# 项目大总结

### 项目中的语法:

document.getElementById()

document.querySelector()

addEventListener（）

document.body.style.backgroundColor

Math.floor()

Math.random()

array.length

document.querySelectorAll()

forEach()

addEventListener()

currentTarget property

classList

textContent

objects

DOMContentLoaded

classList.toggle()

classList.remove()

classList.add()

arrays

objects

map, reduce, and filter

innerHTML

includes method

classList.contains()

play()

pause()

getFullYear()

getBoundingClientRect()

slice method

window.scrollTo()

getMonth()

getDate()

setInterval()

clearInterval()

parseInt()

isNaN()

slice method

event.preventDefault()

new Date()

createAttribute()

setAttributeNode()

appendChild()

filter()

map()

localStorage()

### 正则表达式：

/：正则表达式的开始和结束符号

^：匹配字符串的开头

[ ]:匹配方括号内的任意字符

^\s@ 匹配非空格和非@字符

+：匹配前面的字符一次或多次

@：匹配@符号

\.匹配点号

$:匹配字符串的结尾

.:匹配除换行符以外的任意字符

\*:匹配前面的字符零次或多次

?:匹配前面的字符零次或一次

{m,n}:匹配前面的字符至少m次，至多n次

|:匹配两个或多个分支中的一个

():将括号内的内容作为一个整体进行匹配

\d:匹配数字字符，等价于[0-9]

\w:匹配字母,数字或下划线，等价于[A-Za-z0-9]

\s：匹配空白字符，包括空格，制表符，换行符等等；

/：正则表达式的开始和结束符号。

^：匹配字符串的开头。

(?=)：正向肯定预查，用于检查字符串中是否包含括号内的内容，但不捕获匹配内容。

.\*：匹配任意字符零次或多次。

[a-z]：匹配小写字母。

[A-Z]：匹配大写字母。

\d：匹配数字字符，等价于 [0-9]。

$：匹配字符串的结尾。

.test(pass)：使用正则表达式的 test 方法检查 pass 是否符合正则表达式定义的规则。

更多的正则表达式可在MDN查看。

match：方法检索返回一个字符串匹配正则表达式的结果记住只会返回第一个满足表达式结果只是他这里用到了g全局匹配全部返回。

例：

const paragraph = 'The quick brown fox jumps over the lazy dog. It barked.';

const regex = /[A-Z]/g;

const found = paragraph.match(regex);

console.log(found);

// Expected output: Array ["T", "I"]

Match和test得到区别位置不一样

'string'.match(/regex/);/regex/.test('string');

let quoteSample = "Beware of bugs in the above code; I have only proved it correct, not tried it.";

let vowelRegex = /[aeiou]/gi; // Change this line

let result = vowelRegex.test(quoteSample); // Change this line

let chewieQuote =  "Aaaaaaaaaaaaaaaarrrgh!";

// Only change code below this line

let chewieRegex = /Aa\*/; // Change this line

// Only change code above this line

let c = /Aa+/g

let result = chewieQuote.match(chewieRegex);

let results = chewieQuote.match(c);

console.log(result);

console.log(results);

[

'Aaaaaaaaaaaaaaaa',

index: 0,

input: 'Aaaaaaaaaaaaaaaarrrgh!',

groups: undefined

]

[ 'Aaaaaaaaaaaaaaaa' ]

正则表达式默认是贪婪模式如果不想贪婪模式可以在表达式后面打上？

不捕获匹配内容的意思是，正则表达式会检查字符串中是否包含括号内的内容，但不会将匹配到的内容作为结果返回。

举个例子，假设我们有一个字符串 abc123，我们想检查它是否包含数字。我们可以使用正则表达式 /\d/ 来进行匹配。如果使用 match 方法，它会返回匹配到的内容，即 ['1']。

但是，如果我们使用正向肯定预查，即 /(?=\d)/，它只会检查字符串中是否包含数字，但不会返回匹配到的内容。如果使用 test 方法，它只会返回一个布尔值，表示字符串中是否包含数字。

### CSS语法：

**box-sizing: border-box; 是一种CSS属性，用于控制元素的盒模型尺寸行为。默认情况下，当你设置一个元素的宽度或高度时，它不包括内边距和边框的尺寸。这意味着元素的总尺寸可能会超过指定的宽度或高度，导致布局问题。**

**box-sizing: border-box; 应用于元素，你改变了其尺寸的计算方式。使用这个属性时，指定的宽度和高度值将包括内边距和边框，元素内的任何内容将被调整以适应这些尺寸。**

### HTML：

<b>, <strong>, <em>, <i> 是HTML中常用的用于标记文本样式和语义的标签。虽然它们在视觉效果上可能相似，但它们在语义和用途上有一些区别。

<b>标签（Bold）是用于将文本加粗显示的标签，仅仅表示文本的粗体外观，没有特定的语义含义。它在视觉上改变文本的加粗样式，但不提供额外的语义信息。一般情况下，建议使用更具语义的标签，如<strong>。

<strong>标签是用于表示重要性或强调的文本内容。它具有强调的语义含义，表示文本的重要性。浏览器会默认对<strong>标签中的文本应用加粗样式，但其主要作用是强调文本的语义，而不仅仅是视觉上的样式。

<em>标签是用于表示强调的文本内容。它表示文本的强调语义，通常会以斜体显示。<em>标签可以用于强调一些关键词、短语或句子，使其在语义上更具有重要性。

<i>标签是用于表示斜体文本的标签。它通常用于标记文本的引用、技术术语、外语短语或其他需要斜体强调的内容。然而，<i>标签在HTML5中失去了强调的语义含义，而更多地被视为样式标签。在强调文本的语义方面，推荐使用<em>标签。

总结一下它们的区别：

<b>标签只改变文本的视觉效果，表示文本的加粗样式。

<strong>标签用于表示文本的重要性或强调，具有强调的语义含义，并默认显示为加粗样式。

<em>标签用于表示文本的强调，通常以斜体显示，具有强调的语义含义。

<i>标签用于表示文本的斜体显示，但在语义上已经失去了强调的含义，更多地被视为样式标签。

在实际应用中，应根据文本的语义和目的来选择合适的标签，以确保文本样式与其语义含义一致。

<del>标签是HTML中用于表示删除的文本内容的标签。它用于标记已经被删除或不再有效的文本。

<del>标签的使用会导致被包围的文本显示为删除线，并且在某些浏览器中还可能应用其他样式，以突出显示删除的文本。

<del>标签是HTML中用于表示删除的文本内容的标签。它用于标记已经被删除或不再有效的文本。

<del>标签的使用会导致被包围的文本显示为删除线，并且在某些浏览器中还可能应用其他样式，以突出显示删除的文本。

<ins>标签用于表示插入的文本内容，即在文本中插入的新内容。它常与<del>标签配对使用，<del>用于表示被删除的文本。<ins>标签可以在文本中以不同的样式呈现，以指示插入的文本。

**<small>标签用于表示较小或次要的文本内容。它可以将包含在其中的文本显示为较小的字体尺寸。<small>标签并没有明确的语义含义，但通常用于表示辅助性的或次要的信息。**

**<mark>标签用于突出显示或标记重要的文本，通常以黄色背景显示。**

**<ins>标签用于表示插入的文本内容，常与<del>标签配对使用。**

**<small>标签用于表示较小或次要的文本内容。**

**<sup>标签用于表示上标文本，将其中的文本显示为相对基线上方的上标形式。通常用于表示指数、上标数字、科学符号等。**

**<sup>标签用于表示上标文本，将其中的文本显示为相对基线上方的上标形式。通常用于表示指数、上标数字、科学符号等。**

**<iframe>标签是HTML中用于嵌入另一个HTML文档或外部网页的标签。它可以在当前页面中创建一个内联的框架，用于显示其他网页内容或嵌入其他HTML文档。**

**<iframe>标签通过指定一个src属性来确定要显示的内容，该属性可以是一个URL，指向外部网页或其他HTML文档。浏览器会加载指定URL的内容，并在当前页面中呈现出来**

**<iframe>标签是HTML中用于嵌入另一个HTML文档或外部网页的标签。它可以在当前页面中创建一个内联的框架，用于显示其他网页内容或嵌入其他HTML文档。**

**<iframe>标签通过指定一个src属性来确定要显示的内容，该属性可以是一个URL，指向外部网页或其他HTML文档。浏览器会加载指定URL的内容，并在当前页面中呈现出来**

**text-decoration-line 是一个 CSS 属性，用于设置文本装饰线的样式。**

**该属性可以接受以下值：**

**none：表示没有文本装饰线。**

**underline：表示文本下方有一条线。**

**overline：表示文本上方有一条线。**

**line-through：表示文本中间有一条线穿过。**

**underline overline：表示文本同时有上方和下方的线。**

**underline line-through：表示文本同时有下方的线和中间的线。**

**overline line-through：表示文本同时有上方的线和中间的线。**

**underline overline line-through：表示文本同时有上方、下方和中间的线。**

**除了 text-decoration-line 属性外，还有其他用于控制文本装饰的 CSS 属性。这些属性可用于定义文本的下划线、删除线、上划线等效果，以及设置它们的颜色、样式和粗细。**

**以下是一些常用的文本装饰相关属性：**

**text-decoration-color：用于设置文本装饰线的颜色。**

**text-decoration-style：用于设置文本装饰线的样式，如实线、虚线、双实线等。**

**text-decoration-thickness：用于设置文本装饰线的粗细。**

**text-decoration-skip-ink：用于控制文本装饰线是否避开 Ink（墨水）。**

**text-underline-offset：用于设置文本下划线的偏移量。**

**text-underline-position：用于设置文本下划线的位置，如位于文本的基线或文字下方。**

**text-decoration：用于同时设置文本装饰线的多个属性，如颜色、样式和粗细。**

**这些属性可以单独使用或结合在一起，以实现不同的文本装饰效果。通过调整这些属性的值，可以实现自定义的文本装饰样式。**

### 语法理解：

Math.random ： 遇到的一个坑我直接拿这个去加要随机的长度，但实际是乘（随机函数）,Math.floor（向下取整），Math.round（四舍五入），Math.ceil（向上取整）

currentTarget : 当事件被触发时，他会到父元素直到没有父元素的节点，也就是他包括所有这个事件触发的节点，

Target ： 当事件触发时target表示触发该事件的元素，再冒泡阶段中他指向最初的元素

在事件处理程序内部，this 对象始终等于 currentTarget 的值，而 target 只包含事件的实际

目标。这点一定要记住！！！！！

Contains ： 判断variable.contains(“这里面的类名是否在变量中”)

textContent：跟innerHTML一样设置文本内容的标准方式，但是innerHTML不严谨所有现在用的还是textContent

Style:主要用于访问和修改DOM元素的内联样式（inline style），即直接应用于元素本身的样式属性。

Toggle：如果类名列表中已经存在指定的 value，则删除；如果不存在，则添加。（这是个好语法切换用的）

classList：属性如下：

add(value)，向类名列表中添加指定的字符串值 value。如果这个值已经存在，则什么也不做。

 contains(value)，返回布尔值，表示给定的 value 是否存在。

 remove(value)，从类名列表中删除指定的字符串值 value。

 toggle(value)，如果类名列表中已经存在指定的 value，则删除；如果不存在，则添加。

Function getRandNumber(){

Return Math.floor(Math.random \* (指定的长度))；

}

strictButton.addEventListener(‘click’,(event) => {

If(strictButton.checked === true){ //检查这个按钮是否选上

Strict = true;

}else{

Strict = false;

}

})

**sessionStorage：**

**localStorage：**

用户登录系统遇到的坑：

New bing 的解答：你在注册界面中使用的键是list，但是在登录界面中使用的键是mylist，这两个键不一致，所以你无法获取到之前存储的数据。你应该统一使用一个键，比如mylist。

你在注册界面中存储的对象属性是vlaue，但是在登录界面中比较的属性是value，这两个单词拼写不一样，所以你无法正确地判断用户名和密码是否匹配。你应该改正拼写错误，统一使用value。

你在登录界面中遍历数组时，使用了forEach方法，但是这个方法不能返回值，所以你的函数永远会返回false。你应该使用some方法来判断数组中是否有满足条件的元素，如果有就返回true，如果没有就返回false。

你在登录界面中定义了两个相同的函数nextHtml，这是没有必要的，而且会造成混乱。你应该只保留一个函数，用来跳转到注册界面。😐

DB各个浏览器的支持语法：

const indexedDB = window.indexedDB ||

window.mozIndexedDB ||

window.webkitIndexedDB ||

window.msIndexedDB ||

window.shimIndexedDB;

const request = indexedDB.open("carsDatabase",1);

request.onerror = function(event){

console.error("An error occurred with IndexedDB");

console.error(event);

};

IndexedDB:

open:创建数据库里面的第一个参数是数据库名字，第二个参数是数据库版本

数据存在的话会发送一个打开它的请求；如果不存在也会发送并创建请求

如果 onsuccess

事件处理程序被调用，说明可以通过 event.target.result 访问数据库（IDBDatabase）实例

创建完数据库然后会触发upgradeneeded事件

可以在里面更新数据在这里面，我们可以创建和删除对象存储空间以及构建和删除索引。例如？

 // 建立一个对象仓库来存储我们客户的相关信息，我们选择 ssn 作为键路径（key path）

  // 因为 ssn 可以保证是不重复的

    let objectStore = db.createObjectStore('customers',{keyPath:'id'});

    // 建立一个索引来通过姓名来搜索客户。名字可能会重复，所以我们不能使用 unique 索引

    objectStore.createIndex('name','name',{unique:false});

    // 使用邮箱建立索引，我们向确保客户的邮箱不会重复，所以我们使用 unique 索引

  objectStore.createIndex("email", "email", { unique: true });

增加、读取和删除数据：  
事务提供了三种模式：readonly、readwrite 和 versionchange。

使用 IDBDatabase.transaction (en-US) 启动一个事务。该方法接受两个参数：storeNames (作用域，一个你想访问的对象仓库的数组），事务模式 mode（readonly 或 readwrite）。该方法返回一个包含 IDBIndex.objectStore (en-US) 方法的事务对象，使用 IDBIndex.objectStore (en-US) 你可以访问你的对象仓库。未指定 mode 时，默认为 readonly 模式。

e.stopPropagation（） 阻止附加到上述标签的事件处理程序触发。

e.preventDefault（） 阻止具有它的默认浏览器行为的少数事件。