DOM 编程

从入门到工作：JavaScript 编程接口

版权声明

联系方式

如果你想要购买本课程
请微信联系 xiedaimala02 或 xiedaimala03

如果你发现有人盗用本课程
请微信联系 xiedaimala02 或 xiedaimala03

前置知识

- 学习本课需要什么知识

- ✓ 理解简单的 JS 语法，如变量、if else、循环
- ✓ 会背 JS 的七种数据类型
- ✓ 会背 JS 的五个 falsy 值
- ✓ 知道函数是对象，数组也是对象
- ✓ 会用 div 和 span 标签
- ✓ 会简单的 CSS 布局

网页其实是一棵树

第一个知识点

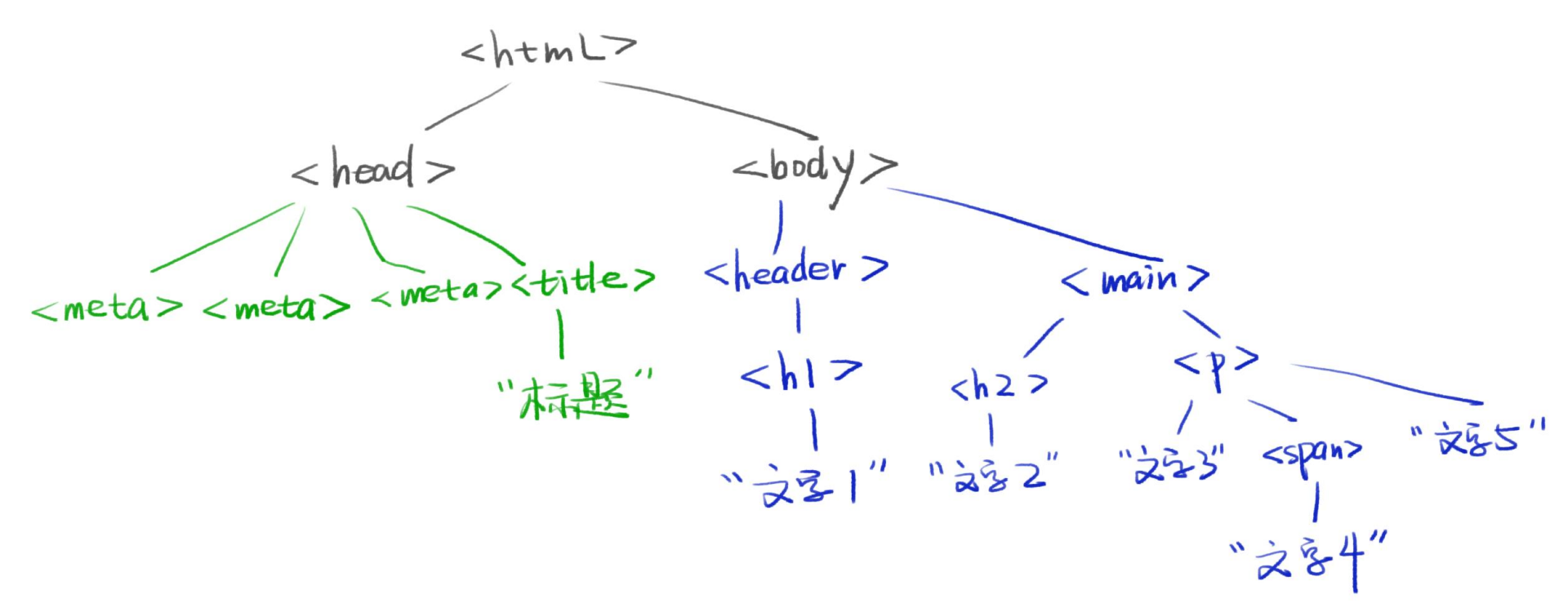
```html
<!DOCTYPE html>
<html lang="zh">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>标题</title>
</head>

<body>
  <header>
    <h1>文字1</h1>
  </header>
  <main>
    <h2>文字2</h2>
    <p>文字3 <span>文字4</span> 文字5</p>
  </main>
</body>

</html>
```

```
1  <!DOCTYPE html>
2  <html lang="zh">
3
4  <head>
```




```
19 </body>
20
21 </html>
```

JS 如何操作这棵树

浏览器往 window 上加一个 document 即可

JS 用 document 操作网页

这就是 Document Object Model 文档对象模型

记住一个事实

DOM 很难用

下节课我们会想办法解决这个难题

如果你觉得 DOM 很傻

不要怀疑自己，你觉得的是对的

获取元素，也叫标签

- 有很多 API
  - window.idxxx 或者直接 idxxx
  - document.getElementById('idxxx')
  - document.getElementsByTagName('div')[0] 找到所有标签名为div的元素
  - document.getElementsByClassName('red')[0]
  - document.querySelector('#idxxx') 找到第一个满足条件的元素？
  - document.querySelectorAll('.red')[0] 找到所有满足条件的元素？

```
document.querySelectorAll('div')[2]
```

- 用哪一个
  - 工作中用 querySelector 和 querySelectorAll
  - 做 demo 直接用 idxxx，千万别让人发现
  - 要兼容 IE 的可怜虫才用 getElement(s)ByXXX

获取特定元素

- 获取 html 元素
  - document.documentElement
- 获取 head 元素
  - document.head
- 获取 body 元素
  - document.body
- 获取窗口（窗口不是元素）
  - window
- 获取所有元素
  - document.all 其实是数组，可以用document.all[3]获取元素。
  - 这个 document.all 是个奇葩，第 6 个 falsy 值 为了区分是不是ie

获取到的元素是个啥

显然是一个对象，我们需要搞清它的原型

抓一只 div 对象来看看

- console.dir(div1) 看原型链
  - 告诉你一个秘密，Chrome 显示错了
  - 自身属性：className、id、style 等等
  - 第一层原型 HTMLDivElement.prototype
  - 这里面是所有 div 共有的属性，不用细看
  - 第二层原型 HTMLElement.prototype
  - 这里面是所有 HTML 标签共有的属性，不用细看
  - 第三层原型 Element.prototype
  - 这里面是所有 XML、HTML 标签的共有属性，你不会以为浏览器只能展示 HTML 吧
  - 第四层原型 Node.prototype
  - 这里面是所有节点共有的属性，节点包括 XML 标签文本注释、HTML 标签文本注释等等
  - 第五层原型 EventTarget.prototype
  - 里面最重要的函数属性是 addEventListener
  - 最后一层原型就是 Object.prototype 了

div 完整原型链

自身属性和共有属性，点击查看

节点？元素？傻傻分不清

- 节点 Node 包括以下几种
  - MDN 有完整描述，x.nodeType 得到一个数字
  - 1 表示元素 Element，也叫标签 Tag
  - 3 表示文本 Text
  - 8 表示注释 Comment
  - 9 表示文档 Document
  - 11 表示文档片段 DocumentFragment
  - 记住 1 和 2 即可

节点的增删改查

程序员的宿命就是增删改查

增

- 创建一个标签节点
  - let div1 = document.createElement('div')
  - document.createElement('style')
  - document.createElement('script')
  - document.createElement('li')
- 创建一个文本节点 文本节点是对象
  - text1 = document.createTextNode('你好') ‘你好’是字符串
- 标签里面插入文本
  - div1.appendChild(text1)
  - div1.innerText = '你好' 或者 div1.textContent = '你好'
  - 但是不能用 div1.appendChild('你好')

增(续)

- 插入页面中
  - 你创建的标签默认处于 JS 线程中
  - 你必须把它插到 head 或者 body 里面，它才会生效
  - document.body.appendChild(div) 这里div是前面那页的变量
  - 或者
  - 已在页面中的元素.appendChild(div)

appendChild

- 代码

✓ 页面中有 div#test1 和 div#test2

```
let div = document.createElement('div')
test1.appendChild(div)
test2.appendChild(div)
```

✓ 请问最终 div 出现在哪里?

1. test1 里面
2. test2 里面
3. test1 里面和 test2 里面

答案：test2 里面

`let div2=div1.clonenNode(true);ture`是深拷贝，false是浅拷贝

一个元素不能出现在两个地方，除非复制一份

删

- 两种方法 `div2.parentNode.removeChild(div2);div2是变量`
  - ✓ 旧方法：parentNode.childChild(childNode)
  - ✓ 新方法：childNode.remove() `ie不能用`
- 思考
  - ✓ 如果一个 node 被移出页面（DOM树）
  - ✓ 那么它还可以再次回到页面中吗？ 可以。内存没被删。要删就 div2=null;
  - ✓ 试试就知道了

改属性

- 写标准属性
  - ✓ 改 class：div.className = 'red blue'（全覆盖）
  - ✓ 改 class：div.classList.add('red') 或者div.className+=’red’;
  - ✓ 改 style：div.style = 'width: 100px; color: blue;' 全覆盖
  - ✓ 改 style 的一部分：div.style.width = '200px'
  - ✓ 大小写：div.style.backgroundColor = 'white'
  - ✓ 改 data-* 属性：div.dataset.x = 'frank'
  - 改id:div2.id=’id’
- 读标准属性
  - ✓ div.classList / a.href
  - ✓ div.getAttribute('class') / a.getAttribute('href')
  - ✓ 两种方法都可以，但值可能稍微有些不同

```
"http://js.jirengu.com/xxx"

"/xxx"
```

```
8 <a id=test
  href="/xxx">/xxx</a>
```

```
1 console.log(test.href)
2 console.log(test.getAttribu
  te('href'))
```

改事件处理函数

- div.onclick 默认为 null
  - ✓ 默认点击 div 不会有任何事情发生
  - ✓ 但是如果你把 div.onclick 改为一个函数 fn
  - ✓ 那么点击 div 的时候，浏览器就会调用这个函数
  - ✓ 并且是这样调用的 fn.call(div, event)
  - ✓ div 会被当做 this
  - ✓ event 则包含了点击事件的所有信息，如坐标
- div.addEventListener
  - ✓ 是 div.onclick 的升级版，之后的课程单独讲

```
<div id="test">test</div>
```

Output

test

```
console.log(test.onclick)
test.onclick = function(x){
  console.log(this)
  console.log(x)
}

// test.onclick.call(test, event)
```

改内容

- 改文本内容
  - ✓ div.innerText = 'xxx'
  - ✓ div.textContent = 'xxx'
  - ✓ 两者几乎没有区别 `div.innerText='<p>aaa</p>';` 标签无效，会当作文本显示
- 改 HTML 内容
  - ✓ div.innerHTML = '重要内容'
- 改标签
  - ✓ div.innerHTML = '' // 先清空
  - ✓ div.appendChild(div2) // 再加内容

改爸爸

- 想要找一个新爸爸?
  - ✓ newParent.appendChild(div)
  - ✓ 直接这样就可以了，直接从原来的地方消失

查

- 查爸爸
  - node.parentNode 或者 node.parentElement
- 查爷爷
  - node.parentNode.parentNode

```
<ul>
  <li>1</li>
  <li>2</li>
  <li>3</li>
</ul>
```

- 查子代

```
let a=document.querySelector('ul');
console.log(a.childNodes.length);
```

长为7。回车、换行为文本节点

  - node.childNodes 或者 node.children
  - 思考：当子代变化时，两者也会实时变化吗?
- 查兄弟姐妹

```
let a=document.querySelector('ul');
console.log(a.children.length);
```

长为3。children是element。

  - node.parentNode.childNodes 还要排除自己
  - node.parentNode.children 还有排除自己

查

- 查看老大 `node.children[0]`
  - ✓ node.firstChild
- 查看老幺
  - ✓ node.lastChild
- 查看上一个哥哥/姐姐
  - ✓ node.previousSibling
- 查看下一个弟弟/妹妹
  - ✓ node.nextSibling

可能查到文本节点。避开：node.nextElementSibling

查

- 遍历一个 div 里面的所有元素

```
travel = (node, fn) => {
  fn(node)
  if(node.children){
    for(let i=0;i<node.children.length;i++){
      travel(node.children[i], fn)
    }
  }
}
travel(div1, (node) => console.log(node))
```

- 看，数据结构多么有用

DOM 操作是跨线程的

详解

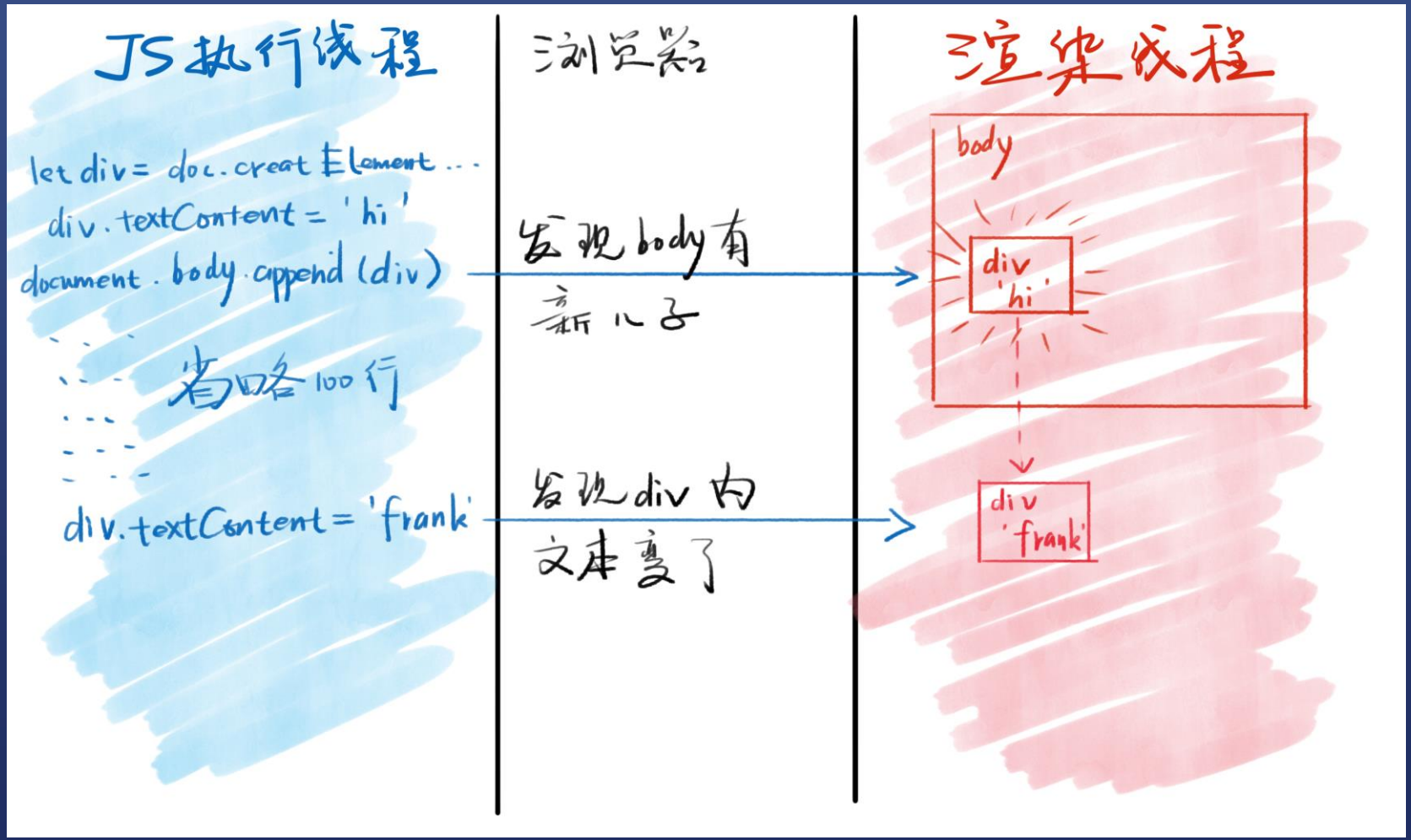
还记得《JS世界》里讲的浏览器功能划分吗

浏览器分为渲染引擎和JS引擎

跨线程操作

- 各线程各司其职
  - JS 引擎不能操作页面，只能操作 JS
  - 渲染引擎不能操作JS，只能操作页面
  - document.body.appendChild(div1)
  - 这句 JS 是如何改变页面的?
- 跨线程通信
  - 当浏览器发现 JS 在 body 里面加了个 div1 对象
  - 浏览器就会通知渲染引擎在页面里也新增一个 div 元素
  - 新增的 div 元素所有属性都照抄 div1 对象

图示跨线程操作


JS执行线程
浏览器
渲染线程
let div = doc.creatElement...
div.textContent = 'hi'
document.body.append(div)
发现body有新儿子
body
div
'hi'
省略100行
div.textContent = 'frank'
发现div内文本变了
div
'frank'

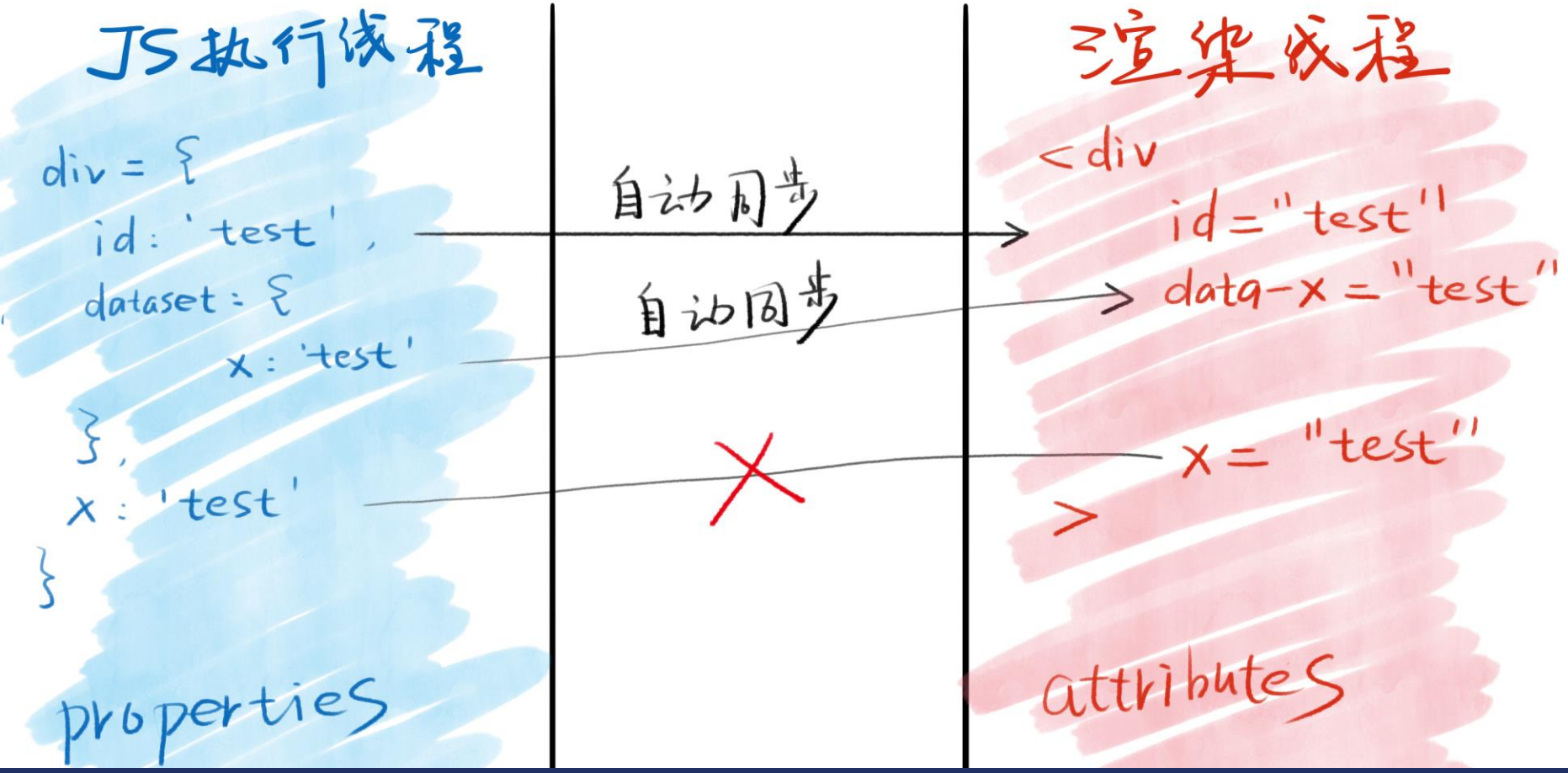
插入新标签的完整过程

- 在 div1 放入页面之前
  - 你对 div1 所有的操作都属于 JS 线程内的操作
- 把 div1 放入页面之时
  - 浏览器会发现 JS 的意图
  - 就会通知渲染线程在页面中渲染 div1 对应的元素
- 把 div1 放入页面之后
  - 你对 div1 的操作都有可能会触发重新渲染
  - div1.id = 'newId' 可能会重新渲染，也可能不会
  - div1.title = 'new' 可能会重新渲染，也可能不会
  - 如果你连续对 div1 多次操作，浏览器可能会合并成一次操作，也可能不会（之前在动画里提到过）

属性同步

- 标准属性
  - 对 div1 的标准属性的修改，会被浏览器同步到页面中
  - 比如 id、className、title 等
- data-* 属性
  - 同上
- 非标准属性
  - 对非标准属性的修改，则只会停留在 JS 线程中
  - 不会同步到页面里
  - 比如 x 属性，示例代码
- 启示
  - 如果你有自定义属性，又想被同步到页面中，请使用 data- 作为前缀

图示


JS执行线程
div = {
id: 'test',
dataset: {
x: 'test'
},
x: 'test'
}
properties
自动同步
自动同步
渲染线程
<div
id="test"
data-x="test"
x="test"
>
attributes

Property v.s. Attribute

- property 属性
  - JS 线程中 div1 的所有属性，叫做 div1 的 property
- attribute 也是属性
  - 渲染引擎中 div1 对应标签的属性，叫做 attribute
- 区别
  - 大部分时候，同名的 property 和 attribute 值相等
  - 但如果不是标准属性，那么它俩只会在一开始时相等
  - 但注意 attribute 只支持字符串
  - 而 property 支持字符串、布尔等类型

下节课我们造一个 DOM 轮子

因为 DOM 默认的 API 太难用啦！