> 此试题由 kurryluo 整理，用于自学前端群成员
>
> 更多知识点：请访问 Github 项目：https://github.com/kurryluo/front-end-interview-guide
>
> 听说给了 star 的人都找到工作，升职加薪了~

JS

1. JS 语言的特点？和 C 语言、Java 语言的区别？
    - 特点：
        - 解释性语言；
        - 类似于 C 语言的语法结构；
        - 动态语言；
        - 基于原型的面向对象字面量的表现能力；
        - 函数式编程；
        - 安全性语言，它不被允许访问本地的硬盘
    - 区别：
        - JS 与 C 和 Java 所不同的一点在于，JS 是一种动态语言。从代码的角度看，动态语言的变量和函数是不指定返回值类型的；
        - C、JAVA 是强类型语言，需要预编译；JS 是弱类型的语言；
        - C、JAVA 是多线程运行的；JS 单线程，需要异步机制配合；
2. 原始类型有哪几种？null 是对象嘛？
    - 7 种:
        - Number
        - String
        - Boolean
        - Undefined
        - Null
        - Symbol：凡是属性名属于 Symbol 类型，就都是独一无二的，可以保证不会与其他属性名产生冲突
        - Bigint：比 Number 数据类型支持的范围更大的整数值。
    - Null 不是对象，但是用操作符 typeOf 检测出来是对象，因为一个美丽的错误：JS 存在的一个悠久 Bug ：在 JS 的最初版本中使用的是 32 位系统，为了性能考虑使用低位存储变量的类型信息，000 开头代表是对象，然而 null 表示为全零，所以将它错误的判断为 object 。虽然现在的内部类型判断代码已经改变了，但是对于这个 Bug 却是一直流传下来。
3. 对象类型和原始类型的不同之处？函数参数是对象会发生什么问题？
    - 在 JS 中，除了原始类型那么其他的都是对象类型了。对象类型和原始类型不同的是，原始类型存储的是值，对象类型存储的是地址（指针），因此，当我们修改对象类型数据的时候，就会修改存放在地址（指针）上的值，也就导致了两个拥有同样地址（指针的）变量都发生了改变。（被 “绿” 定理）
    - JS 函数参数都是按值传递，下面的例子可以说明，用反证法证明，假设 person 是按引用传递的参数，则在函数中 name 属性已经被赋值“greg”，所以结果应该为 greg 才对，而结果是 nick。

```
function setName(obj){
  obj.name = "nick";
  obj = new Object();
  obj.name = "greg";
}
var person = new Object();
setName(person);
alert(person.name);    //"nick"
```

4. typeof 是否能正确判断类型？instanceof 能正确判断对象的原理是什么？

- typeof 对于原始类型来说，除了 null 都可以显示正确的类型；
- typeof 对于对象来说，除了函数都会显示 object；
- 对于对象来说，可以考虑使用 instanceof，因为内部机制是通过原型链来判断的。
- 对于原始原始类型，也可以使用 instanceof 判断：

```
class PrimitiveString {
  static [Symbol.hasInstance](x) {
  return typeof x === 'string'
  }
}
console.log('kurryluo' instanceof PrimitiveString) // true
```

5. 转换规则：基本类型转换为布尔值？基本类型转换为数字？基本类型和引用类型转换为字符串？

- 转换为 Boolean。除了 Undefined， Null， False， NaN， ''， 0， -0，其他所有值都转为 true，包括所有对象。
- 转换为 Number。

	类型	参数值	转换结果
Number()	Boolean	TRUE	1
		FALSE	0
	Number	10	10
	Object	NULL	0
	-	Undefined	NaN
	String	"0123"/"0123"	123
		"1.1"	1.1
		"0xA"	10
		""	0
		"abc"	NaN
ParseInt()	String	"12345hhh"	12345
		""	NaN
		"0xA"	10
		"22.5"	22
		"070"	56(八进制)
		"70"	70(十进制)
		"0xf"	15(十六进制)
ParseFloat()	String	"12345hhh"	12345
		"0xA"	0
		"11.11"	11.11
		"11.11.11"	11.11
		"011.11.11"	11.11
		"3.14e4"	31400

- 转换为字符串。

类型	参数值	转换结果
Number	1	"1"
Null	null	"null"
Undefined	undefined	"undefined"
Boolean	true	"true"
BigInt	5n	"5"
Symbol	Symbol(1), Symbol(a)	"Symbol(1)", "Symbol([object Object])"
Object	{a:1}	"[object Object]"
Array	[1,2],[1],[]	"1,2", "1", ""
Function	function test(a){return a + 1}	"function test(a){return a + 1}"

CSS

1. CSS 的全称？加载 CSS 的方式？
   - Cascading Style Sheets，层叠样式表；
   - 内联样式：在 html 标签中直接写 css 样式，特殊情况下有效使用，存在弊端。
   - 通过 style 标签引入，写在 head 中，这样的样式表只能针对本页有效。
   - 通过 Link 引用 CSS：`<link rel ="styleshect" href="style.css">`，把 css 单独写到一个 css 文件内，然后在源代码中以 link 方式链接。好处是不但本页可以调用，其它页面也可以调用，是最常用的一种形式
   - 通过 @impoint 导入样式，一般常用在另一个表的内部。如 index.css 为主页所用的样式，那么我们可以把全局所需要用的公共样式放到一个 public.css 的文件中，这样就可以使代码起到，很好的重用性；

2. -webkit-，-moz-，-ms-，-o - 具体指什么了？

   浏览器私有前缀名。

   - -ms - 代表 IE 浏览器私有属性，代表 Microsoft
   - -moz - 代表 Firefox 浏览器私有属性，代表开发商：Mozilla，火狐浏览器全称叫做 Mozilla Firefox。
   - -webkit - 代表 chrome、safari 私有属性，代表 Webkit（引擎）内核
   - -o - 代表 Opera 私有属性，代表 Opera 浏览器

前缀	浏览器	内核
-ms-	IE	Trident
-moz-	Firefox	Trident 内核
-webkit-	Chrome 和 Safari	Webkit 内核
-o-	Opera	Presto 内核

3. CSS 选择器的优先级是如何计算的？

   浏览器通过优先级规则，判断元素展示哪些样式。优先级通过 4 个维度指标确定，我们假定以 `a、b、c、d` 命名，分别代表以下含义：

   - `a` 表示是否使用内联样式（inline style）。如果使用，`a` 为 1，否则为 0。
   - `b` 表示 ID 选择器的数量。
   - `c` 表示类选择器、属性选择器和伪类选择器数量之和。
   - `d` 表示标签（类型）选择器和伪元素选择器之和。

优先级的结果并非通过以上四个值生成一个得分，而是每个值分开比较。`a、b、c、d` 权重从左到右，依次减小。判断优先级时，从左到右，一一比较，直到比较出最大值，即可停止。所以，如果 `b` 的值不同，那么 `c` 和 `d` 不管多大，都不会对结果产生影响。比如 `0，1，0，0` 的优先级高于 `0，0，10，10`。

当出现优先级相等的情况时，最晚出现的样式规则会被采纳。如果你在样式表里写了相同的规则（无论是在该文件内部还是其它样式文件中），那么最后出现的（在文件底部的）样式优先级更高，因此会被采纳。

4. 盒子模型？IE 盒子和标准盒子的区别？

   CSS 盒模型描述了以文档树中的元素而生成的矩形框，并根据排版模式进行布局。

   盒子包括四个部分：

   content、padding、border、margin。

   但是浏览器在我们选中一个元素的时候，一般显示的宽度是：

   content 宽度 + padding（左、右）+ border（左、右）。

   所以我们会看到设置了 content 宽度一样的两个元素，最终可能因为 padding、border 的不一样而显得大小不一。

   这个时候有个属性可以帮助我们解决：

   box-sizing: content-box | border-box | ~~padding-box~~(已废弃)

   她有三个属性，决定了我们设置的 “width 宽度” 到底包含盒子模型的哪些成分，从字面意思上就能看出来：

   - content-box：我们设置的 width 就只是内容（content）的宽度，如果要正确计算整个元素的宽度，还得加上 padding 和 border。
   - border-box：我们设置的 width 是内容（content） + padding + border 。

   这样设置后的元素的位置在视觉上，可能会发生改变，因为 padding 透明的。这个也是实现内容居中的一个方式。

   - 标准盒子模型（默认盒子，相当于设置了属性 content-box）：宽度 = 内容的宽度（content）+ border + padding + margin
   - 低版本 IE 盒子模型（相当于设置了属性 border-box）：宽度 = 内容宽度（content+padding + border）+ margin

5. 请阐述块格式化上下文（Block Formatting Context）及其工作原理，哪些设置可以生成 BFC，BFC 的应用？

   块格式上下文（BFC）是 Web 页面的可视化 CSS 渲染的部分，是块级盒布局发生的区域，也是浮动元素与其他元素交互的区域。

   一个 HTML 盒（Box）满足以下任意一条，会创建块格式化上下文：

   - `float` 的值不是 `none`.
   - `position` 的值不是 `static` 或 `relative`.
   - `display` 的值是 `table-cell`、`table-caption`、`inline-block`、`flex`、或 `inline-flex`。
   - `overflow` 的值不是 `visible`。

   在 BFC 中，每个盒的左外边缘都与其包含的块的左边缘相接。

   两个相邻的块级盒在垂直方向上的边距会发生合并（collapse）。

HTML

1. HTML 全称是什么？在前端中的作用。

HyperText Markup Language， 超文本标记语言，是一种用于创建网页的标准标记语言。借助标签，结构化内容，便于搜索引擎抓取。

2. DOCTYPE 有什么用？

   `DOCTYPE` 是 “document type” 的缩写。它是 HTML 中用来区分标准模式和 怪异模式 的声明，用来告知浏览器使用标准模式渲染页面。

3. 举出 10 个常用的标签， 5 个 H5 新标签。

   - 10 个常用的：
     - div
     - span
     - a
     - head
     - body
     - script
     - img
     - button
     - input
     - ul
   - 5 个 H5 新标签
     - article
     - header
     - nav
     - footer
     - canvas

4. meta 标签的作用？

   meta 标签提供了 HTML 文档的元数据。元数据不会显示在客户端，但是会被浏览器解析。通常用于指定网页的描述，关键词，文件的最后修改时间，作者及其他元数据。这些元数据可以指导浏览器如何显示内容或重新加载页面，是否响应式加载，以及更好地让搜索引擎（关键词）分析网页内容，或其他 Web 服务调用。

5. 如何提供包含多种语言内容的页面？在设计开发多语言网站时，需要留心哪些事情？

   - 在 HTML 中使用 `lang` 属性。
   - 引导用户切换到自己的母语——让用户能够轻松地切换到自己的国家或语言，而不用麻烦。
   - 在图片中展示文本会阻碍网站规模增长——把文本放在图片中展示，仍然是一种非常流行的方式。这样做可以在所有终端上，都能显示出美观的非系统字体。然而，为了翻译图片中的文本，需要为每种语言单独创建对应的图片，这种做法很容易在图片数量不断增长的过程中失控。
   - 限制词语或句子的长度——网页内容在使用其他语言表述时，文字长度会发生变化。设计时，需要警惕文字长度溢出布局的问题，最好不要使用受文字长度影响较大的设计。比如标题、标签、按钮的设计，往往很受文字长度影响，这些设计中的文字与正文或评论部分不同，一般不可以自由换行。
   - 注意颜色的使用——颜色在不同的语言和文化中，意义和感受是不同的。设计时应该使用恰当的颜色。
   - 日期和货币的格式化——日期在不同的国家和地区，会以不同的方式显示。比如美国的日期格式是“May 31, 2012”，而在欧洲部分地区，日期格式是“31 May 2012”。
   - 不要使用连接的翻译字符串——不要做类似这样的事情，比如 `“今天的日期是”+ 具体日期`。这样做可能会打乱其他语言的语序。替代方案是，为每种语言编写带变量替换的模版字符串。请看下面两个分别用英语和中文表示的句子：`I will travel on {% date %}` 和 `{% date %} 我会出发`。可以看到，语言的语法规则不同，变量的位置是不同的。

- 注意语言阅读的方向——在英语中，文字是从左向右阅读的；而在传统日语中，文字是从右向左阅读的。