正课:

1. 什么是DOM

2. DOM Tree

3. 查找

4. 修改

1. 什么是DOM:

js: ES(核心语法)+DOM(操作网页)+BOM(操作浏览器)

DOM: Document Object Model

专门操作网页内容的API标准——W3C

为什么: 统一操作网页内容的API标准

只要用DOM API操作网页100%兼容所有浏览器

能做什么: 查找，修改，添加，删除，事件

2. DOM Tree:

什么是: 内存中保存所有网页内容的树形结构

为什么: 树形结构最适合保存上下级包含关系

何时: 只要浏览器读到一个.html网页时，会自动创建DOM树

如何:

1. 先创建唯一的根节点对象document，代表整个网页

2. 依次创建每个后代节点对象Node

网页中每项内容(元素, 属性, 文本)，都是一个节点对象

所有节点对象都是document的后代节点

节点对象:

每个节点对象都有三大属性:

node.nodeType: 判断节点的类型

值: 数字

包括: 9-document 1-element 2-attribute 3-text

node.nodeName: 判断节点的名称\*\*\*

值: 字符串

包括:

document #document

element 全大写的标签名

attribute 属性名

text #text

node.nodeValue: 判断节点的值

document null

element null

attribute 属性值

text 文本内容

3. 查找:

什么是:找到要修改的元素对象

何时: 几乎所有的功能第一步都是查找！

如何: 4种:

1. 不需要查找就可直接获得的元素:

document.documentElement <html>

document.head <head>

document.body <body>

document.forms[i] <form>

2. 按节点间关系查找:

何时: 前提: 已经获得一个节点，要找周围附近的节点时

节点树: 包含所有节点对象的完整树型结构

2大类:

1. 父子关系:

node.parentNode 获得node的父节点

node.childNodes 获得node的所有直接子节点

返回所有直接子节点的集合

node.firstChild 获得node的第一个直接子节点

node.lastChild 获得node的最后一个直接子节点

2. 兄弟关系:

node.previousSibling 获得node的前一个兄弟

node.nextSibling 获得node的后一个兄弟

问题: 包含看不见的回车和空字符，干扰查找！

解决:

元素树: 只包含元素节点的树结构

优: 不包含回车和看不见的空字符，查找不受干扰

2大类:

1. 父子关系:

elem.parentElement 获得elem的父元素

elem.children 获得elem的所有直接子元素

返回所有直接子元素的集合

elem.firstElementChild 获得elem的第一个直接子元素

elem.lastElementChild获得elem的最后一个直接子元素

2. 兄弟关系:

elem.previousElementSibling 获得elem的前一个兄弟元素

elem.nextElementSibling 获得elem的后一个兄弟元素

问题: 兼容性 IE9+

遍历一个父元素下所有后代节点: 2步

1. 定义函数仅遍历一级直接子元素

2. 对每个子元素，调用和父元素完全相同的函数——递归

Vue框架中，就是利用遍历后代节点的方法，找到需要实时更新内容个元素，并不断监视。

3. 按HTML特征查找: 4个API:

1. 按id查找:

var elem=document.getElementById("id")

强调: 1. 只能用document调用

2. 总是返回一个元素对象，找不到返回null

2. 按标签名查找:

var elems=parent.getElementsByTagName("标签名")

强调: 1. 可在任意父元素上调用

2. 返回一个类数组对象，找不到返回空类数组对象

3. 不仅查找直接子元素，而且查找所有后代

3. 按name属性查找:

var elems=document.getElementsByName("name")

强调: 1. 只能用document调用

2. 返回一个类数组对象

4. 按class属性查找:

var elems=parent.getElementsByClassName("class")

强调: 1. 可在任意父元素上调用

2. 返回一个类数组对象

3. 不仅查找直接子元素，而且查找所有后代

4. 不必使用所有class名找，只要用其中一个查找就能获得当前元素

问题: 按HTML查找，一次只能用一个条件查找

如果查找条件复杂，代码很繁琐！

解决:

4. 按选择器查找:

何时: 当查找条件复杂时，首选按选择器查找

如何: 2个API:

1. 只查找一个元素:

var elem=parent.querySelector("selector")

返回值: 一个元素对象，找不到返回null

2. 查找多个元素:

var elems=parent.querySelectorAll("selector")

返回值: 包含多个元素对象的类数组对象，如果找不到返回空类数组对象

强调: 1. 都能在任意父元素上调用

2. 兼容性受制于浏览器

鄙视: 按HTML查找 vs 按选择器查找

1. 返回值:

按HTML查找，返回动态集合

动态集合: 不实际存储所有数据，每次访问集合，都要重新查找DOM树

按选择器查找，返回非动态集合

非动态集合: 实际存储完整数据，及时反复访问集合，也不会导致重新查找DOM树

2. 效率:

首次查找: 按HTML查找，快

按选择器查找，慢

https://jsperf.com/queryselectorall-vs-getelementsbytagname

3. 易用性:

按选择器查找，好用, 按HTML查找，繁琐

总结:

只靠一个条件就可找到想要的元素时，首选按html查找

查找条件复杂时，首选按选择器查找

4. 修改:

3种:

1. 内容: 3种情况:

1. 获取或修改原始HTML代码片段:

elem.innerHTML

2. 获取或修改纯文本内容:

elem.textContent

比innerHTML多做两件事:

1. 将转义字符翻译为正文

2. 去掉所有内嵌的标签

兼容性问题: IE9+

3. 获取或修改表单元素的值:

elem.value

2. 属性:

3. 样式:

修改内联样式:js: elem.style.css属性="值"

html: <elem style="css属性:值"

强调: css属性名必须去横线变驼峰

比如: fontSize, backgroundColor, listStyleType

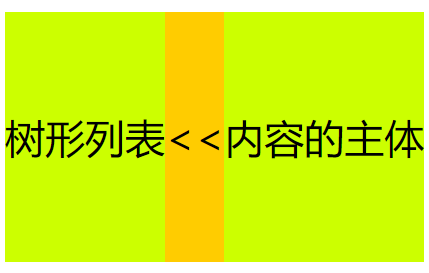
课后任务:

(1)复习: DOM查找API

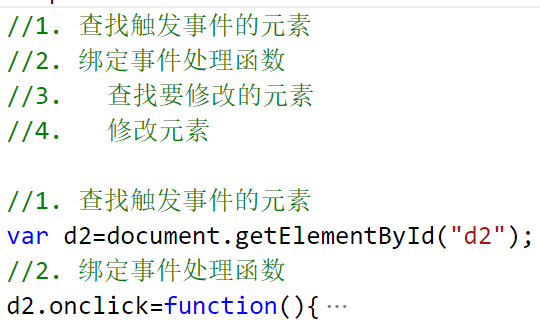
(2)作业: 完成课后练习:  
 题目要求:

使用DOM API实现开关门效果，要求支持过渡效果

运行效果:



提示:



(3)项目:

题目要求:

运行效果: