02-JS基础-常⻅⾯试题

# 说出 JavaScript 中的常⻅数据类型

Number -> 代表数据类型是数值整数

浮点

数组

String -> 代表字符串类型 -> 通常是⼀段⽂本

boolean -> 布尔类型

true false

NULL -> 空值

undeﬁned -> 变量未定义 Object -> 对象类型 BigInt -> ⼤整数类型 Symbol -> 符号类型

# 说出count++和++count的区别?

如果它们在计算之后的值不会被使⽤ 是没有区别的

如果需要⽴刻使⽤⾃增之后的值 就⽤前置型 反之后置型

var num = 1 + ++count // 不推荐这么写

//需要⽴刻使⽤的话

count++

var num = 1 +count

//++count

//不需要的话

var num = 1 + count //count++

count++

# 说出==和===的区别?

== (普通相等)

在类型不相同的情况下, 会将运算元先转成Number的值, 再进⾏⽐较(隐式转换)

null⽐较特殊: null在进⾏⽐较的时候, 应该是会被当成⼀个对象和原⽣类型进⾏⽐较的

=== (严格不等)

在类型不同的情况下,直接返回false

# 逻辑与&&和逻辑或||的本质和区别？

## 逻辑与:也叫作短路与

从左往右,依次计算

当计算第⼀个运算元,先隐式转换为Boolean值进⾏⽐较

true ,继续下⼀个⽐较

false ,直接返回该运算元的初始值

如果找到最后也没有找到,就返回最后⼀个运算元

## 逻辑或:也叫做短路或

从左往右,依次计算

当计算第⼀个运算元,先隐式转换为Boolean值进⾏⽐较

true ,直接返回该运算元的初始值 false ,继续下⼀个⽐较

如果找到最后也没有找到,就返回最后⼀个运算元

# 说说什么是函数回调？什么匿名函数？

回调函数：

⼀个函数作为另外⼀个函数的参数，称之为回调函数，也称为⾼阶函数匿名函数：

如果在传⼊⼀个函数时，我们没有指定这个函数的名词或者通过函数表达式指定函数对应的变量，那么这个函数称之为匿名函数

# 说出对DOM和document对象的理解

DOM:⽂档对象模型(Document Object Model)将⻚⾯所有的内容表示为可以修改的对象浏览器将我们编写在HTML中的每⼀个元素(Element)都抽象成了⼀个个对象

所有这些对象都可以通过JavaScript来对其进⾏访问，那么我们就可以通过JavaScript来操作

⻚⾯;

所以，我们将这个抽象过程称之为 ⽂档对象模型(Document Object Model) Document节点表示的整个载⼊的⽹⻚，它的实例是全局的document对象:

对DOM的所有操作都是从 document 对象开始的

它是DOM的⼊⼝点，可以从document开始去访问任何节点元素;

# 说说常⻅的节点（Node）属性

nodeType

获取节点的类型

⽐如 注释节点8 ⽂本节点3 元素节点1

tagName

获取元素的标签名词 仅适⽤于Element节点

nodeName

获取元素的标签名词 适⽤于任何Node节点

innerHTML,textContent

前者将元素中的HTML获取为字符串属性 后者仅仅获取⽂本内容

outerHTML

包含了完整的HTML

相当于innerHTML加上元素本身

nodeValue/data

获取⾮元素节点的⽂本内容

hidden

⽤于设置元素隐藏(全局属性)

# 说说attribute和Property的区别和关系

attribute

浏览器解析HTML元素时 会将对应的属性(attribute)放在对应的元素对象上具体分为标准的属性和⾮标准的属性

标准属性: id class href type value等等

⾮标准属性(⾃定义) :abc age height

Property

对于标准的attribute 会在DOM对象上创建对应的property属性

⼤多数情况下 他们是相互作⽤的 改变其中⼀个 另⼀个也会随之改变

⼤多数情况 推荐获取attribute 使⽤property⽅式 因为它默认是有类型的

# 说说事件冒泡和事件捕获的理解?

事件冒泡:

从最内层的元素向外依次传递的顺序, 默认是事件冒泡事件捕获:

从外层到内层依次传递的顺序 ,可以通过addEventListener("click",fn,true) 监听事件捕获

# 说说你对事件委托的理解?

利⽤事件的冒泡机制，以及事件对象中可以准确获知触发事件的元素机制(e.target)，将⼦元素事件委托给⽗元素处理的现象。

# 说说load和DOMContentLoaded的区别

load

浏览器加载完所有的HTML 还加载完所有的外部资源 样式 图⽚等

DOMContentLoaded

HTML⽂档所有资源都加载完成 并构建了DOM树 但是⼀些外部资源还没有加载完成 如图⽚的src

# 说说JSON的相关⽤法和应⽤场景？

三种⽤法

简单值: 数字 字符串 布尔类型

对象值 key value组成 必须添加双引号 value 可以是简单值 对象值 数组值数组值 内容可以是对象值 简单值 数组值

应⽤场景

⽹络传输的JSON数据项⽬的某些配置⽂件

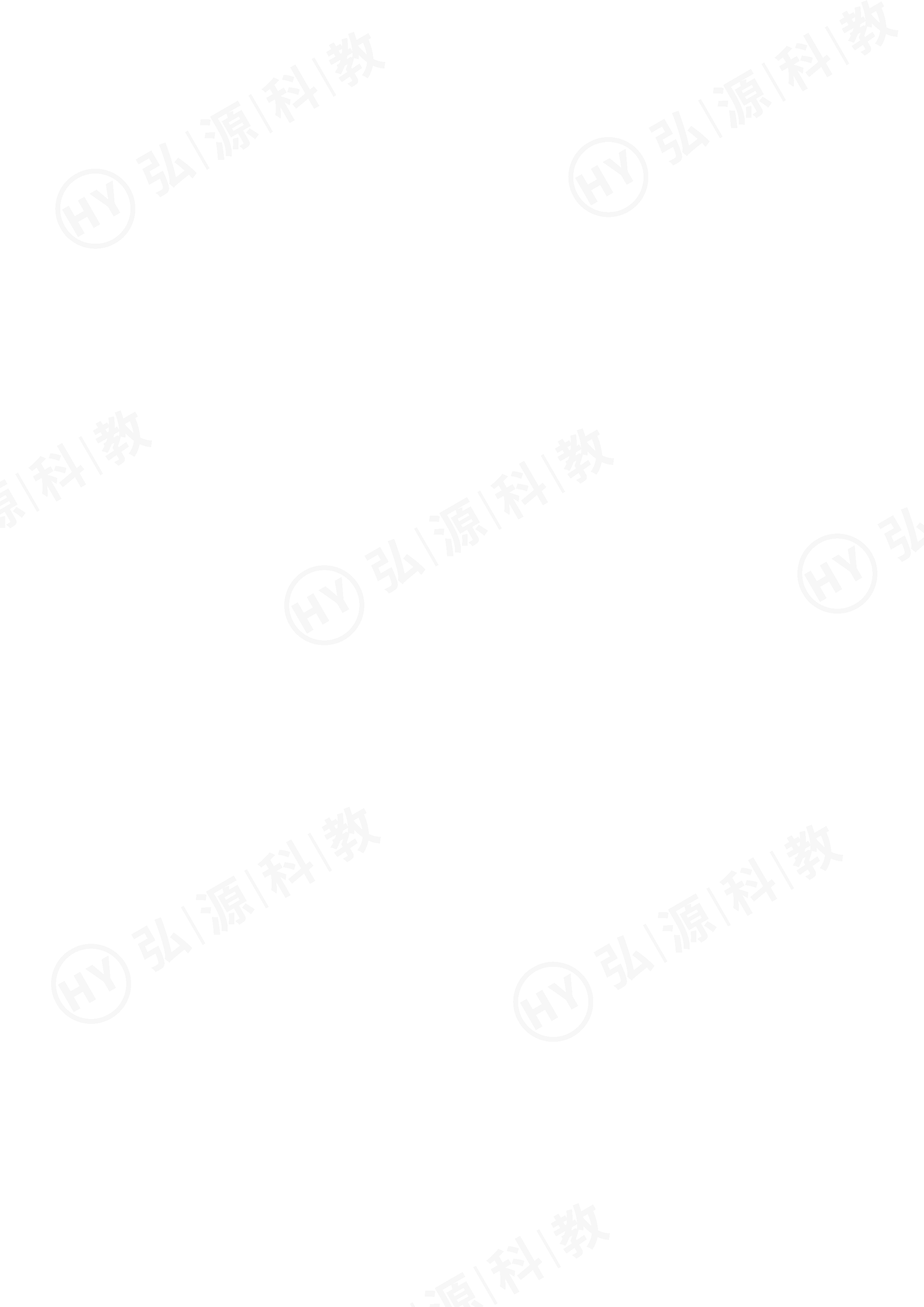
⾮关系型数据库将JSON作为存储⽂件

# prefetch 与 preload 的区别是什么?

<link rel="prefetch" href="style.css" as="style" />

<link rel="preload" href="main.js" as="script" />

**preload** 是⼀个声明式 fetch，可以**强制浏览器在不阻塞 document 的 onload 事件的情况下请求资源**。 preload 顾名思义就是⼀种预加载的⽅式，它通过声明向浏览器声明⼀个需要提前加载的

资源，当资源真正被使⽤的时候⽴即执⾏，就⽆需等待⽹络的消耗。

**prefetch** 告诉浏览器**这个资源将来可能需要**，但是什么时间加载这个资源是由浏览器来决定的。若能预测到⽤户的⾏为，⽐如懒加载，点击到其它⻚⾯等则相当于提前预加载了需要的资源。

# preload和prefetch的应⽤场景？

## webpack优化之preload和prefetch：

单⻚⾯应⽤由于⻚⾯过多，可能会导致代码体积过⼤，从⽽使得⾸⻚打开速度过慢。所以**切分代码，优化⾸屏打开速度尤为重要**。

但是所有的技术⼿段都不是完美的。当我们切割代码后，⾸屏的js⽂件体积减少了好多。但是也有⼀个突出的问题：

那就是当跳转其他⻚⾯的时候，需要下载相应⻚⾯的js⽂件，这就导致体验极其不好，每⼀次点击访问新⻚⾯都要等待js⽂件下载，然后再去请求接⼝获取数据。频繁出现loading动画的体验真的不好

所以如果我们在进⼊⾸⻚后，在浏览器的空闲时间提前下好⽤户可能会点击⻚⾯的js⽂件，这样⾸屏的js

⽂件⼤⼩得到了控制，⽽且再点击新⻚⾯的时候，相关的js⽂件已经下载好了，就不再会出现loading动画。

## 动态引⼊js⽂件，实现code-splitting，减少⾸屏打开时间

按引⼊情况加载，只需添加注释即可

代码分割注释：/*webpackChunkName: 'mp-supports'*/ prefetch注释：/\* webpackPrefetch: true \*/

更多的，可以查看 webpack 注释⿊魔法：https://webpack.js.org/api/module-methods/#magic-com ments

使⽤案例

const { default: lodash } = await import(/\* webpackChunkName: "lodash" \*/ /\* webpackPrefetch: true \*/ 'lodash');

// Multiple possible targets import(

/\* webpackInclude: /\.json$/ \*/

/\* webpackExclude: /\.noimport\.json$/ \*/

/\* webpackChunkName: "my-chunk-name" \*/

/\* webpackMode: "lazy" \*/

/\* webpackPrefetch: true \*/

/\* webpackPreload: true \*/

`./locale/${language}`

);

# 如何实现⻚⾯⽂本不可复制

有 CSS 和 JS 两种⽅法，以下任选其⼀或结合使⽤使⽤ CSS 如下：

user-select: none;

或使⽤ JS 如下，监听

selectstart

当⽤户选中⼀⽚区域时，将触发即可实现⻚⾯⽂本不可复制。

事件，禁⽌选中。

事件，Selection API 将会选中⼀⽚区域。禁⽌选中区域

selectstart

document.body.onselectstart = (e) => { e.preventDefault();

};

document.body.oncopy = (e) => { e.preventDefault();

};

# DOM 中 Element 与 Node 有何区别

Element 继承于 Node，具有 Node 的⽅法，同时⼜拓展了很多⾃⼰的特有⽅法。

# 点击 a 标签下载⽂件如何做？

有两种⽅式:

当指定 a 标签的属性

a.download

download

download

属性时，点击该链接会直接保存为⽂件，⽂件名为

通过对 a 标签指定的 URL 在服务器设置响应头

Content-Disposition: attachment;

可直接下载

filename="filename.jpg"