19 DOM 基本功

更新时间：2020-05-26 14:55:19

> 立志是事业的大门，工作是登堂入室的旅程。——巴斯德

在框架、工具琳琅满目的今天，很多同学可能会对 DOM 不屑一顾——比如说同样是 API，DOM API 能做的事情，相对于框架 API 来说太单薄、太无聊了。这么基础的玩意儿，学它干啥？

站在面试的角度看，其实是这么回事儿：如果各位的 target team 是小型创业团队，这类团队中有一些单纯业务导向的，他们的业务需求确实可能是比较急迫。对他们来说，主要目的是招到一个“干活的”。这种情况下，你能用熟一个轮子就 OK 了。但是如果你的目标是中型、大型公司的优质团队，或者是技术意识稍微强一些的小团队，你都无法在面试环节规避开面对官对 DOM 基础 和 DOM 底层原理的考察。

站在个人的角度，笔者也希望能在这里再次给大家强调一遍原生 JS 能力的重要性。DOM 基础说起来大家都知道一点，但能够系统掌握、面试时不露怯的人比大家想象中少太多太多了。笔者自己也万万想不到，曾经的“标配”基本功，在前端技术令人眼花缭乱的今天，竟然会被同学们抛诸脑后。如今，考察 React、Vue 相关的基础知识，甚至都没有考察原生 DOM 区分度高，实在令人唏嘘。

本节，我们首先会围绕 DOM 基础，给大家进行一轮知识点扫盲。如果在阅读完本节之后，你能够确保每个知识点都成竹于胸、问到不虚，那么下一章的《浏览器渲染原理》系列所涉及的 DOM 原理相关知识，一定会给予你更大的帮助和启发。

理解 DOM

> 文档对象模型（DOM）是HTML和XML文档的编程接口。它提供了对文档的结构化的表述，并定义了一种方式可以使从程序中对该结构进行访问，从而改变文档的结构，样式和内容。DOM 将文档解析为一个由节点和对象（包含属性和方法的对象）组成的结构集合。简言之，它会将web页面和脚本或程序语言连接起来。 —— MDN

DOM 的概念看似抽象，实则只是把 HTML 中各个标签定义出的元素以对象的形式包装起来。这样做的目的，就是确保开发者可以通过 JS 脚本来操作 HTML。

DOM 不仅是一套接口，更是一套规范

大家知道，网页是由浏览器来渲染出来的，浏览器一定程度上可以认为是我们源代码到展示结果这个过程中的一个编译器。

在这个世界上，有数不清的公司在做浏览器，进而也就有了数不清的浏览器品牌和型号。不同的浏览器之间存在着从内核到外壳的多重差异，如果说这些浏览器之间彼此没有办法达成共识，各出一套自己的标准的话，前端开发的适配工作量将是巨大的。

正是为了避免这种情况，才有了大家耳熟能详的 W3C 标准。W3C 中把许多行为的准则都统一掉了，而 DOM 作为 W3C 规范的一部分，约束了浏览器中 JS 与 HTML 间交互的方式。多亏有了 DOM，我们现在在任何一个浏览器上都可以用同一套 API 去操作 HTML，而不必关系浏览器底层的实现差异。

DOM 树的解析

DOM 结构以树的形态存在。在了解这棵树之前，我们首先应该认知的是树中的最小单元——节点。

在 DOM 中，每个元素都是一个节点，节点类型细数起来可以有很多种，我们这里强调以下 4 种：

Document

Document 就是指这份文件，也就是这份 HTML 档的开端。当浏览器载入 HTML 文档，它就会成为 **Document 对象**。

Element

Element 就是指 HTML 文件内的各个标签，像是 `<div>`、`<span>` 这样的各种 HTML 标签定义的元素都属于 Element 类型。

Text

Text 就是指被各个标籤包起来的文字，举个例子：

```
<span>哈哈哈</span>
```

这里的“哈哈哈”被 `<span>` 标签包了起来，它就是这个 Element 的 Text。

Attribute

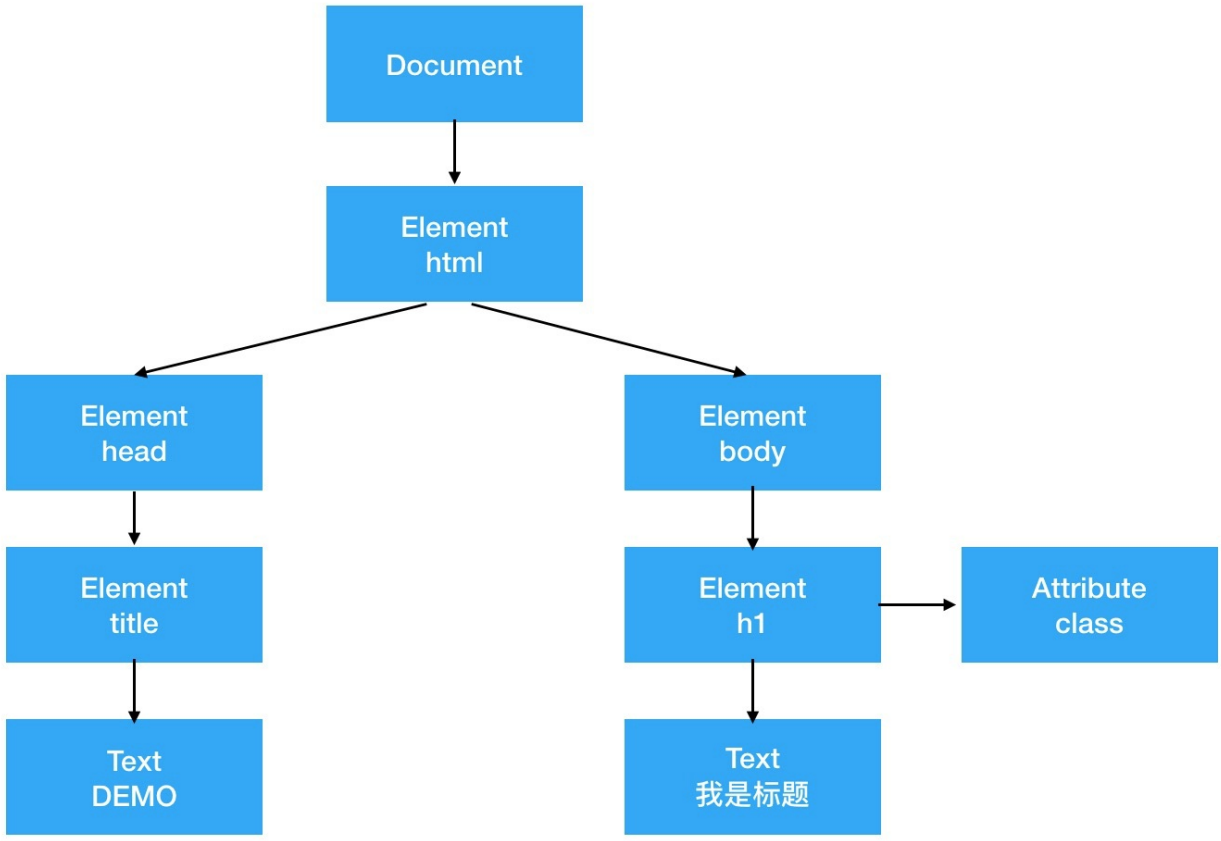
Attribute 类型表示元素的特性。从技术角度讲，这里的特性就是说各个标签里的属性。

一起来解析一棵 DOM 树

咱们现在用一个 demo 来直观表达一下上面所说的事项。尝试把下面这段代码转换成一棵 DOM 树：

```
<html>
  <head>
    <title>DEMO</title>
  </head>
  <body>
    <h1 class="title">我是标题</h1>
  </body>
</html>
```

在这个 DEMO 里，整个文件可以被看做是一个 document，根节点就是 HTML 这个 element。从 HTML 出发，往下走遇到了 head 和 body 这两个 element。在 head 内部，又划分出子节点 title 标签对应的 element，title 内部有一个 text 类型的子节点，值为“DEMO”；在 body 内部，又划分出子节点 h1，h1 内部也有一个 text 类型的子节点，值为“我是标题”；h1 元素同时具有一个 attribute 属性，即 class=“title” 这个键值对。我们把这一系列关系用树这种数据结构表示出来：

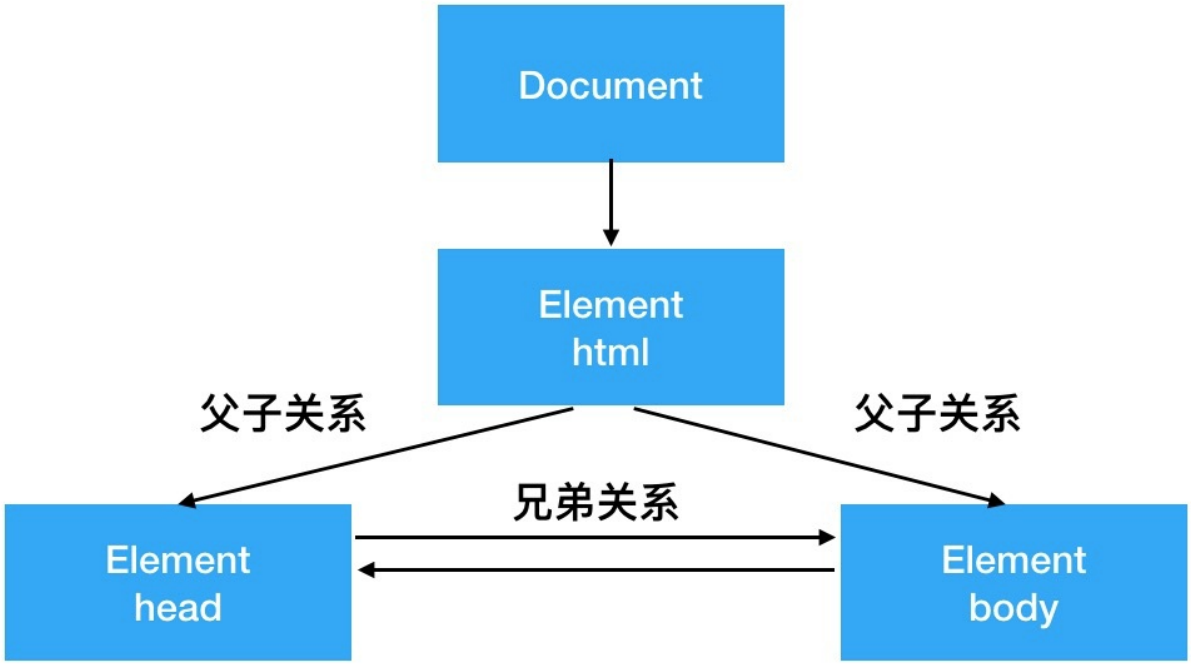


DOM 节点间关系

在树状结构的 DOM 里，节点间关系可以划分为以下两类：

- **父子节点**：表示节点间的嵌套关系
- **兄弟节点**：表示节点层级的平行关系，兄弟节点共享一个父节点

在上面这棵 DOM 树里，父子兄弟关系就可以表示如下：



常考操作一网打尽

DOM API常见操作，无非是增删改查。这块大家不用怕，背就完了。这里我为大家收敛了一部分面试时考察频率较高的 API：

查：DOM 节点的获取

DOM 节点的获取可以说是一类使用频率最高的操作，是各种复杂操作的基础。大家关注以下 API：

```
- getElementById // 按照 id 查询
- getElementsByTagName // 按照标签名查询
- getElementsByClassName // 按照类名查询
- querySelectorAll // 按照 css 选择器查询
```

一起来看看具体怎么用

```
// 按照 id 查询
var imooc = document.getElementById('imooc') // 查询到 id 为 imooc 的元素

// 按照标签名查询
var pList = document.getElementsByTagName('p')  // 查询到标签为 p 的集合
console.log(divList.length)
console.log(divList[0])

// 按照类名查询
var moocList = document.getElementsByClassName('mooc') // 查询到类名为 mooc 的集合

// 按照 css 选择器查询
var pList = document.querySelectorAll('.mooc') // 查询到类名为 mooc 的集合
```

增：DOM 节点的创建

真题：请你创建一个新节点，并把它添加到指定节点的后面。

假如题中已知的 HTML 结构如下：

```
<html>
  <head>
    <title>DEMO</title>
  </head>
  <body>
    <div id="container">
      <h1 id="title">我是标题</h1>
    </div>
  </body>
</html>
```

要求你添加一个有内容的 span 节点到 id 为 title 的节点后面，那么我们的做法就是：

```
// 首先获取父节点
var container = document.getElementById('container')

// 创建新节点
var targetSpan = document.createElement('span')
// 设置 span 节点的内容
targetSpan.innerHTML = 'hello world'

// 把新创建的元素塞进父节点里去
container.appendChild(targetSpan)
```

删：DOM 节点的删除

真题节选：删除指定的 DOM 节点

假如题中已知的 HTML 结构如下：

```
<html>
  <head>
    <title>DEMO</title>
  </head>
  <body>
    <div id="container">
      <h1 id="title">我是标题</h1>
    </div>
  </body>
</html>
```

需要删除 id 为 title 的元素，我们的做法是：

```
// 获取目标元素的父元素
var container = document.getElementById('container')
// 获取目标元素
var targetNode = document.getElementById('title')
// 删除目标元素
container.removeChild(targetNode)
```

或者通过子节点数组来完成删除：

```
// 获取目标元素的父元素
var container = document.getElementById('container')
// 获取目标元素
var targetNode = container.childNodes[1]
// 删除目标元素
container.removeChild(targetNode)
```

改：修改 DOM 元素

修改 DOM 元素这个动作可以分很多维度，比如说移动 DOM 元素的位置，修改 DOM 元素的属性等。

真题：将指定的两个 DOM 元素交换位置

假如题中已知的 HTML 结构如下：

```
<html>
  <head>
    <title>DEMO</title>
  </head>
  <body>
    <div id="container">
      <h1 id="title">我是标题</h1>
      <p id="content">我是内容</p>
    </div>
  </body>
</html>
```

现在需要你调换 title 和 content 的位置，我们可以考虑 insertBefore 或者 appendChild。这里我们给出 insertBefore 的操作示范：

```
// 获取父元素
var container = document.getElementById('container')

// 获取两个需要被交换的元素
var title = document.getElementById('title')
var content = document.getElementById('content')

// 交换两个元素，把 content 置于 title 前面
container.insertBefore(content, title)
```

真题：DOM 元素属性的获取和修改

仍然是上面这个 HTML 结构，现在假如需要你获取并修改 title 元素的 id 名，我们可以让 getAttribute 和 setAttribute 来帮忙：

```
var title = document.getElementById('title')

// 获取 id 属性
var titleId = title.getAttribute('id')

// 修改 id 属性
title.setAttribute('id', 'anothorTitle')
```

除了 getAttribute 和 setAttribute 外，我们通过访问 DOM 对象中提供给我们的属性名，也可以达成查询并修改某一些属性的目的。考虑到本文已针对类似的基础内容作了大量的阐述，故此处不再赘述。

结语

笔者创作专栏的初衷，是希望帮助大家用更舒服的姿势解决面试中的痛点、难点。在这样一套强调区分度的面试专栏里，穿插进像本节和下节这样极为基础的内容，我希望各位可以领悟到其中的用心——DOM 基本功，真的很重要。

本文的目的在于帮助大家明确 DOM 的本质、明确 DOM 的考察方式与权重。碍于篇幅和专栏特性的限制，笔者无法、也不会在此处充当一个 DOM API 的搬运工。但这绝不意味着你读完这么一篇总结性的文章、或者读完网上搜罗到的任何一篇 DOM API 科普，就可以高枕无忧了——DOM 操作是前端基础能力的重中之重，也是各位绝不可忽视的面试考点。

如果在读完本节后，你仍对自己的 DOM 基础感到不自信，想到面试考 DOM 就发怵、对 DOM 相关的考题心存侥幸而不是底气。那么强烈推荐你反复阅读《JavaScript——DOM编程艺术》这本书，对 DOM 基础做真正深入、全面、系统的学习。

再次强调，不要小看任何一块知识，最后决定你面试结果的或许恰恰就是曾经被你忽略的短板。

}

18 ES2015+考点集中解析

20 DOM事件体系（一）