javascript">
  <!--
  javascript语言
  -->
</script>
```

## 2、外部导入方式（推荐），在head标签中使用

```
<script type="text/javascript" src="my.js"> </script>
```

示例：

```
<script>
  var name = "hello"; // 定义变量
  alert(name); // 警告框方法，浏览器提示消息
  /* alert( "你好" ) */ // 单行与多行注释
</script>
```

**事件：**指的是当HTML中发生某些事件时所调用的方法（处理程序）。

例如点击按钮，点击后做相应操作，例如弹出一句话

示例：

```
<button type="button" onclick="alert('亲，有什么可以帮助你? ')">点我</button>
```

- onclick：是一个常用CSS事件属性，当元素有鼠标点击时触发JS脚本。
- alert()：是一个JS内置函数，在浏览器输出警告框。一般于代码测试，可判断脚本执行位置或者输出变量值。

# 选择器

**想操作元素，必须先找到元素，**主要通过以下三种方法：

- 通过id（常用）
- 通过类名
- 通过标签名

# 选择器

示例：通过id查找元素

```
<button type="button" id="btn">点我</button>
<script>
  var x = document.getElementById( "btn" ); //获取id为btn的元素
  x.onclick = function () { //绑定点击事件
    alert('亲，有什么可以帮助你？ ')
  }
</script>
```

示例：通过标签名

```
<div id="main">
  <p>Hello world! 1</p>
  <p>Hello world! 2</p>
  <p>Hello world! 3</p>
</div>

<script type="text/javascript">
  var x = document.getElementById( "main" ); //获取id为main的元素
  var y = x.getElementsByTagName("p"); // 返回的是一个集合，下标获取
  document.write( "div中的第二段文本是： " + y[1].innerHTML); //向当前文档写入内容
</script>
```



## 插入内容:

```
document.write( "<p>这是JS写入的段落</p>" ); //向文档写入HTML内容
x = document.getElementById( 'demo' ); //获取id为demo的元素
x.innerHTML= "Hello"    //向元素插入HTML内容
```

## 改变标签属性:

```
document.getElementById( "image" ).src= "b.jpg " //修改img标签src属性值
```

## 改变标签样式:

```
x = document.getElementById( "p" ) //获取id为p的元素
x.style.color= "blue"    //字体颜色
```

# 数据类型：字符串

在JS中，数据类型有：字符串、数字、布尔、数组、对象、Null、Undefined

## 字符串处理：

```
var s = "hello world";  
s.length; // 字符串长度  
s[4] //根据索引获取值  
s.replace('h','H'); //替换某个字符  
s.split("分隔符") //分隔为数组  
s.match("w") //找到返回匹配的字符，否则返回null
```

## 字符串拼接：“+”

# 数据类型：数组

**数组：**是一个序列的数据结构。

**定义：**

```
var computer = new Array();
```

或

```
var computer = ["主机","显示器","键盘","鼠标"]
```

**向数组添加元素：**

```
computer[0]="主机";
```

```
computer[1]="显示器";
```

```
computer[2]="键盘";
```

或

```
array.push("鼠标")
```

**通过索引获取元素：**

```
computer[2]
```

# 数据类型：对象

**对象：**是一个具有映射关系的数据结构。用于存储有一定关系的元素。

格式：d = {'key1':value1, 'key2':value2, 'key3':value3}

注意：对象通过key来访问value，因此字典中的key不允许重复。

**定义：**

```
var user = {  
    name:"阿良",  
    sex: "男",  
    age:"30"  
};
```

**通过键查询值：**

```
n = user.name;
```

或

```
sex = user['sex'];
```

**增加或修改：**

```
user.height = "180cm"
```

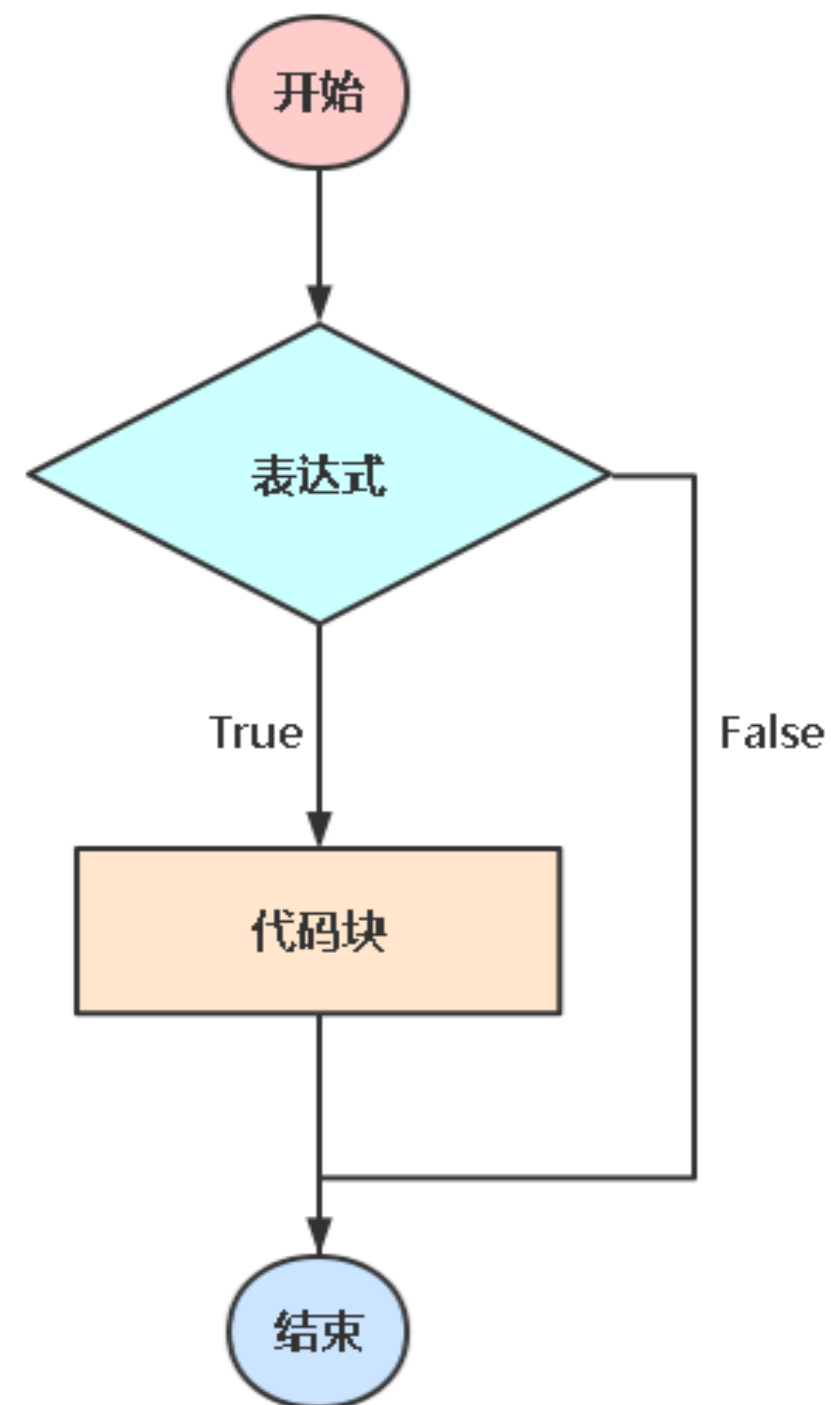
```
user['height'] = "180cm"
```

# 操作符

类型	操作符
比较操作符	<ul style="list-style-type: none"><li>• == 等于</li><li>• != 不等于</li><li>• &gt; 大于</li><li>• &lt; 小于</li><li>• &gt;= 大于等于</li><li>• &lt;= 小于等于</li></ul>
算术操作符	<ul style="list-style-type: none"><li>• + 加法</li><li>• - 减法</li><li>• * 乘法</li><li>• / 除法</li><li>• % 取余</li><li>• ++ 自增, 自动+1</li><li>• -- 自减, 自动-1</li></ul>
逻辑操作符	<ul style="list-style-type: none"><li>• &amp;&amp; 与</li><li>•    或</li><li>• !() 结果取反</li></ul>
赋值运算符	<ul style="list-style-type: none"><li>• = 赋值</li><li>• += 加法赋值</li><li>• -= 减法赋值</li><li>• *= 乘法赋值</li><li>• /= 除法赋值</li><li>• %= 取余赋值</li></ul>

**操作符：**一个特定的符号，用它与其他数据类型连接起来组成一个表达式。常用于条件判断，根据表达式返回True/False采取动作。

# 条件判断



**if条件判断：**判定给定的条件是否满足（True或False），根据判断的结果决定执行的语句。

**语法：**

```
if (表达式) {  
    <代码块>  
} else if (表达式) {  
    <代码块>  
} else {  
    <代码块>  
}
```

# 条件判断

示例：根据用户点击做不同操作

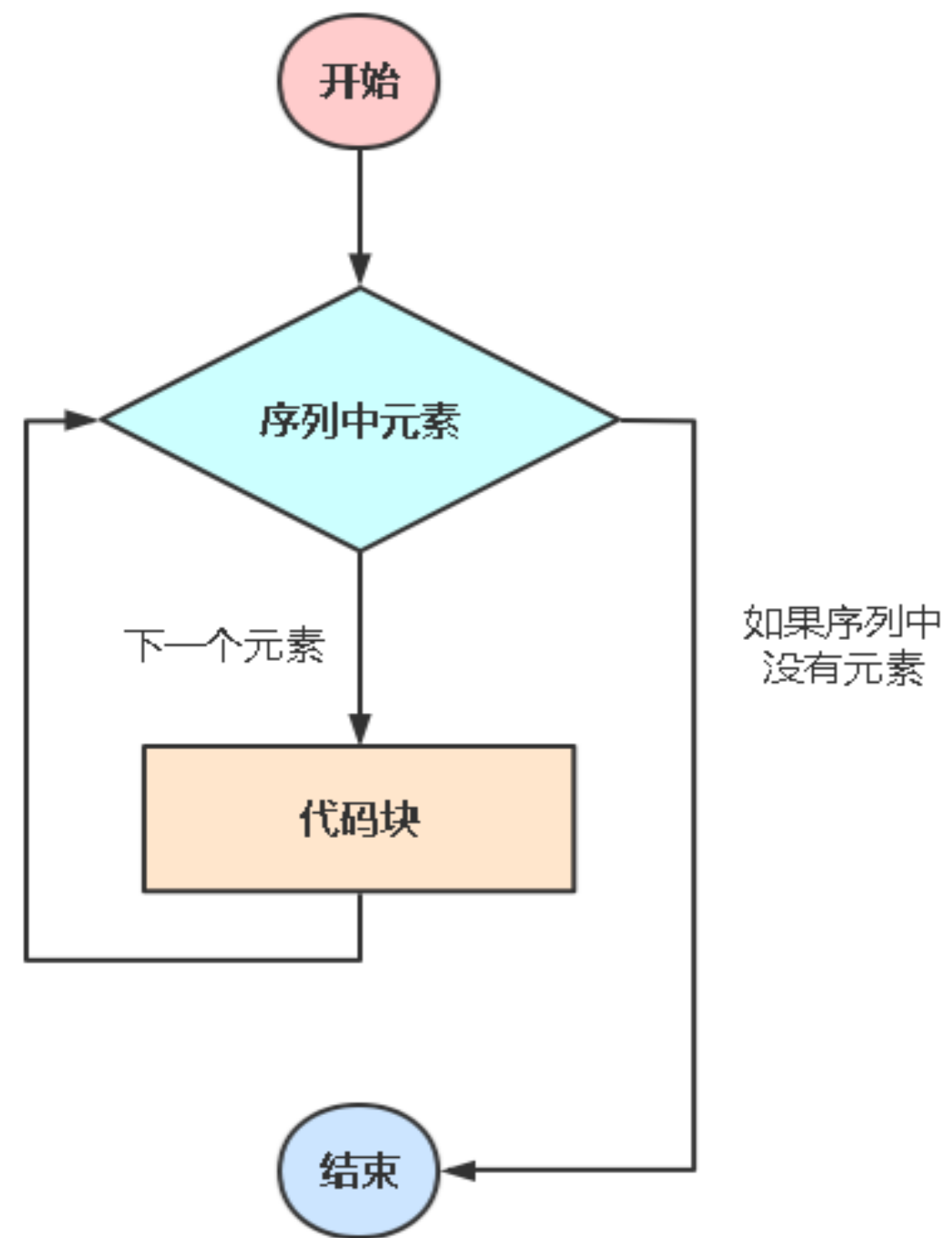
```

<button type="button" onclick="changeImage('on')">开灯</button>
<button type="button" onclick="changeImage('off')">关灯</button>

<script type="text/javascript">
  function changeImage(status) {
    x = document.getElementById('myimage');
    if (status == 'on') {
      x.src = "img/on.jpg";
    } else if (status == 'off') {
      x.src = "img/off.jpg";
    }
  }
</script>
```



# 循环



**for循环：**一般用于遍历数据类型的元素进行处理，例如字符串、数组、对象。

**语法：**

```
for (<变量> in <序列>) {  
    <代码块>  
}
```



# 循环

示例：遍历数组

```
var computer = ["主机","显示器","键盘","鼠标"];
```

方式1:

```
for(i in computer) {  
    console.log(computer[i]) // 使用索引获取值  
}
```

方式2:

```
computer.forEach(function (e) {  
    console.log(e)  
})
```

示例：遍历对象

```
var user = {name:"阿良",sex:"男",age:"30"};
```

方式1:

```
for(let k in user) {  
    console.log(k + ":" + user[k])  
}
```

方式2:

```
Object.keys(user).forEach(function (k) {  
    console.log(k + ":" + user[k])  
})
```

# 函数：定义与调用

**函数：**是指一段可以直接被另一段程序或代码引用的程序或代码。

在编写代码时，常将一些常用的功能模块编写成函数，放在函数库中供公共使用，可减少重复编写程序段和简化代码结构。

语法：

```
function 函数名称(参数1, 参数2, ...) {  
    <代码块>  
    return <表达式>  
}
```

示例：

```
<button type="button" id="btn" onclick="hello()">你好</button>  
  
function hello() {  
    alert("hello world")  
}
```

## 函数参数：接收参数

示例：

```
<button type="button" onclick="myFunc('阿良', '30')">点我</button>
<script type="text/javascript">
  function myFunc(name, age) {
    alert("欢迎" + name + ", 今年" + age);
  }
</script>
```

# 函数：匿名函数与箭头函数

**匿名函数与箭头函数：**没有名字的功能，一般仅用于单个表达式。

示例：

```
<script type="text/javascript">
// 普通函数
function sum1(x,y) {
    return x+y;
}
// 匿名函数
sum2 = function(x,y) {
    return x+y;
}
// 箭头函数，相比匿名函数又简化了很多
sum3 = (x,y) => {
    return x+y;
}
console.log(sum1(1,2))
console.log(sum2(3,4))
console.log(sum3(5,6))
</script>
```

# window对象：location属性处理URL

示例：刷新按钮

```
<button type="button" onclick="location.reload()">刷新当前页面</button>  
<button type="button" onclick="location.href=location.href">重新请求当前页面</button>  
<button type="button" onclick="location.href='http://www.baidu.com'">请求别的页面</button>
```

# JavaScript库：jQuery，简化编程

- jQuery介绍
- 基本使用
- 选择器
- 操作HTML
- Ajax 前后端数据交互

# | jQuery介绍

**jQuery 是一个 JavaScript 库。**极大地简化了 JavaScript 编程，例如JS原生代码几十行实现的功能，jQuery可能一两行就可以实现，因此得到前端程序猿广泛应用。

官方网站： <https://jquery.com>

发展至今，主要有三个大版本：

- 1.x：常用版本
- 2.x, 3.x：除非特殊要求，一般用的少

# | jQuery基本使用

下载地址：

<https://cdn.bootcdn.net/ajax/libs/jquery/1.12.4/jquery.min.js>

jQuery代码编写位置与JS位置一样，但需要先head标签里引入jquery.min.js文件：

```
<head>
  <script type="text/javascript" src="https://cdn.bootcdn.net/ajax/libs/jquery/1.12.4/jquery.min.js"> </script>
</head>
<body>
  <script type="text/javascript">
    // jquery代码
  </script>
</body>
```



# jQuery基本使用

## jQuery语法:

```
<button type="button" id="btn">点我</button>
<script type="text/javascript">
    $("#btn").click(function () {
        alert('亲, 有什么可以帮助你? ')
    })
</script>
```

## 基础语法是: **\$(selector).action()**

- \$: 代表jQuery本身
- (selector): 选择器, 查找HTML元素
- action(): 对元素的操作

## JS语法:

```
<button type="button" id="btn">点我</button>
<script type="text/javascript">
    var x = document.getElementById("btn")
    x.onclick = function () {
        alert('亲, 有什么可以帮助你? ')
    }
</script>
```

# 选择器

名称	语法	示例
标签选择器	element	\$( "h2" ) 选取所有h2元素
类选择器	.class	\$( ".title " ) 选取所有class为title的元素
ID选择器	#id	\$( "#title" ) 选取id为title的元素
并集选择器	selector1,selector2,...	\$( "div,p,.title" ) 选取所有div、 p和拥有class为title的元素
属性选择器		\$( "input[name]= 'username' " ) 选取input标签名为username的元素 \$( "[href= '#' ]" ) 选取href值等于 "#" 的元素

# | jQuery操作HTML

## 隐藏和显示元素：

- hide() ： 隐藏某个元素
- show() ： 显示某个元素
- toggle() ： hide()和show()方法之间切换

## 示例：

```
<p id="demo">这是一个段落。 </p>
<button id="hide" type="button">隐藏</button>
<button id="show" type="button">显示</button>
<button id="toggle" type="button">切换</button>

<script type="text/javascript">
  $("#hide").click(function () {
    $("p").hide();
  });
  $("#show").click(function () {
    $("p").show();
  });
  $("#toggle").click(function () {
    $("p").toggle();
  })
</script>
```

# jQuery操作HTML

## 获取与设置内容：

- text() 设置或返回所选元素的文本内容
- html() 设置或返回所选元素的HTML内容
- val() 设置或返回表单字段的值

示例：

```
<p id="txt">这是一个<b>段落</b>。</p>
<button type="button" id="btn1">显示文本</button>
<button type="button" id="btn2">显示HTML</button>
<p id="demo"></p>

<script type="text/javascript">
    $("#btn1").click(function () {
        x = $("#txt").text();
        $("#demo").text(x).css("color","red") //不会解析b标签
    });
    $("#btn2").click(function () {
        x = $( "#txt" ).html(); //获取
        $( "#demo" ).html(x).css("color","red") //会解析b标签, .html()设置
    })
</script>
```

示例：

```
<h1>欢迎访问运维管理系统</h1>
用户名： <input type="text" id="uname" name="username"> <br>
密码： <input type="text" id="pwd" name="password"> <br>
<button type="button" id="btn">显示输入内容</button>
<p id="demo"></p>

<script type="text/javascript">
    $("#btn").click(function () {
        x = $("#uname").val();
        y = $("#pwd").val();
        $( "#demo" ).text(x + ' , ' + y).css("color","red")
    })
</script>
```

# | jQuery操作HTML

## 设置CSS样式:

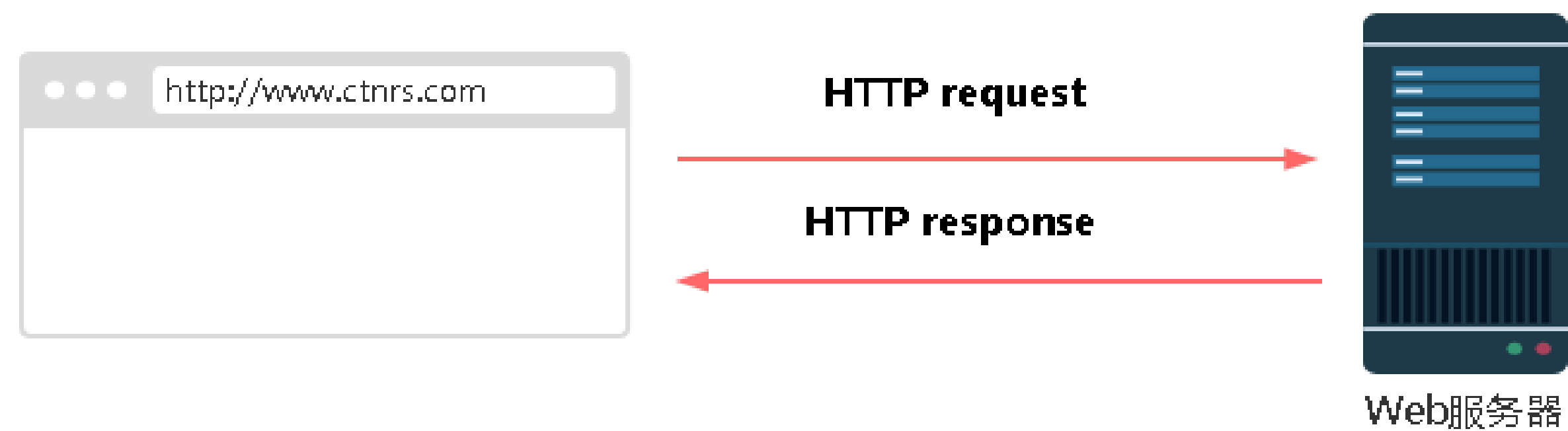
- `css()` 设置或返回样式属性（键值）
- `addClass()` 向被选元素添加一个或多个类样式
- `removeClass()` 从被选元素中删除一个或多个类样式
- `toggleClass()` 对被选元素进行添加/删除类样式的切换操作

## 示例:

```
<div id="demo">
  <p>这是一个段落</p>
</div>
<button id="btn">添加样式</button>

<script>
  $("#btn").click(function () {
    $("#demo p").css("color", "red")
    // $("#demo p").css({"color":"red","font-size": "30px"})
    // $("#demo").addClass("cls")
    // $("#demo").removeClass("cls")
  })
</script>
```

# | jQuery Ajax前后端数据交互



浏览器访问网站一个页面时，Web服务器处理完后会以消息体方式返回浏览器，浏览器自动解析HTML内容。如果局部有新数据需要更新，需要刷新浏览器重新发起页面请求获取最新数据，如果每次都是通过刷新解决这个问题，势必会给服务器造成负载加重，页面加载缓慢。

# | jQuery Ajax前后端数据交互

Ajax (Asynchronous JavaScript And XML, 异步JavaScript和XML) , AJAX 是一种在无需重新加载整个网页的情况下, 能够更新部分网页的技术。例如在不刷新页面的情况下查询数据、登录验证等

## 无刷新的好处:

- 减少带宽、服务器负载
- 提高用户体验

# | jQuery Ajax前后端数据交互

jQuery Ajax主要使用\$.ajax()方法实现，用于向服务端发送HTTP请求。

语法：\$.ajax([settings]);

settings 是\$.ajax ( )方法的参数列表，用于配置 Ajax 请求的键值对集合，参数如下：

参数	类型	描述
url	string	发送请求的地址，默认为当前页地址
type	string	请求方式，默认为GET
data	object、 array、 string	发送到服务器的数据
dataType	string	预期服务器返回的数据类型，包括JSON、XML、text、HTML等
contentType	string	发送信息至服务器时内容编码类型。默认值: "application/x-www-form-urlencoded"。
timeout	number	设置请求超时时间
headers	object	设置请求头信息
async	Boolean	默认true，所有请求均为异步请求。设置false发送同步请求



# jQuery Ajax前后端数据交互

```
<div id='demo'>
  <p id='notice' style="color: red;"> </p>
  <h1>用户列表</h1>
  <ul> </ul>
</div>

<script type="text/javascript">
$.ajax({
  type: "GET",
  url: "http://www.aliangedu.cn/test-table/user.json",
  success: function (result) { // result是API返回的JSON数据
    if(result.code == 200) {
      for (i in result.data) {
        $('#demo ul').append("<li>" + result.data[i]['username'] + "</li>"); // 将li标签追加到ul标签
      }
    } else {
      $('#notice').text('数据获取失败！ ')
    }
  },
  error: function () {
    $('#notice').text('连接服务器失败，请稍后再试！ ')
  }
})
</script>
```

Ajax使用案例

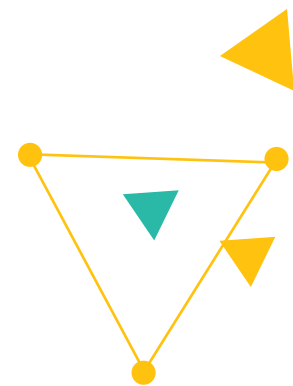
# | jQuery Ajax前后端数据交互

jQuery Ajax 是 jQuery 库中用于向服务器异步请求数据的方法。它使用 XMLHttpRequest 对象，但通过 jQuery 库的封装，使得使用更加简单。jQuery Ajax 支持多种数据格式，如 JSON、XML、HTML 等。在发送请求时，可以通过设置各种选项来控制请求的行为，如超时时间、数据类型、回调函数等。回调函数是 jQuery Ajax 中非常重要的概念，它允许我们在请求完成后执行特定的操作。回调函数可以接收请求返回的数据，并对其进行处理。jQuery Ajax 还支持链式调用，使得代码更加简洁和易读。总的来说，jQuery Ajax 为前端开发人员提供了一种方便、灵活的方式来与服务器进行数据交互。

回调函数：参数引用一个函数，并将数据作为参数传递给该函数。

参数	函数格式	描述
beforeSend	function(jqXHR,object)	发送请求前调用的函数，例如添加自定义 HTTP 头
success	function(data, String textStatus,jqXHR)	请求成功后调用的函数，参数data：可选，由服务器返回的数据（JSON）
error	function(jqXHR,String textStatus,errorThrown)	请求失败时调用的函数
complete	function(jqXHR, String textStatus)	请求完成后（无论成功还是失败）调用的函数

jqXHR：一个XMLHttpRequest对象



# 谢谢

---



阿良个人微信



DevOps技术栈公众号

阿良教育: [www.aliangedu.cn](http://www.aliangedu.cn)

