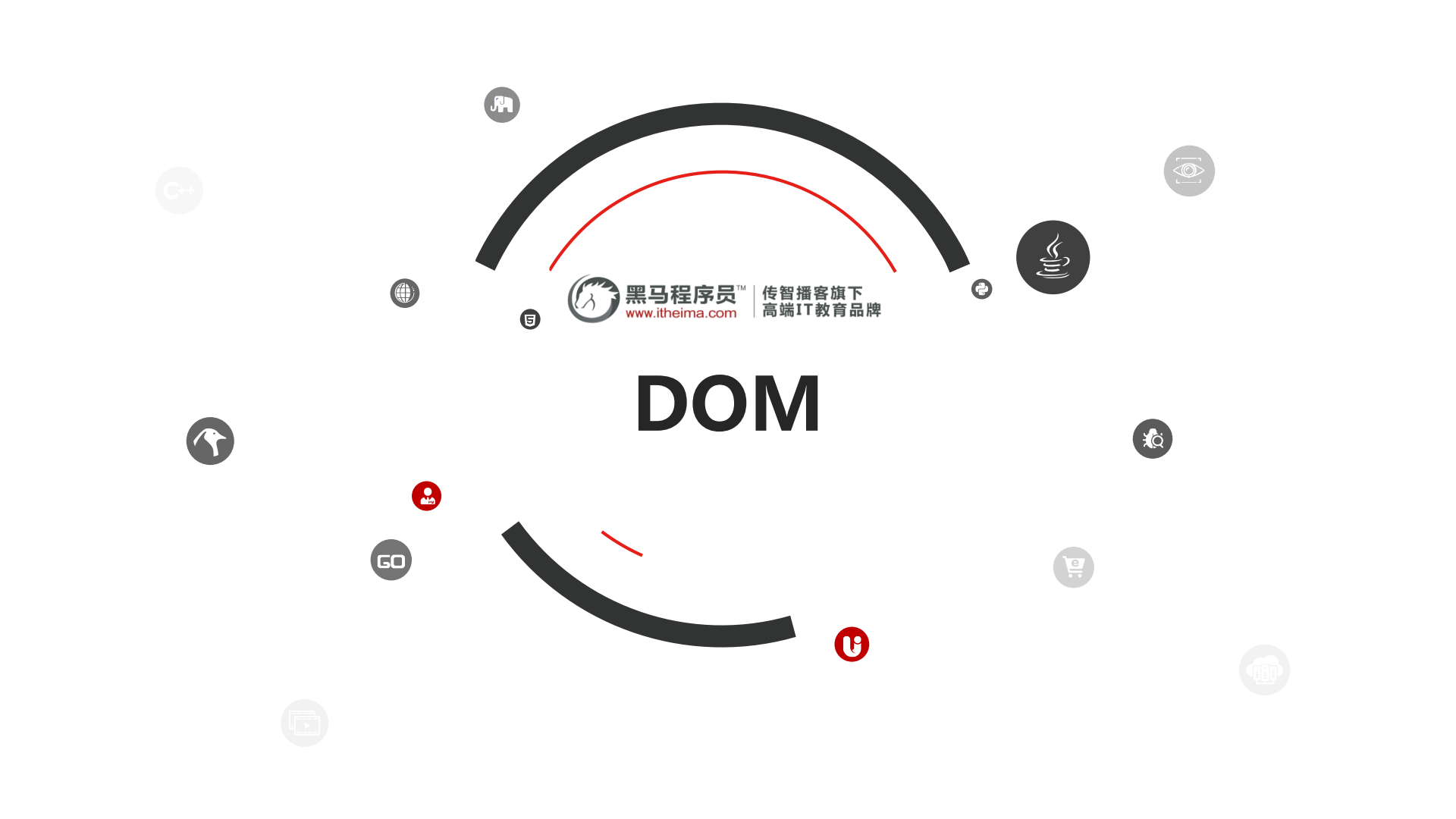


DOM

目录 Contents

1. DOM 简介

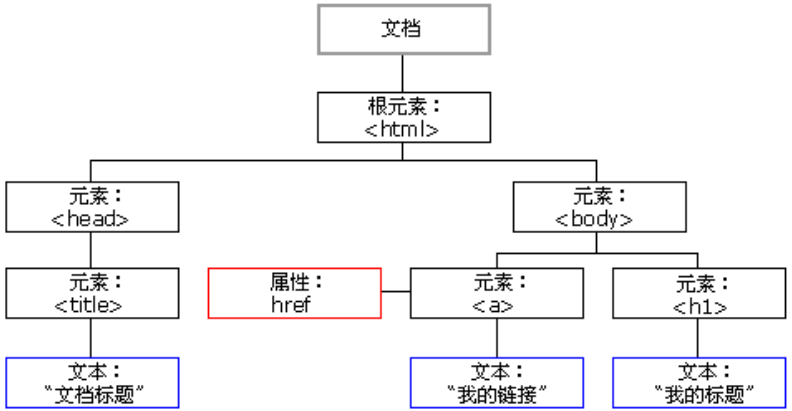
1.1 什么是 DOM

文档对象模型（Document Object Model，简称 DOM），是 W3C 组织推荐的处理可扩展标记语言（HTML或者XML）的标准编程接口。

W3C 已经定义了一系列的 DOM 接口，通过这些 DOM 接口可以改变网页的内容、结构和样式。

1. DOM 简介

1.2 DOM 树



- 文档：一个页面就是一个文档，DOM 中使用 document 表示
- 元素：页面中的所有标签都是元素，DOM 中使用 element 表示
- 节点：网页中的所有内容都是节点（标签、属性、文本、注释等），DOM 中使用 node 表示

DOM 把以上内容都看做是对象

目录 Contents

2. 获取元素

2.1 如何获取页面元素

DOM在我们实际开发中主要用来操作元素。

我们如何来获取页面中的元素呢?

获取页面中的元素可以使用以下几种方式:

- 根据 ID 获取
- 根据标签名获取
- 通过 HTML5 新增的方法获取
- 特殊元素获取

2. 获取元素

2.2 根据 ID 获取

使用 getElementById() 方法可以获取带有 ID 的元素对象。

```
document.getElementById('id');
```

使用 console.dir() 可以打印我们获取的元素对象，更好的查看对象里面的属性和方法。

2. 获取元素

2.3 根据标签名获取

使用 getElementsByTagName() 方法可以返回带有指定标签名的对象的集合。

```
document.getElementsByTagName('标签名');
```

注意：

1. 因为得到的是一个对象的集合，所以我们想要操作里面的元素就需要遍历。
2. 得到元素对象是动态的
3. 如果获取不到元素,则返回为空的伪数组(因为获取不到对象)

2. 获取元素

2.3 根据标签名获取

还可以获取某个元素(父元素)内部所有指定标签名的子元素.

```
element.getElementsByTagName('标签名');
```

注意：父元素必须是单个对象(必须指明是哪一个元素对象). 获取的时候不包括父元素自己。

2. 获取元素

2.4 通过 HTML5 新增的方法获取

```
1. document.getElementsByClassName('类名'); // 根据类名返回元素对象集合
```

```
2. document.querySelector('选择器');          // 根据指定选择器返回第一个元素对象
```

```
3. document.querySelectorAll('选择器');       // 根据指定选择器返回
```

注意：

querySelector 和 querySelectorAll里面的选择器需要加符号,比如:document.querySelector('#nav');

2. 获取元素

2.5 获取特殊元素（body，html）

获取body元素

```
1. doucumnet.body  // 返回body元素对象
```

获取html元素

```
1. document.documentElement  // 返回html元素对象
```

目录 Contents

3. 事件基础

3.1 事件概述

JavaScript 使我们有能力创建动态页面，而事件是可以被 JavaScript 侦测到的行为。

简单理解： 触发--- 响应机制。

网页中的每个元素都可以产生某些可以触发 JavaScript 的事件，例如，我们可以在用户点击某按钮时产生一个事件，然后去执行某些操作。

3. 事件基础

3.2 事件三要素

1. 事件源 （谁）
2. 事件类型 （什么事件）
3. 事件处理程序 （做啥）

3. 事件基础

案例：点击按钮弹出警示框

页面中有一个按钮，当鼠标点击按钮的时候，弹出“你好”警示框。

3. 事件基础

案例分析

① 获取事件源（按钮）
② 注册事件（绑定事件），使用 onclick
③ 编写事件处理程序，写一个函数弹出 alert 警示框

3. 事件基础

实现代码

```
var btn = document.getElementById('btn');
btn.onclick = function() {
  alert('你好吗');
};
```

3.3 执行事件的步骤

1. 获取事件源
2. 注册事件（绑定事件）
3. 添加事件处理程序（采取函数赋值形式）

3.3 常见的鼠标事件

鼠标事件	触发条件
onclick	鼠标点击左键触发
onmouseover	鼠标经过触发
onmouseout	鼠标离开触发
onfocus	获得鼠标焦点触发
onblur	失去鼠标焦点触发
onmousemove	鼠标移动触发
onmouseup	鼠标弹起触发
onmousedown	鼠标按下触发

3.4 分析事件三要素

下拉菜单三要素

关闭广告三要素

目录 Contents

4. 操作元素

JavaScript 的 DOM 操作可以改变网页内容、结构和样式，我们可以利用 DOM 操作元素来改变元素里面的内容 、属性等。注意以下都是属性

4.1 改变元素内容

```
element.innerText
```

从起始位置到终止位置的内容, 但它去除 html 标签， 同时空格和换行也会去掉

```
element.innerHTML
```

起始位置到终止位置的全部内容，包括 html 标签，同时保留空格和换行

4. 操作元素

4.2 常用元素的属性操作

1. innerText、innerHTML 改变元素内容
2. src、href
3. id、alt、title

4. 操作元素

案例： 分时显示不同图片,显示不同问候语

根据不同时间，页面显示不同图片，同时显示不同的问候语。

如果上午时间打开页面，显示上午好，显示上午的图片。

如果下午时间打开页面，显示下午好，显示下午的图片。

如果晚上时间打开页面，显示晚上好，显示晚上的图片。

4. 操作元素

案例分析

① 根据系统不同时间来判断，所以需要用到日期内置对象

② 利用多分支语句来设置不同的图片

③ 需要一个图片，并且根据时间修改图片，就需要用到操作元素src属性

④ 需要一个div元素，显示不同问候语，修改元素内容即可

4. 操作元素

4.3 表单元素的属性操作

利用 DOM 可以操作如下表单元素的属性：

```
type、value、checked、selected、disabled
```

4. 操作元素

案例：仿京东显示密码

点击按钮将密码框切换为文本框，并可以查看密码明文。

4. 操作元素

案例分析

① 核心思路： 点击眼睛按钮，把密码框类型改为文本框就可以看见里面的密码

② 一个按钮两个状态，点击一次，切换为文本框，继续点击一次切换为密码框

③ 算法：利用一个flag变量，来判断flag的值，如果是1 就切换为文本框，flag 设置为0，如果是0 就切换为密码框，flag设置为1

4. 操作元素

实现代码

4. 操作元素

4.4 样式属性操作

我们可以通过 JS 修改元素的大小、颜色、位置等样式。

```
1. element.style      行内样式操作
2. element.className 类名样式操作
```

注意：

1.JS 里面的样式采取驼峰命名法 比如 fontSize、 backgroundColor

2.JS 修改 style 样式操作，产生的是行内样式，CSS 权重比较高

4. 操作元素

案例： 淘宝点击关闭二维码

当鼠标点击二维码关闭按钮的时候，则关闭整个二维码。

4. 操作元素

案例分析

① 核心思路： 利用样式的显示和隐藏完成， display:none 隐藏元素 display:block 显示元素

② 点击按钮，就让这个二维码盒子隐藏起来即可

4. 操作元素

实现代码

```
var btn = document.querySelector('.close-btn');
var box = document.querySelector('.box');
// 2.注册事件 程序处理
btn.onclick = function() {
box.style.display = 'none';
```

4. 操作元素

案例： 循环精灵图背景

可以利用 for 循环设置一组元素的精灵图背景

4. 操作元素

案例分析

① 首先精灵图图片排列有规律的

② 核心思路： 利用for循环 修改精灵图片的 背景位置 background-position

③ 剩下的就是考验你的数学功底了

④ 让循环里面的 i 索引号 * 44 就是每个图片的y坐标

4. 操作元素

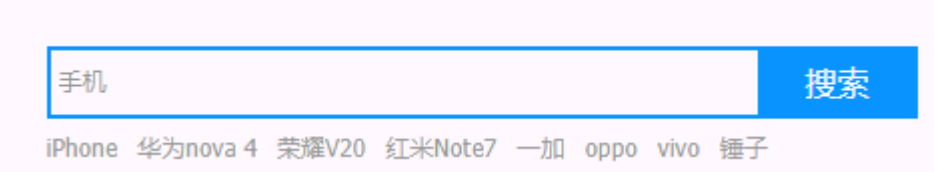
实现代码

```
var lis = document.querySelectorAll('li');
for (var i = 0; i < lis.length; i++) {
// 让索引号 乘以 44 就是每个li 的背景y坐标 index就是我们的y坐标
var index = i * 44;
lis[i].style.backgroundPosition = '0 -' + index + 'px';
}
```

4. 操作元素

案例：显示隐藏文本框内容

当鼠标点击文本框时，里面的默认文字隐藏，当鼠标离开文本框时，里面的文字显示。



4. 操作元素

案例分析

① 首先表单需要2个新事件，获得焦点 onfocus 失去焦点 onblur

② 如果获得焦点， 判断表单里面内容是否为默认文字，如果是默认文字，就清空表单内容

③ 如果失去焦点， 判断表单内容是否为空，如果为空，则表单内容改为默认文字

4. 操作元素

4.4 样式属性操作

我们可以通过 JS 修改元素的大小、颜色、位置等样式。

```
1. element.style      行内样式操作
2. element.className 类名样式操作
```

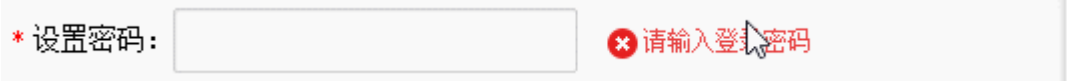
注意:

1. 如果样式修改较多，可以采取操作类名方式更改元素样式。
2. class因为是个保留字，因此使用className来操作元素类名属性
3. className 会直接更改元素的类名，会覆盖原先的类名。

4. 操作元素

案例： 密码框格式提示错误信息

用户如果离开密码框，里面输入个数不是6~16，则提示错误信息，否则提示输入正确信息



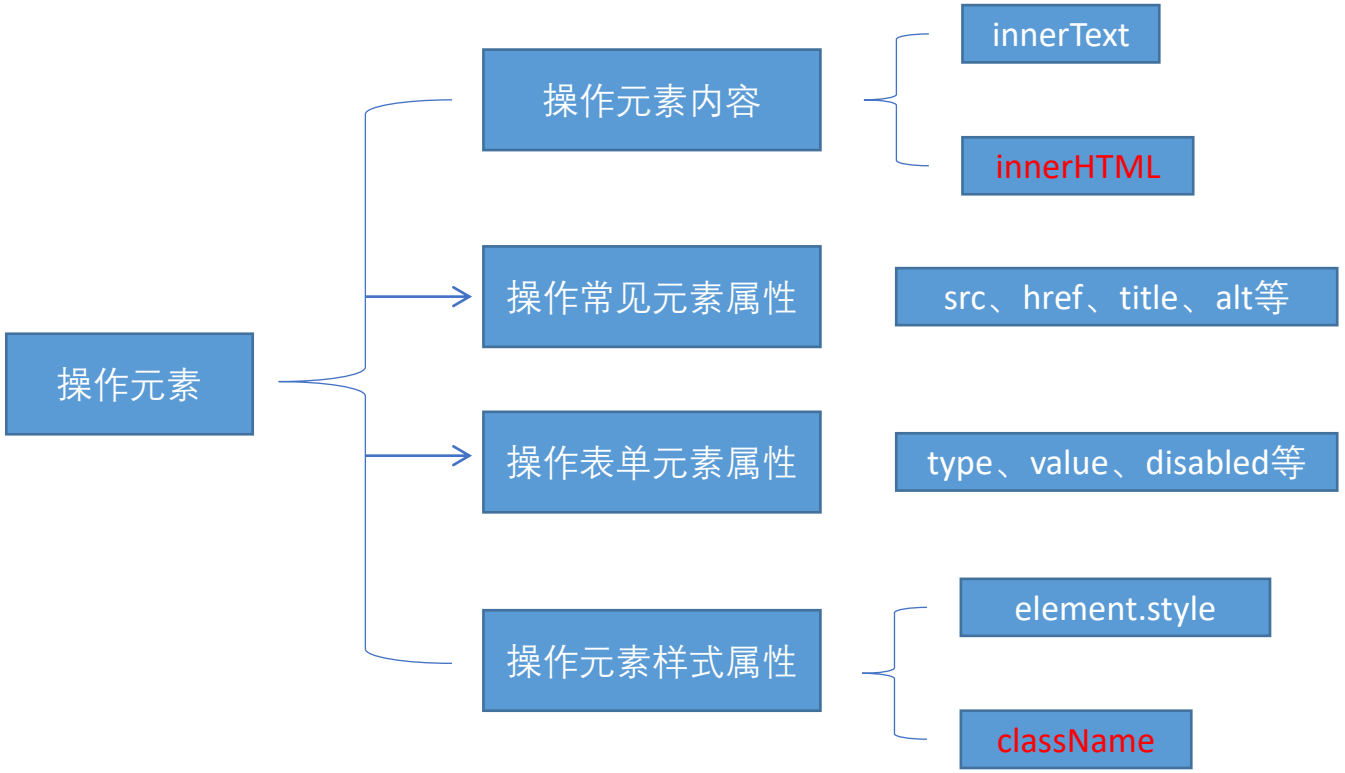
4. 操作元素

案例分析

① 首先判断的事件是表单失去焦点 onblur
② 如果输入正确则提示正确的信息颜色为绿色小图标变化
③ 如果输入不是6到16位，则提示错误信息颜色为红色 小图标变化
④ 因为里面变化样式较多，我们采取className修改样式

4. 操作元素总结

操作元素是 DOM 核心内容



4. 操作元素

作业

1. 世纪佳缘 用户名 显示隐藏内容
2. 京东关闭广告（直接隐藏即可）
3. 新浪下拉菜单（微博即可）
4. 开关灯案例（见素材）

4. 操作元素

4.5 排他思想

如果有同一组元素，我们想要某一个元素实现某种样式， 需要用到循环的排他思想算法：

1. 所有元素全部清除样式（干掉其他人）

2. 给当前元素设置样式 （留下我自己）

3. 注意顺序不能颠倒，首先干掉其他人，再设置自己

4. 操作元素

案例：百度换肤

4. 操作元素

案例分析

① 这个案例练习的是给一组元素注册事件
② 给4个小图片利用循环注册点击事件
③ 当我们点击了这个图片，让我们页面背景改为当前的图片
④ 核心算法： 把当前图片的src 路径取过来，给 body 做为背景即可

4. 操作元素

实现代码

```
// 1. 获取元素
var imgs = document.querySelector('.baidu').querySelectorAll('img');
// 2. 循环注册事件
for (var i = 0; i < imgs.length; i++) {
imgs[i].onclick = function() {
document.body.style.backgroundImage = 'url(' + this.src + ')';
}
}
```

4. 操作元素

案例：表格隔行变色

代码	名称	最新公布净值	累计净值	前单位净值	净值增长率	公布日期
003526	农银金穗3个月定期开放债券	1.075	1.079	1.074	+0.047%	2019-01-11
270047	广发理财30天债券B	0.903	3.386	0.000	0.000%	2019-01-16
163417	兴全合宜混合A	0.860	0.860	0.863	-0.382%	2019-01-16
003929	中银证券安进债券A	1.034	1.088	1.034	+0.077%	2019-01-16
360020	光大添天盈月度理财债券B	0.950	3.557	0.000	0.000%	2019-01-16

4. 操作元素

案例分析

① 用到新的鼠标事件 鼠标经过 onmouseover　鼠标离开 onmouseout

② 核心思路：鼠标经过 tr 行，当前的行变背景颜色， 鼠标离开去掉当前的背景颜色

③ 注意： 第一行（thead里面的行）不需要变换颜色，因此我们获取的是 tbody 里面的行

4. 操作元素

案例：表单全选取消全选案例

业务需求：

1. 点击上面全选复选框，下面所有的复选框都选中（全选）
2. 再次点击全选复选框，下面所有的复选框都不中选（取消全选）
3. 如果下面复选框全部选中，上面全选按钮就自动选中
4. 如果下面复选框有一个没有选中，上面全选按钮就不选中
5. 所有复选框一开始默认都没选中状态

	商品	价钱
☐	iPhone8	8000
☐	iPad Pro	5000
☐	iPad Air	2000
☐	Apple Watch	2000

4. 操作元素

案例分析

① 全选和取消全选做法： 让下面所有复选框的checked属性（选中状态） 跟随 全选按钮即可

② 下面复选框需要全部选中， 上面全选才能选中做法： 给下面所有复选框绑定点击事件，每次点击，都要循环查看下面所有的复选框是否有没选中的，如果有一个没选中的， 上面全选就不选中。

③ 可以设置一个变量，来控制全选是否选中

4. 操作元素

4.6 自定义属性的操作

1. 获取属性值

- element.属性　获取属性值。
- element.getAttribute('属性');

区别：

- element.属性　获取内置属性值（元素本身自带的属性）
- element.getAttribute('属性')；　主要获得自定义的属性　（标准）　我们程序员自定义的属性

4. 操作元素

4.6 自定义属性的操作

2. 设置属性值

- element.属性 = ‘值’　　设置内置属性值。
- element.setAttribute('属性', '值');

区别：

- element.属性　设置内置属性值
- element.setAttribute(‘属性’);　主要设置自定义的属性　(标准)

4.6 自定义属性的操作

3. 移除属性

- `element.removeAttribute('属性');`

4. 操作元素

案例：tab 栏切换（重点案例）

当鼠标点击上面相应的选项卡（tab），下面内容跟随变化

4. 操作元素

案例分析

① Tab栏切换有2个大的模块

② 上的模块选项卡，点击某一个，当前这一个底色会是红色，其余不变（排他思想） 修改类名的方式

③ 下面的模块内容，会跟随上面的选项卡变化。所以下面模块变化写到点击事件里面。

④ 规律：下面的模块显示内容和上面的选项卡一一对应，相匹配。

⑤ 核心思路： 给上面的tab_list 里面的所有小li 添加自定义属性，属性值从0开始编号。

⑥ 当我们点击tab_list 里面的某个小li，让tab_con 里面对应序号的 内容显示，其余隐藏（排他思想）

4. 操作元素

4.7 H5自定义属性

自定义属性目的：是为了保存并使用数据。有些数据可以保存到页面中而不用保存到数据库中。

自定义属性获取是通过getAttribute('属性') 获取。

但是有些自定义属性很容易引起歧义，不容易判断是元素的内置属性还是自定义属性。

H5给我们新增了自定义属性：

1. 设置H5自定义属性

H5规定自定义属性data-开头做为属性名并且赋值。

比如

或者使用 JS 设置

element.setAttribute('data-index', 2)

4. 操作元素

4.7 H5自定义属性

2. 获取H5自定义属性

1. 兼容性获取 element.getAttribute（‘data-index’）;
2. H5新增 element.dataset.index 或者 element.dataset[‘index’] ie 11才开始支持

目录 Contents

5. 节点操作

5.1 为什么学节点操作

获取元素通常使用两种方式：

1. 利用 DOM 提供的方法获取元素

- document.getElementById()
- document.getElementsByTagName()
- document.querySelector 等
- 逻辑性不强、繁琐

2. 利用节点层级关系获取元素

- 利用父子兄节点关系获取元素
- 逻辑性强， 但是兼容性稍差

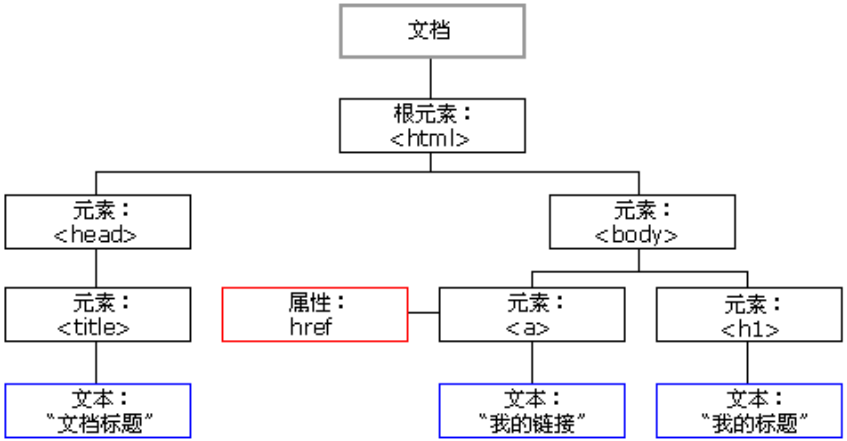
这两种方式都可以获取元素节点，我们后面都会使用，但是节点操作更简单

5. 节点操作

5.2 节点概述

网页中的所有内容都是节点（标签、属性、文本、注释等），在DOM 中，节点使用 node 来表示。

HTML DOM 树中的所有节点均可通过 JavaScript 进行访问，所有 HTML 元素（节点）均可被修改，也可以创建或删除。



5. 节点操作

5.2 节点概述

一般地，节点至少拥有nodeType（节点类型）、nodeName（节点名称）和nodeValue（节点值）这三个基本属性。

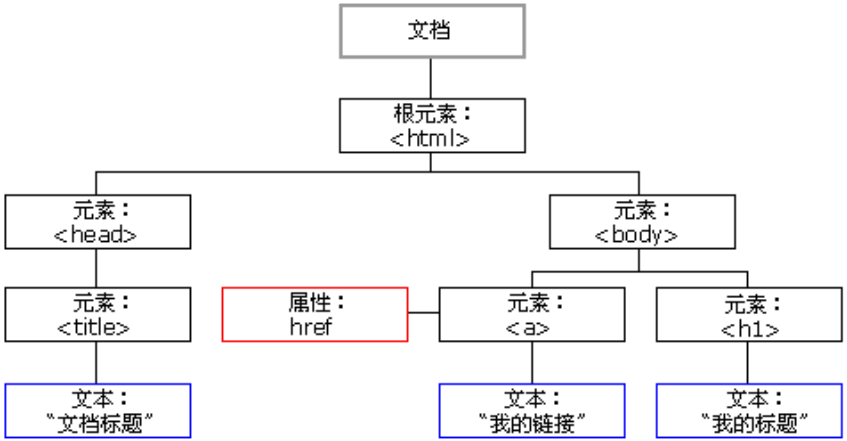
- 元素节点 nodeType 为 1
- 属性节点 nodeType 为 2
- 文本节点 nodeType 为 3 （文本节点包含文字、空格、换行等）

我们在实际开发中，节点操作主要操作的是**元素节点**

5. 节点操作

5.3 节点层级

利用 DOM 树可以把节点划分为不同的层级关系，常见的是**父子兄层级关系**。



5. 节点操作

5.3 节点层级

利用 DOM 树可以把节点划分为不同的层级关系，常见的是**父子兄层级关系**。

1. 父级节点

```
node.parentNode
```

- parentNode 属性可返回某节点的父节点，注意是最近的一个父节点
- 如果指定的节点没有父节点则返回 null

5. 节点操作

5.3 节点层级

2. 子节点

1. parentNode.**childNodes（标准）**

parentNode.childNodes 返回包含指定节点的子节点的集合，该集合为即时更新的集合。

注意：返回值里面包含了所有的子节点，包括元素节点，文本节点等。

如果只想要获得里面的元素节点，则需要专门处理。 所以我们一般不提倡使用childNodes

```
var ul = document. querySelector('ul');
for(var i = 0; i < ul.childNodes.length;i++) {
    if (ul.childNodes[i].nodeType == 1) {
        // ul.childNodes[i] 是元素节点
        console.log(ul.childNodes[i]);
    }
}
```

5. 节点操作

5.3 节点层级

2. 子节点

```
2. parentNode.children（非标准）
```

`parentNode.children` 是一个只读属性，返回所有的子元素节点。它只返回子元素节点，其余节点不返回 （**这个是我们重点掌握的**）。

虽然children 是一个非标准，但是得到了各个浏览器的支持，因此我们可以放心使用

5. 节点操作

5.3 节点层级

2. 子节点

```
3. parentNode.firstChild
```

firstChild 返回第一个子节点，找不到则返回null。同样，也是包含所有的节点。

```
4. parentNode.lastChild
```

lastChild 返回最后一个子节点，找不到则返回null。同样，也是包含所有的节点。

5.3 节点层级

2. 子节点

```
5. parentNode.firstElementChild
```

`firstElementChild` 返回第一个子元素节点，找不到则返回null。

```
6. parentNode.lastElementChild
```

`lastElementChild` 返回最后一个子元素节点，找不到则返回null。

注意：这两个方法有兼容性问题，`IE9` 以上才支持。

5. 节点操作

5.3 节点层级

2. 子节点

实际开发中，`firstChild` 和 `lastChild` 包含其他节点，操作不方便，而 `firstElementChild` 和 `lastElementChild` 又有兼容性问题，那么我们如何获取第一个子元素节点或最后一个子元素节点呢?

解决方案:

1. 如果想要第一个子元素节点，可以使用 `parentNode.chilren[0]`
2. 如果想要最后一个子元素节点，可以使用 `parentNode.chilren[parentNode.chilren.length - 1]`

5. 节点操作

案例：下拉菜单

5. 节点操作

案例分析

① 导航栏里面的li 都要有鼠标经过效果，所以需要循环注册鼠标事件

② 核心原理： 当鼠标经过li 里面的 第二个孩子 ul 显示， 当鼠标离开，则ul 隐藏

5. 节点操作

实现代码

```
var nav = document.querySelector('.nav');
var lis = nav.children; // 得到4个小li
for (var i = 0; i < lis.length; i++) {
lis[i].onmouseover = function() {
this.children[1].style.display = 'block';
}
lis[i].onmouseout = function() {
this.children[1].style.display = 'none';
}
}
```

5. 节点操作

5.3 节点层级

3. 兄弟节点

```
1. node.nextSibling
```

nextSibling 返回当前元素的下一个兄弟元素节点，找不到则返回null。同样，也是包含所有的节点。

```
2. node.previousSibling
```

previousSibling 返回当前元素上一个兄弟元素节点，找不到则返回null。同样，也是包含所有的节点。

5. 节点操作

5.3 节点层级

3. 兄弟节点

```
3. node.nextElementSibling
```

`nextElementSibling` 返回当前元素下一个兄弟元素节点，找不到则返回null。

```
4. node.previousElementSibling
```

`previousElementSibling` 返回当前元素上一个兄弟节点，找不到则返回null。

注意：这两个方法有兼容性问题， `IE9` 以上才支持。

5. 节点操作

5.3 节点层级

3. 兄弟节点

问：如何解决兼容性问题？

答：自己封装一个兼容性的函数

```
function getNextElementSibling(element) {
   var el = element;
   while (el = el.nextSibling) {
     if (el.nodeType === 1) {
         return el;
     }
   }
   return null;
 }
```

5. 节点操作

5.4 创建节点

```
document.createElement('tagName')
```

`document.createElement()` 方法创建由 `tagName` 指定的 `HTML` 元素。因为这些元素原先不存在，是根据我们的需求动态生成的，所以我们也称为动态创建元素节点。

5. 节点操作

5.4 添加节点

```
1. node.appendChild(child)
```

`node.appendChild()` 方法将一个节点添加到指定父节点的子节点列表末尾。类似于 `CSS` 里面的 `after` 伪元素。

```
2. node.insertBefore(child, 指定元素)
```

`node.insertBefore()` 方法将一个节点添加到父节点的指定子节点前面。类似于 `CSS` 里面的 `before` 伪元素。

5. 节点操作

案例：简单版发布留言案例

5. 节点操作

案例分析

① 核心思路： 点击按钮之后，就动态创建一个li，添加到ul 里面。

② 创建li 的同时，把文本域里面的值通过li.innerHTML 赋值给 li

③ 如果想要新的留言后面显示就用 appendChild 如果想要前面显示就用insertBefore

5. 节点操作

5.5 删除节点

```
node.removeChild(child)
```

`node.removeChild()` 方法从 `DOM` 中删除一个子节点，返回删除的节点。

5. 节点操作

案例：删除留言案例

5. 节点操作

案例分析

① 当我们把文本域里面的值赋值给li 的时候，多添加一个删除的链接

② 需要把所有的链接获取过来，当我们点击当前的链接的时候，删除当前链接所在的li

③ 阻止链接跳转需要添加 javascript:void(0); 或者 javascript:;

5. 节点操作

5.6 复制节点(克隆节点)

```
node.cloneNode()
```

`node.cloneNode()` 方法返回调用该方法的节点的一个副本。 也称为克隆节点/拷贝节点

注意:

1. 如果括号参数为空或者为 `false` ，则是浅拷贝，即只克隆复制节点本身，不克隆里面的子节点。
2. 如果括号参数为 `true` ，则是深度拷贝，会复制节点本身以及里面所有的子节点。

5. 节点操作

案例：动态生成表格

姓名	科目	成绩	操作
魏璎珞	JavaScript	100	删除
弘历	JavaScript	90	删除
傅恒	JavaScript	99	删除
明玉	JavaScript	89	删除

5. 节点操作

案例分析

① 因为里面的学生数据都是动态的，我们需要js 动态生成。 这里我们模拟数据，自己定义好数据。 数据我们采取对象形式存储。

② 所有的数据都是放到tbody里面的行里面。

③ 因为行很多，我们需要循环创建多个行（对应多少人）

④ 每个行里面又有很多单元格（对应里面的数据），我们还继续使用循环创建多个单元格，并且把数据存入里面（双重for循环）

⑤ 最后一列单元格是删除，需要单独创建单元格。

⑥ 最后添加删除操作，单击删除，可以删除当前行。

5. 节点操作

5.8 三种动态创建元素区别

- document.write()
- element.innerHTML
- document.createElement()

区别

1. document.write 是直接将内容写入页面的内容流，但是文档流执行完毕，则它会导致页面全部重绘
2. innerHTML 是将内容写入某个 DOM 节点，不会导致页面全部重绘
3. innerHTML 创建多个元素效率更高（不要拼接字符串，采取数组形式拼接），结构稍微复杂
4. createElement() 创建多个元素效率稍低一点点，但是结构更清晰

总结：不同浏览器下，innerHTML 效率要比 createElement 高

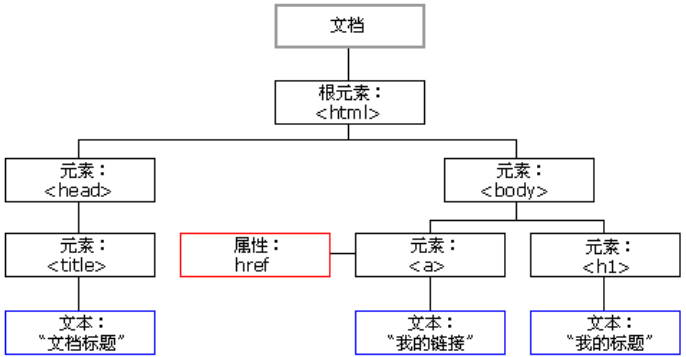
6. DOM 重点核心

文档对象模型（Document Object Model，简称 DOM），是 W3C 组织推荐的处理可扩展标记语言（HTML或者XML）的标准编程接口。

W3C 已经定义了一系列的 DOM 接口，通过这些 DOM 接口可以改变网页的内容、结构和样式。

1. 对于JavaScript，为了能够使JavaScript操作HTML，JavaScript就有了一套自己的dom编程接口。

2. 对于HTML，dom使得html形成一棵dom树. 包含 文档、元素、节点



我们获取过来的DOM元素是一个对象（object），所以称为 文档对象模型

6. DOM 重点核心

关于dom操作，我们主要针对于元素的操作。主要有创建、增、删、改、查、属性操作、事件操作。

6.1 创建

1. document.write
2. innerHTML
3. createElement

6. DOM 重点核心

关于dom操作，我们主要针对于元素的操作。主要有创建、增、删、改、查、属性操作、事件操作。

6.2 增

1. appendChild
2. insertBefore

6. DOM 重点核心

关于dom操作，我们主要针对于元素的操作。主要有创建、增、删、改、查、属性操作、事件操作。

6.3 删

1. removeChild

6. DOM 重点核心

关于dom操作，我们主要针对于元素的操作。主要有创建、增、删、改、查、属性操作、事件操作。

6.4 改

主要修改dom的元素属性，dom元素的内容、属性, 表单的值等

1. 修改元素属性： src、href、title等
2. 修改普通元素内容： innerHTML 、innerText
3. 修改表单元素： value、type、disabled等
4. 修改元素样式： style、className

6. DOM 重点核心

关于dom操作，我们主要针对于元素的操作。主要有创建、增、删、改、查、属性操作、事件操作。

6.5 查

主要获取查询dom的元素

1. DOM提供的API 方法： getElementById、getElementsByTagName 古老用法 不太推荐
2. H5提供的新方法： querySelector、querySelectorAll 提倡
3. 利用节点操作获取元素： 父(parentNode)、子(children)、兄(previousElementSibling、nextElementSibling) 提倡

6. DOM 重点核心

关于dom操作，我们主要针对于元素的操作。主要有创建、增、删、改、查、属性操作、事件操作。

6.6 属性操作

主要针对于自定义属性。

1. setAttribute：设置dom的属性值
2. getAttribute：得到dom的属性值
3. removeAttribute移除属性

6. DOM 重点核心

关于dom操作，我们主要针对于元素的操作。主要有创建、增、删、改、查、属性操作、事件操作。

6.7 事件操作

给元素注册事件， 采取 事件源.事件类型 = 事件处理程序

鼠标事件	触发条件
onclick	鼠标点击左键触发
onmouseover	鼠标经过触发
onmouseout	鼠标离开触发
onfocus	获得鼠标焦点触发
onblur	失去鼠标焦点触发
onmousemove	鼠标移动触发
onmouseup	鼠标弹起触发
onmousedown	鼠标按下触发

传智播客旗下高端IT教育品牌