# 问题

1. **js有哪些基本类型，每种基本类型是干啥的**
2. **对象是什么，函数是什么，数组是什么**
3. **数组，函数，对象的常用方法有哪些**

数组方法：

toString（）：将数组转换成数组值返回；

join（）：将数组转换成数组值返回，可指定中间符号；

push, pop, shift, unshift

索引从0开始

裁剪：slice

合并：concat：返回合并后的

拼接、删除：splice(atr1,atr2,’\*\*’): 返回已删除数组、atr1拼接的位置，atr2删除的数量

对象方法：

通过如下语法来访问对象方法：

objectName.methodName()

this 关键词

在 JavaScript 中，被称为 this 的事物，指的是拥有该 JavaScript 代码的对象。

this 的值，在函数中使用时，是“拥有”该函数的对象。

请注意 this 并非变量。它是关键词。您无法改变 this 的值。

1. **DOM是什么，我能用它做什么**

DOM 是一项 W3C (World Wide Web Consortium) 标准。

DOM 定义了访问文档的标准：

“W3C 文档对象模型（DOM）是中立于平台和语言的接口，它允许程序和脚本动态地访问、更新文档的内容、结构和样式。”

W3C DOM 标准被分为 3 个不同的部分：

* Core DOM - 所有文档类型的标准模型
* XML DOM - XML 文档的标准模型
* HTML DOM - HTML 文档的标准模型

HTML DOM是页面的文档对象模型

通过这个对象模型，JavaScript 获得创建动态 HTML 的所有力量：

* JavaScript 能改变页面中的所有 HTML 元素
* JavaScript 能改变页面中的所有 HTML 属性
* JavaScript 能改变页面中的所有 CSS 样式
* JavaScript 能删除已有的 HTML 元素和属性
* JavaScript 能添加新的 HTML 元素和属性
* JavaScript 能对页面中所有已有的 HTML 事件作出反应
* JavaScript 能在页面中创建新的 HTML 事件

1. **jquery是什么？**

# 要点

函数是对象，JavaScript 函数都有属性和方法。

定义为对象属性的函数，被称为对象的方法。

为创建新对象而设计的函数，被称为对象构造函数（对象构造器）。

**函数参数（parameter）**指的是在函数定义中列出的**名称**。

**函数参数（argument）**指的是传递到函数或由函数接收到的真实**值**。

## 参数通过值传递

函数调用中的参数（parameter）是函数的参数（argument）。

JavaScript 参数通过**值**传递：函数只知道值，而不是参数的位置。

如果函数改变了参数的值，它不会改变参数的原始值。

参数的改变在函数之外是不可见的。

## 对象是由引用传递的

在 JavaScript 中，对象引用是值。

正因如此，对象的行为就像它们通过**引用**来传递：

如果函数改变了对象属性，它也改变了原始值。

对象属性的改变在函数之外是可见的。

## HTML DOM：

**HTML DOM 方法是您能够（在 HTML 元素上）执行的动作。**

**HTML DOM 属性是您能够设置或改变的 HTML 元素的值。**

*element*.addEventListener(*event*, *function*, *useCapture*);

第一个参数是事件的类型（比如 "click" 或 "mousedown"）。

第二个参数是当事件发生时我们需要调用的函数。

第三个参数是布尔值，指定使用事件冒泡还是事件捕获。此参数是可选的。

**注意：**请勿对事件使用 "on" 前缀；请使用 "click" 代替 "onclick"。

### 事件冒泡还是事件捕获？

在 HTML DOM 中有两种事件传播的方法：冒泡和捕获。

事件传播是一种定义当发生事件时元素次序的方法。假如 <div> 元素内有一个 <p>，然后用户点击了这个 <p> 元素，应该首先处理哪个元素“click”事件？

在冒泡中，最内侧元素的事件会首先被处理，然后是更外侧的：首先处理 <p> 元素的点击事件，然后是 <div> 元素的点击事件。

在捕获中，最外侧元素的事件会首先被处理，然后是更内侧的：首先处理 <div> 元素的点击事件，然后是 <p> 元素的点击事件。

在 addEventListener() 方法中，你能够通过使用“useCapture”参数来规定传播类型：

## DOM节点

通过 JavaScript，您可以使用以下节点属性在节点之间导航：

* parentNode
* childNodes[*nodenumber*]
* firstChild
* lastChild
* nextSibling
* previousSibling

### nodeName 属性

nodeName 属性规定节点的名称。

* nodeName 是只读的
* 元素节点的 nodeName 等同于标签名
* 属性节点的 nodeName 是属性名称
* 文本节点的 nodeName 总是 #text
* 文档节点的 nodeName 总是 #document

### nodeValue 属性

nodeValue 属性规定节点的值。

* 元素节点的 nodeValue 是 undefined
* 文本节点的 nodeValue 是文本文本
* 属性节点的 nodeValue 是属性值

### nodeType 属性

nodeType 属性返回节点的类型。nodeType 是只读的。

最重要的 nodeType 属性是：

|  |  |  |
| --- | --- | --- |
| **节点** | **类型** | **例子** |
| ELEMENT\_NODE | 1 | <h1 class="heading">W3School</h1> |
| ATTRIBUTE\_NODE | 2 | class = "heading" （弃用） |
| TEXT\_NODE | 3 | W3School |
| COMMENT\_NODE | 8 | <!-- 这是注释 --> |
| DOCUMENT\_NODE | 9 | HTML 文档本身（<html> 的父） |
| DOCUMENT\_TYPE\_NODE | 10 | <!Doctype html> |

Type 2 在 HTML DOM 中已弃用。XML DOM 中未弃用。

### 创建、删除和替换节点

var para = document.createElement("p");

var node = document.createTextNode("这是新文本。");

前面例子中的 appendChild() 方法，追加新元素作为父的最后一个子。

除此之外您还可以使用 insertBefore() 方法：

parent.appendChild(child);

parent.insertBefore(child1, child2);

如需删除某个 HTML 元素，您需要知晓该元素的父：

parent.removeChild(child);

如需替换元素的，请使用 replaceChild() 方法：

parent.replaceChild(child1, child2);

### HTMLCollection

getElementsByTagName() 方法返回 **HTMLCollection** 对象。

HTMLCollection 对象是类数组的 HTML 元素列表（集合）。

可以通过索引来访问，HTMLCollection对象拥有length，但是不能用其他的数组方法如toString、pop、push等

### NodeList

**NodeList** 对象是从文档中提取的节点列表（集合）。

NodeList 对象与 HTMLCollection 对象几乎相同。

如使用 getElementsByClassName() 方法，某些（老的）浏览器会返回 NodeList 对象而不是 HTMLCollection。

所有浏览器都会为 childNodes 属性返回 NodeList 对象。

大多数浏览器会为 querySelectorAll() 方法返回 NodeList 对象。

## HTMLCollection 与 NodeList 的区别

HTMLCollection是 HTML 元素的集合。

NodeList 是文档节点的集合。

NodeList 和 HTML 集合几乎完全相同。

HTMLCollection 和 NodeList 对象都是类数组的对象列表（集合）。

它们都有定义列表（集合）中项目数的 length 属性。

它们都可以通过索引 (0, 1, 2, 3, 4, ...) 像数组那样访问每个项目。

访问 HTMLCollection 项目，可以通过它们的名称、id 或索引号。

访问 NodeList 项目，只能通过它们的索引号。

只有 NodeList 对象能包含属性节点和文本节点。