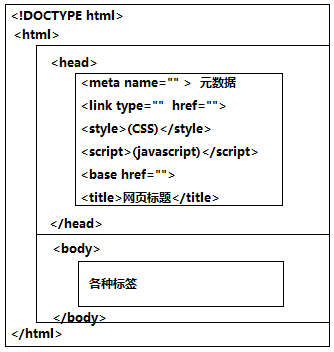
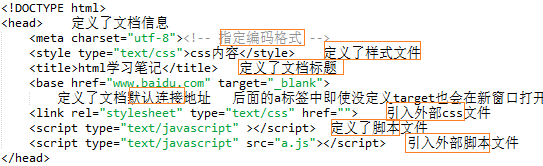
# 前端与移动 笔记

## HTML

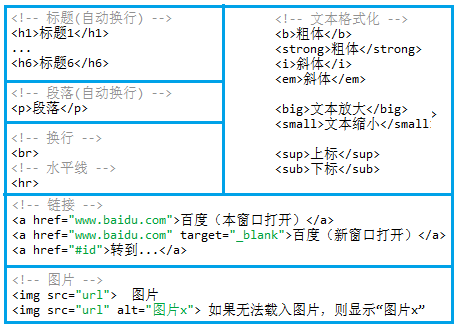
### 第一章 html 文档结构



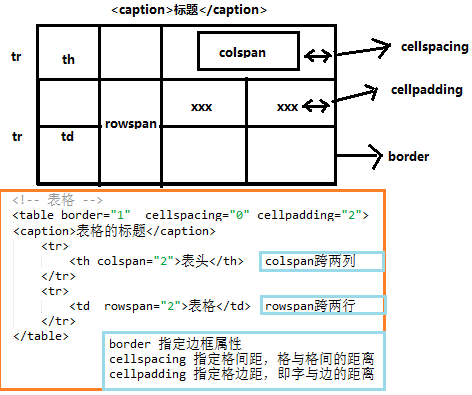
### 第二章 head 头部



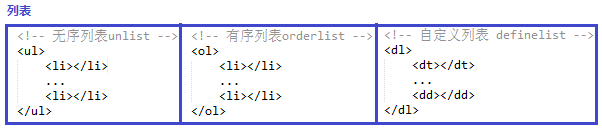
### 第三章 常用标签



### 第四章 表格



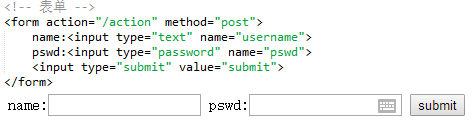
### 第五章 列表

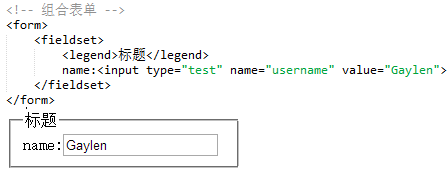


### 第六章 容器



### 第七章 表单



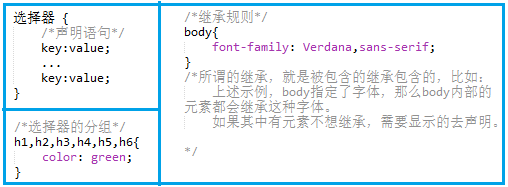




## CSS

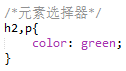
### 第一章 CSS基础

#### 1.1 CSS基础语法

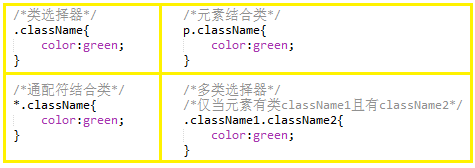


### 第二章 CSS 选择器

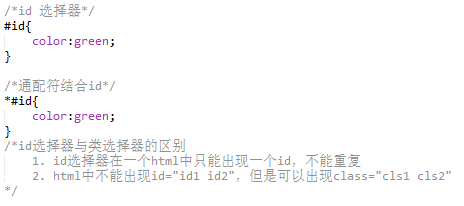
#### 2.1 元素选择器



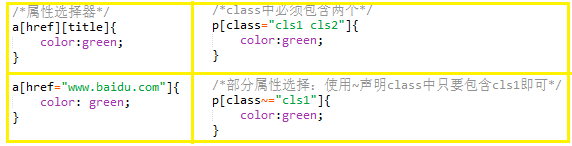
#### 2.2 类选择器

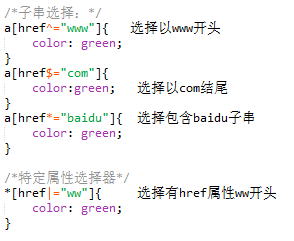


#### 2.3 id选择器

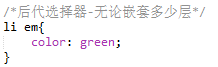


#### 2.4 属性选择器

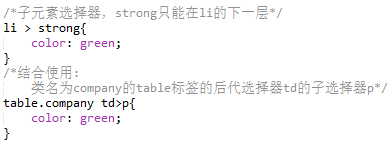




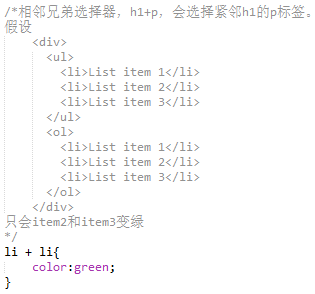
#### 2.5 后代选择器



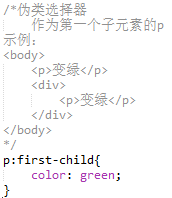
#### 2.6 子元素选择器



#### 2.7 相邻兄弟选择器



#### 2.8 伪类选择器





### 第三章 CSS 高级

## 第三节 Bootstrap

## Javascript

### 第一章 JavaScript基础

#### 1.1 基础语法

JavaScript 是完全的面向对象脚本语言。

JavaScript 中一切都是对象，因此无论是数字还是字符串都有对应的方法进行操作。

1. 注释

// 单行注释

/\*

多行注释

\*/

2. 变量

var x =3;

var y =3.14;

var z ="hello";

var value; //value = undefined

3. 数据类型

// JavaScript拥有数字，字符串，布尔

// 数组，对象，Null，Undefined这几种数据类型

1. 数字 - JavaScript 只有一种数字类型。不区分整数和小数

var num = new Number(20);

var x = 34;

var y =3.14;

var z = 123e5; //12300000

2. 字符串

var str = new String("JavaScript");

var str = "JavaScript";

var sen = "my name is 'Gaylen'"; //引号嵌套

3. 布尔

var bool = new Boolean(false);

var bool = true;

var x =false;

4. 数组

var arr = new Array();

var arr = new Array(5);

var arr = new Array(1,2,3,4);

var arr =

5. 对象

var obj = new Object();

obj.name="mike";

var person={

name:"Bill",

age:23,

method:function{}

}

var obj = new Object();

6. null //清除

var value = null;

7. undefined //变量中不含有值

var value ;

#### 1.2 流程控制

运算符：

|  |  |  |  |
| --- | --- | --- | --- |
| 算术运算符 | 赋值运算符 | 比较运算符 | 逻辑运算符 |
| + 加 | = | == 等于 | && 与 |
| - 减 | += | === 全等（值和类型） | || 或 |
| \* 乘 | -= | != 不等于 | ! 非 |
| / 除 | \*= | > 大于 | (condition)?value1:value2;  三目运算符 |
| % 取余 | /= | >= 大于等于 |  |
| ++ 自增 | %= | < 小于 |  |
| -- 自减 |  | <= 小于等于 |  |

流程控制：

// if语句

//1. if语句

if(expression){

//code ...

}

//2. if-else语句

if(expression){

//code ...

}else{

//code ...

}

//3. if - else if - else语句

if(expression1){

//code ...

}else if(expression2){

//code ...

}else{

//code ...

}

//switch 语句

switch(value){

case value1:

// code ...

break;

case value2:

//code ...

break;

default:

///code ...

}

//for 语句

//1. for语句

for(var i ;i<10; i++){

//code ...

}

//2. for - in 语句

var person{

name:"mike",

age:23

};

for(var key in person){

// person[key]

}

//while 循环

//1. while 循环

var count = 0;

while(count<10){

//code

count++;

}

//2. do - while 循环

var count = 0;

do{

//code ...

count++;

}while(count<10);

//break 与 continue

break语句用于跳出整个循环

continue语句用于跳出本次循环，执行下一次循环

#### 1.3 函数

//JavaScript函数

function methodName(){

//code ...

}

//示例：

function myFunc(a,b){

return a+b;

}

#### 1.4 异常

//1. try - catch 语句

try{

// code ...

}catch(err){

// code ...

}

//2. try - catch - throw

try{

//code

if() throw "empty";

}catch(err){

// code ...

}

### 第二章 JavaScript DOM

#### 2.1 获取HTML中的元素

1. 通过id找到HTML元素

var x = document.getElementById("idName");

2. 通过标签名找到HTML元素

var x = document.getElementByTagName("h1");

3. 通过类名找到HTML元素，需要自己定义函数

4. 通过name找到HTML元素，因为name不唯一，因此返回的是元素数组

var x = document.getElementByName();

#### 2.2 改变HTML元素的内容（innerHTML）

// 改变HTML元素的内容

document.getElementById(id).innerHTML=”new HTML”;

//改变HTML元素的属性

document.getElementById(id).attributeName=“new value“；

#### 2.3 改变HTML 元素的样式(CSS)

//改变HTML元素的样式

document.getElementById(id).style.propertyName=”new style”;

//示例：

<button type="button" onclick="document.getElementById('id1').style.color='red'">

#### 2.4 对HTML DOM事件作出反应

#### 2.5 添加/删除HTML 元素

1. 添加元素

/\*\*

\* 添加元素分为两步，创建新元素，追加到已有元素

\*/

var para = document.createElement("input");

//设置属性1

para.value = "这是一个按钮";

//设置属性2

para.setAttribute("type","button");

body.appendChild(para);

//或者

document.getElementByTagName("body").appendChild(pare);

2. 删除元素

/\*\*

\*JavaScript删除元素，分为两步

\* 1. 获取父节点

\* 2. 删除子节点

\*/

var parent = document.getElementById();

var chile = document.getElementById();

parent.removeChild(child);

### 第三章 JavaScript 高级

#### 3.1 面向对象

1. 创建对象

//创建对象

var person = new Object();

person.name = "mike";

...

person.methodName = function{

//code ...

}

//创建直接的对象

var person = {name:"mike",age:23};

//使用函数构造对象

function person(firstname,lastname,age,eyecolor)

{

this.firstname=firstname;

this.lastname=lastname;

this.age=age;

this.eyecolor=eyecolor;

this.methodName = function(parameter){

// code …

};

}

myFather=new person("Bill","Gates",56,"blue");

2.

#### 3.2 常用对象

1. 数字

2. 字符串

3. 数组

4. 日期

5. Boolean

6. Math对象

7. RegExp 正则表达式

#### 3.3 JSON

JSON是指JavaScript的对象表示法

JSON 独立于编程语言

{

employees": [

{ "firstName":"Bill" , "lastName":"Gates" }, //Object1

{ "firstName":"George" , "lastName":"Bush" }, //Object2

{ "firstName":"Thomas" , "lastName":"Carter" } //Object3

]

}

## JQuery

### 第一章 JQuery 基本语法

#### 1.1 Jquery 基础语法

$(document).ready(function(){

$ (selector).action{

//code …

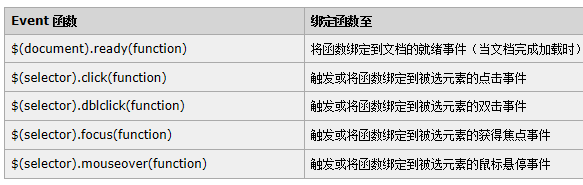
}

});

#### 1.2 Jquery 选择器—HTML的获取



#### 1.3 Jquery 事件



### 第二章 JQuery 操作HTML

#### 2.1 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内容（包括 HTML 标记）

•val() - 设置或返回表单字段的值

$("#btn1").click(function(){

$("#test1").text("Hello world!");

});

$("#btn2").click(function(){

$("#test2").html("<b>Hello world!</b>");

$("#test2").html(function(param){

});

});

$("#btn3").click(function(){

$("#test3").val("Dolly Duck");

});

#### 2.2 设置属性

$("button").click(function(){

$("#w3s").attr(

"href",<http://www.w3school.com.cn/jquery>

"href",<http://www.w3school.com.cn/jquery>

);

$("#w3s").attr("href",function(param){

// code …

});

});

#### 2.3 删除元素或内容

//remove 删除被选元素以及子元素

// empty 从被选元素中删除子元素

$("#div1").remove();

$("#div1").empty();

//过滤被删除元素，如删除class = italic的所有p元素

$("p").remove(".italic");

#### 2.4 操作CSS

•addClass() - 向被选元素添加一个或多个类

$("button").click(function(){

$("#div1").addClass("important blue");

});

•removeClass() - 从被选元素删除一个或多个类

$("button").click(function(){

$("h1,h2,p").removeClass("blue");

});

•toggleClass() - 对被选元素进行添加/删除类的切换操作

$("button").click(function(){

$("h1,h2,p").toggleClass("blue");

});

•css() - 设置或返回样式属性

//设置样式属性

$("p").css("background-color","yellow");

//返回样式属性

$("p").css("background-color");

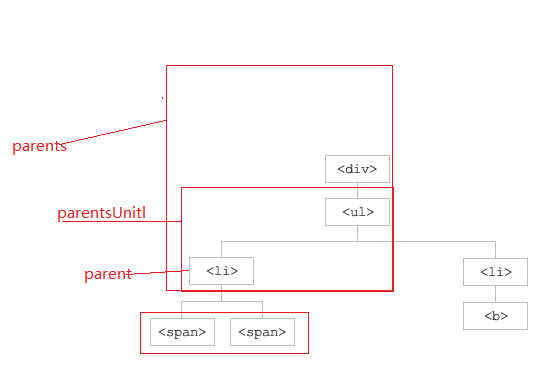
### 第三章 JQuery 实现遍历HTML

#### 3.1 祖先

•parent() 返回span的直接父元素

•parents() 返回span的所有祖先

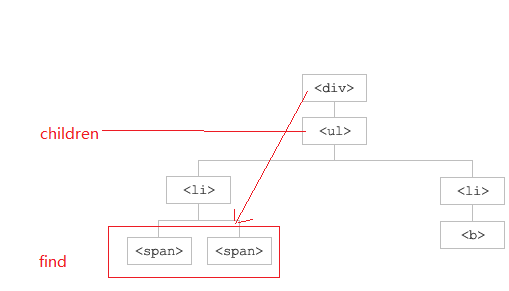
•parentsUntil() 返回span与div之间的祖先



#### 3.2 后代

•children() 返回div的直接子元素

•find() 查找div所有后代中的span元素



#### 3.3 同胞

•siblings() h2的所有同胞

•next() h2的下一个同胞

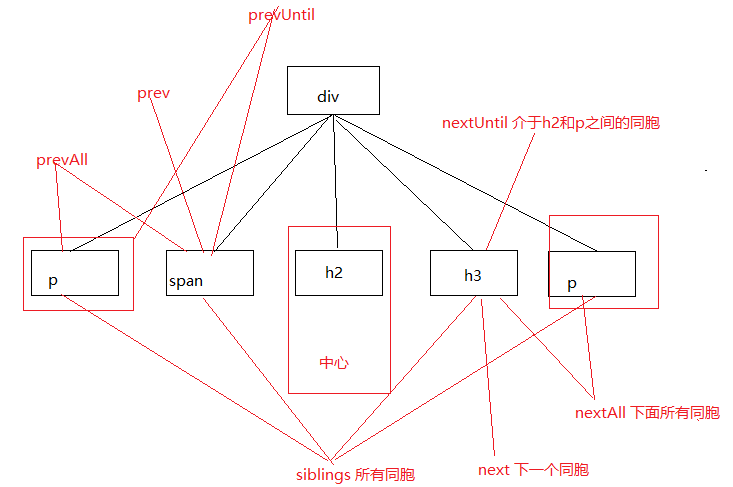
•nextAll() h2后面所有同胞

•nextUntil() h2和p之间的同胞

•prev() h2前一个同胞

•prevAll() h2前面所有同胞

•prevUntil() h2和前面p之间的同胞



#### 3.4 过滤

//选取首个div元素内部的第一个p元素

$(document).ready(function(){

$("div p").first();

});

//选取最后一个div的最后一个p元素

$(document).ready(function(){

$("div p").last();

});

//选取第二个p元素。索引号从0开始

$(document).ready(function(){

$("p").eq(1);

});

### 第四章 JQuery 中的Ajax

### 第五章 JQuery 中的特效

## 第六节 Ajax

### 第一章 Ajax简介

Ajax 是一步的JavaScript和XML。

Ajax并不是新的编程语言。

Ajax是页面与服务器进行数据交换的艺术。不加载整个页面的情况下进行局部刷新。

### 第二章 Ajax 基础

##### 7.2.1 创建XHR对象

创建XMLHttpRequest对象，并且兼容IE5，IE6.

var xmlhttp;

if (window.XMLHttpRequest)

{ // code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{ // code for IE6, IE5

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

##### 7.2.2 向服务器发送请求

Get请求示例：

var xmlhttp;

if (window.XMLHttpRequest)

{// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{// code for IE6, IE5

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

}

xmlhttp.open("GET","../ajax/demo\_get.asp.htm",true);

xmlhttp.send();

##### 7.2.3 获得来自服务器的响应



示例：

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

##### 7.2.4 onreadystatechange 事件



示例：

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

}

### 第三章 Ajax 进阶

### 第四章 Ajax 高级

## 第七节 Angular2

## 第八节 Node.js