千锋教育™
WWW.QFEDU.COM

第3章 DOM入门之获取元素

- 获取元素
- 元素样式操作
- 获取集合
- 实际运用

目录

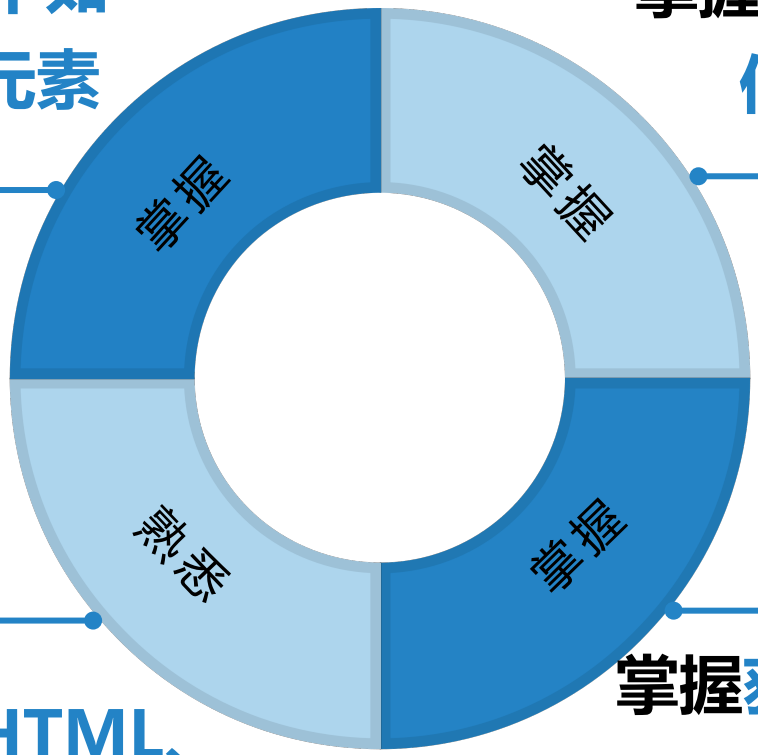
千锋教育
WWW.QFEDU.COM

学习目标

千锋教育 WWW.QFEDU.COM

1 掌握DOM中如何获取元素

2 掌握DOM中如何获取集合



3 掌握获取元素样式操作，行间与非行间

4 熟悉JS跟HTML、CSS之间的通信，并实际运用

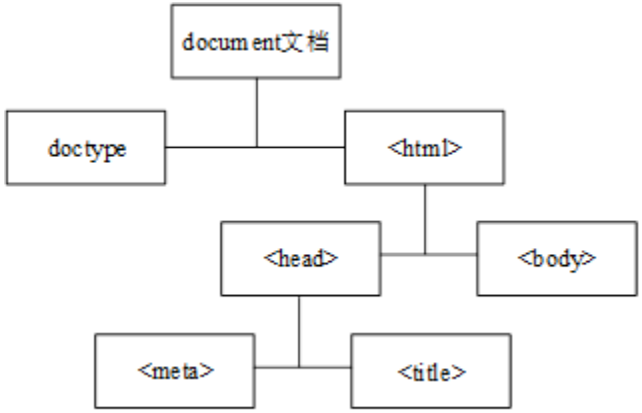
3.1 获取元素

返回目录

3.1 获取元素

3.1.1 document文档

- 如何通过JavaScript获取到网页中的元素？首先需要了解document文档，document文档指整个页面的根对象（最外层对象），通过document文档获取页面中的具体HTML元素。注意document文档的类型为对象类型，即typeof document返回object值，因此，通常document文档亦称document对象。document文档页面的最外层如图所示。



3.1 获取元素

3.1.1 document文档

- document对象常见属性如表所示。

□□□	□□□
doctype	□□□□□□□□□□
documentElement	□□□□□<html>□□
head	□□□□□<head>□□
body	□□□□□<body>□□
title	□□□□□<title>□□

- 接下来通过案例演示document对象的常见属性，具体如例所示。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Hello JS</title>
</head>
<body>
<script>
    console.log(document.doctype);            //  <!doctype html>
    console.log(document.documentElement);    //  <html>
    console.log(document.head);               //  <head>
    console.log(document.body);               //  <body>
    console.log(document.title);              //  <title>
</script>
</body>
</html>
```

3.1 获取元素

3.1.1 document文档

- 在例中，分别输出五句document的属性，分别表示document.doctype（文档头信息）、document.documentELement（<html>标签）、document.head（<head>标签）、document.body（<body>标签）、document.title（<title>标签）。可以看到控制台中输出了对应标签的格式及其子内容。
- 接下来修改<title>的内容，查看发生的变化。具体如例所示。

```
1  <script>
2      document.title = 'hello DOM';          //设置title为hello DOM
3  </script>
```

- 运行结果如图所示。

3.1 获取元素

千锋教育 WWW.QFEDU.COM

3.1.1 document文档

- 在例中，对当前页面的<title>标签内的文本内容进行了重新的定义，通过“document.title = ‘hello DOM’;”进行设置。可以看到示图中的title展示出来的效果。在后面的章节中还会讲解document对象一些常用的属性和方法等复杂内容，这里稍作了解即可。

千锋教育 WWW.QFEDU.COM

3.1.2 getElementById()方法

- 在学习DOM获取HTML中指定的标签元素前，先来思考如何实现CSS样式，可以通过给元素添加id属性，然后通过id选择器进行样式的设置，具体示例代码如下：

```
<style>
    #div1{ width:200px; height:200px; background : red;}
</style>
<body>
    <div id="div1">hello</div>
</body>
```

- DOM操作也可以通过添加id的方式获取和设置指定的HTML元素，通过document对象的getElementById()方法实现。通过单词拼写可直观地发现getElementById()方法即用id的方式获取元素，具体示例代码如下：

```
<body>
    <div id="div1">hello</div>
</body>
<script>
    var div = document.getElementById('div1');
</script>
```

3.1 获取元素

千锋教育 WWW.QFEDU.COM

3.1.2 getElementById()方法

- 把id值通过字符串形式放到此方法的小括号中，即可以获取div元素，通过查看类型，可以发现字符串也属于对象类型，所以说对象是一个极其强大的数据类型，表示各种值组成的集合。
- 注意，在获取元素时，一定要保证<html>标签加载完毕后再获取，否则可能获取不到目的元素。

3.1.3 元素属性操作

- 只获取元素本身并不能对其属性进行操作，可以通过点的方式来获取元素身上的属性，具体示例代码如下：

```
<body>
    <div id="div1" class="box" title="Hello JS">hello</div>
</body>
<script>
    var div = document.getElementById('div1');
    console.log(div.id);              //div1
    console.log(div.className);       //box
    console.log(div.title);           //Hello JS
</script>
```

- 这里需要注意class属性，因为class属于JavaScript中的保留字，所以不能通过class的方式获取，需要通过className的语法形式获取。与class属性类似的还有for属性，需要通过htmlFor的方式获取。
- 除了获取外，也可以对元素的属性进行设置。

3.2 元素样式操作

返回目录

3.2 元素样式操作

3.2.1 行间样式

- 在前面的小节中，讲解了DOM操作HTML元素的属性，同样对HTML元素的style属性也能够获取和设置，接下来通过案例演示调试获取style属性中的样式，具体如例所示。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Hello JS</title>
</head>
<body>
    <div id="div1" style="width:100px; height:50px;
        background:red;">hello</div>
<script>
    var div = document.getElementById('div1');
      console.log(div.style);                // 获取 style 属性
      console.log(div.style.width);          // 获取 style 下的 width
      console.log(div.style.height);         // 获取 style 下的 height
      console.log(div.style.background);     // 获取 style 下的 background
</script>
</body>
</html>
```

3.2 元素样式操作

3.2.1 行间样式

- 可以发现，div.style包含很多默认样式值，当不设置样式时，默认值都为空字符串。也可以设置style中的样式，接下来演示设置style属性中的样式。具体如例所示。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Hello JS</title>
</head>
<body>
    <div id="div1" style="width:100px; height:50px;
        background:red;">hello</div>
<script>
    var div = document.getElementById('div1');
     div.style.width = '200px';            // 设置 width 为 200px
     div.style.height = '100px';           // 设置 height 为 100px
     div.style.background = 'blue';        // 设置 background 为 blue
</script>
</body>
</html>
```

千锋教育 WWW.QFEDU.COM

3.2.2 cssText

- 在style属性下操作多组样式，需要一行行设置，操作起来比较烦琐。DOM提供了cssText属性，可用来一次性设置多组CSS样式，注意cssText属性是在style对象的下面，具体示例代码如下。

```
<body>
    <div id="div1" style="width:100px; height:50px;
      background:red;">hello</div>
</body>
<script>
   var div = document.getElementById('div1');
   div.style.cssText = 'width:200px; height:100px; background:blue;';
</script>
```

- 从上例可以发现，cssText中样式的写法与CSS样式的写法类似，通过冒号和分号进行操作，设置简单明了。注意，当出现多个cssText属性时，后面的cssText会覆盖前面整个cssText中的样式。

3.2.3 非行间样式

- 3.2.2节中获取和设置的样式，都是通过style属性完成，操作都是在行间完成，对于非行间样式并不可用，具体示例代码如下：

```
<style>
    #div1{ width:200px; height:200px; background:red; }
</style>
<body>
    <div id="div1">hello</div>
</body>
<script>
    var div = document.getElementById('div1');
    console.log(div.style.width);          // ''
    console.log(div.style.height);         // ''
    console.log(div.style.background);     // ''
</script>
```

- style属性只能获取行间样式，并不能获取<style>标签中的非行间样式，因此，返回结果并没有值，而是返回空字符串。可以发现，这对于开发并不是特别方便，如需要获取非行间中的样式，DOM该如何操作？

千锋教育 WWW.QFEDU.COM

3.2.3 非行间样式

- window.getComputedStyle()是在标准规范下提供的获取最终样式的方法，最终样式即包括行间样式和非行间样式，因此可通过这种方法来完成需求。其语法格式如下。

```
window.getComputedStyle(元素).样式;
```

- window对象是可选的，表示窗口对象。
- 注意，在JavaScript语法中不支持横杠，因此不能在语法中出现类似background-color的写法。当有此需求时，JavaScript中通过去掉横杠，并且设置横杠后的第一个字母为大写的方式来实现，类似于驼峰的命名方式。具体示例代码如下：

```
console.log(window.getComputedStyle(div).backgroundColor);          //rgb(255,  0,
0)
```

- 上例返回RGB方式的颜色，属于内部转化的形式，这里不作深究。
- 因为获取的是最终样式，所以假设既有行间样式，也有非行间样式，最终样式即是优先级高的样式。

3.2 元素样式操作

3.2.3 非行间样式

- window.getComputedStyle()方法是在标准规范下提供的操作，对于之前一些旧版本的浏览器，可能并不支持，因此，可以利用currentStyle对象实现兼容处理。其语法格式如下。

```
元素.currentStyle.样式;
```

- 无论是getComputedStyle()方法还是currentStyle()方法，都是只能获取到非行间样式，并不能设置非行间样式。
- 那么，如何在JavaScript中设置非行间样式的呢？一般情况下这种需求非常少见，但JavaScript是有这个能力的，只是对此功能进行了解即可，不必深究。

3.3 获取集合

返回目录

3.3 获取集合

3.3.1 getElementsByTagName()方法

- getElementById()方法可以获取到指定id的HTML元素。那么，如何获取一个HTML元素集合呢？例如，一组<li>标签。可以通过getElementsByTagName()方法来实现，该方法通过指定HTML标签名的方式来获取。其语法格式如下。

```
document/祖先元素.getElementsByTagName('标签名');
```

- 注意，一组标签即可以在整个document文档下获取，也可以在指定的祖先元素下获取。此方式与getElementById()方法不一样，getElementById()方法只能在document文档下获取，因为id属性具有唯一性，不存在包含关系。
- 接下来通过案例演示获取document下的所有li，具体如书中例3-10所示。
- 接下来把这个集合在控制台中输出，可以看到里面对应的就是所有的li列表，并且存在length属性，表示集合的长度。

3.3 获取集合

千锋教育 WWW.QFEDU.COM

3.3.1 getElementsByTagName()方法

- 接下来通过案例演示获取ul下的所有li，具体如书中 例3-11所示。
- 获取到元素集合后，如何去操作？在JavaScript语法中不能直接去操作整体集合，必须分别操作集合的每一项元素。可以通过“[]+下标”的方式获取指定集合中的某一项元素，下标从0开始。例如，li[0]表示li集合中的第一个元素，li[1]表示li集合中的第二个元素。接下来通过案例演示，具体如书中例3-12所示。
- 可以通过集合对象下面的length属性获得集合内元素的个数，用来表示集合的长度。接下来通过案例演示，具体如书中例3-13所示。
- 获取到集合的长度，再配合循环语句，就可以对一组HTML元素集合操作，接下来通过案例演示将集合元素背景色变红，具体如书中例3-14所示。

3.3 获取集合

3.3.2 getElementsByClassName()方法

- 除了可以通过标签的方式获取集合，还可以通过class属性的方式获取集合。getElementsByClassName()方法与getElementsByTagName()方法使用方式类似，只是小括号内为class样式名。其语法格式如下。

```
document/祖先元素.getElementsByClassName('样式名');
```

3.3 获取集合

3.3.3 类似CSS方式获取元素

- 在JavaScript中提供了类似于CSS方式获取元素的方法，即querySelector()和querySelectorAll()两个方法。
- querySelector()方法用来获取单一元素，而querySelectorAll()方法用来获取一组元素。该方法与CSS中的选择器方法类似，如CSS中的id选择器方式、CLASS选择器方式、群组选择器方式、包含选择器方式等，具体示例代码如下：

```
<script>
    document.querySelector('#div1');
    document.querySelectorAll('.box');
    document.querySelectorAll('ul li');
    document.querySelectorAll('h1,h2,h3');
</script>
```

- JavaScript中的query方法获取元素的方式与CSS中获取的方式类似。

3.3 获取集合

3.3.4 innerHTML

- innerHTML方法是用来获取和设置指定标签内的内容。其中内容包括文本，也包括标签等信息。接下来通过案例演示，具体如例所示。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Hello JS</title>
</head>
<body>
   <div id="div1"><span>span</span></div>
<script>
    var div = document.getElementById('div1');
    console.log( div.innerHTML );            // '<span>span</span>'
    div.innerHTML = '<h1>h1</h1>';
</script>
</body>
</html>
```

千锋教育 WWW.QFEDU.COM

3.3.4 innerHTML

- 利用循环+innerHTML的方式可以实现添加一组<li>标签到网页中，具体如例所示。

```
<!doctype html>
<html>
<head>
      <meta charset="utf-8">
      <title>Hello JS</title>
</head>
<body>
    <ul id="ul1"></ul>
<script>
      var ul = document.getElementById('ul1');
      for(var i=0;i<3;i++){
          ul.innerHTML += '<li>'+i+'</li>';
      }
</script>
</body>
</html>
```

3.3.4 innerHTML

- 注意，在for循环时，每次都调用innerHTML对页面进行重新渲染，这对于网页来说会消耗一定性能，因此建议大家一次性操作innerHTML方法，改写如下：

```
<body>
    <ul id="ul1">
    </ul>
</body>
<script>
    var ul = document.getElementById('ul1');
    var tmp = '';
    for(var i=0;i<3;i++){
        tmp += '<li>'+i+'</li>';
    }
    ul.innerHTML = tmp;
</script>
```

3.3 获取集合

千锋教育 WWW.QFEDU.COM

3.3.5 获取元素

- 获取元素的方法，分为动态获取和非动态获取两大类。下面将分别进行讲解。
- **1. 动态获取**
- 属于动态获取的方法包括getElementsByTagName()和getElementsByClassName()。动态获取方法可以不在定义变量时获取元素，而是在调用变量时获取元素。
- **2.非动态获取**
- 属于非动态获取的方法包括getElementById()、querySelector()、querySelectorAll()。这些方法是在定义变量时就已经获取完毕，后添加的元素是获取不到的，除非添加完后再获取。

3.4 实际运用

返回目录

3.4 实际运用

千锋教育 WWW.QFEDU.COM

- 在前面的小节中，学习了如何获取元素，讲解了JavaScript与HTML、CSS进行通信的方法。接下来将结合JavaScript语法实现几个小实例，了解在实际中如何去运用这些方法。

3.4 实际运用

3.4.1 隔行换色

- 在CSS3中，可以通过结构伪类选择器实现一组元素的隔行换色效果。在JavaScript中也可以实现这一效果。接下来通过案例演示，具体如例所示。

```
<!doctype html>
<html>
<head>
     <meta charset="utf-8">
     <title>Hello JS</title>
</head>
<body>
     <ul>
        <li></li>
       <li></li>
        <li></li>
        <li></li>
     </ul>
<script>
     var li = document.getElementsByTagName('li');
     for( var i=0; i<li.length; i++ ){
        if(i%2 == 0){                    //隔行换色
          li[i].style.background = 'red';
        }
        else{
           li[i].style.background = 'blue';
        }
     }
</script>
</body>
</html>
```

千锋教育 WWW.QFEDU.COM

3.4.2 拼接背景色

- 一张背景图片，如何利用一组元素集合进行拼接而成呢？可以利用嵌套循环的方式，把集合分成横竖排列组合。例如，把背景图分成5行、10列，具体如例所示。

```
<!doctype html>
<html>
<head>
     <meta charset="utf-8">
     <title>Hello JS</title>
    <style>
         *{ margin:0; padding:0; }
         li{ width:50px; height:50px; position:absolute;
           left:0; top:0; list-style:none;
           background:url(1.png) no-repeat; }
    </style>
</head>
<body>
     <ul id="ul1"></ul>
<script>
     var ul = document.getElementById('ul1');
     var tmp = '';
     for( var i=0; i<5; i++ ){
         for( var j=0; j<10; j++ ){
    tmp += '<li style="left:' + (j*51) + 'px; top:' + (i*51) + 'px;' +
   'background-position:' + (-j*50) + 'px ' + (-i*50) + 'px"></li>';
         }
     }
     ul.innerHTML = tmp;
</script>
</body>
</html>
```

3.4.3 九九乘法表

- 九九乘法表是嵌套循环的一种变化操作，主要利用嵌套循环的顺序和值的变化来实现。接下来通过案例演示，具体如例所示。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Hello JS</title>
</head>
<body>
    <div id="div1"></div>
</body>
<script>
    var div = document.getElementById('div1');
    for( var i=1; i<=9; i++ ){
        for( var j=1; j<=i; j++ ){
            div.innerHTML += i+'*'+j+'='+(i*j) + ' ';
        }
        div.innerHTML += '<br>';
    }
</script>
```

本章小结

- 本章首先讲解了JavaScript与HTML、CSS之间进行交互操作的方法，然后结合JavaScript语法实现了几个小实例。DOM是JavaScript中的重要组成部分，在后面章节还会讲解到DOM相关的知识。

千锋教育™
WWW.QFEDU.COM

THANK YOU