

**مـکـتب ۸۱**

**ســری نهم**



1. **بخش JavaScript**
   1. با استفاده از متد at، تابعی بنویسید که یک ورودی را از کاربر گرفته و به آن تعداد، برای آرایه ورودی تعریف کند و سپس، مقلوب بودن آن آرایه را بررسی کند. به عنوان مثال، آرایه زیر مقلوب است:

const arr = [1,2,3,2,1]

* 1. تابع filterRange را به گونه ای بنویسید که یک آرایه دریافت کند، به دنبال عناصری با مقادیر بالاتر یا مساوی a و کمتر یا مساوی b باشد و نتیجه را به عنوان یک آرایه برگرداند. توجه داشته باشید که تابع نباید آرایه را تغییر دهد و آرایه جدید را برگرداند. مثال:

let arr = [5, 3, 8, 1];

let filtered = filterRange(arr, 1, 4);

alert( filtered ); // 3,1 (matching values)

alert( arr ); // 5,3,8,1 (not modified)

1-3- تابع filterRangeInPlace را به گونه ای بنویسید که یک آرایه دریافت کند و تمام مقادیر را به جز مقادیری که بین a و b هستند از آن حذف کند.

توجه داشته باشید که تابع فقط باید آرایه را اصلاح کند و نباید چیزی را برگرداند.

مثال:

let arr = [5, 3, 8, 1];

filterRangeInPlace(arr, 1, 4); // removed the numbers except from 1 to 4

alert( arr ); // [3, 1]

-1-4 یک سازنده Calculator ایجاد کنید که یک شی محاسبه گر قابل توسعه را ایجاد می کند.

پیاده سازی شامل دو بخش است:

ابتدا، متد calcul را پیاده سازی کنید که رشته ای مانند "1 + 2" را در قالب

"NUMBER operator NUMBER" می گیرد و نتیجه را برمی گرداند. باید + و منهای - را درک کند. به عنوان مثال:

let calc = new Calculator;

alert( calc.calculate("3 + 7") ); // 10

سپس متد addMethod(name, func) را اضافه کنید که به ماشین حساب یک عملیات جدید را آموزش می دهد. نام عملگر و تابع دو آرگومان تابع (a,b) را می گیرد که آن را پیاده سازی می کند. به عنوان مثال:

let powerCalc = new Calculator;

powerCalc.addMethod("\*", (a, b) => a \* b);

powerCalc.addMethod("/", (a, b) => a / b);

powerCalc.addMethod("\*\*", (a, b) => a \*\* b);

let result = powerCalc.calculate("2 \*\* 3");

alert( result ); // 8

1-5- در نظر بگیرید آرایه ای از Object های مشخصات کاربر دارید که هر کدام دارای نام، نام خانوادگی و id هستند.

برنامه ای را بنویسید تا آرایه دیگری از Object ها متشکل از id و fullName ایجاد کند که fullName از نام و نام خانوادگی Object قبلی ایجاد می شود. به عنوان مثال:

let john = { name: "John", surname: "Smith", id: 1 };

let pete = { name: "Pete", surname: "Hunt", id: 2 };

let mary = { name: "Mary", surname: "Key", id: 3 };

let users = [ john, pete, mary ];

let usersMapped = /\* ... your code ... \*/

/\*

usersMapped = [

{ fullName: "John Smith", id: 1 },

{ fullName: "Pete Hunt", id: 2 },

{ fullName: "Mary Key", id: 3 }

]

\*/

alert( usersMapped[0].id ) // 1

alert( usersMapped[0].fullName ) // John Smit.

1-6- تابع getAverageAgeرا بگونه ای بنویسید که آرایه ای از اشیاء با ویژگی سن را دریافت می کند و میانگین سن را برمی گرداند. توجه داشته باشید تعداد اشیا، به هر اندازه ای می تواند باشد.

مثال:

let john = { name: "John", age: 25 };

let pete = { name: "Pete", age: 30 };

let mary = { name: "Mary", age: 29 };

let arr = [ john, pete, mary ];

alert( getAverageAge(arr) ); // (25 + 30 + 29) / 3 = 28

1-7- یک تابع getSecondsToday() بنویسید که تعداد ثانیه ها از ابتدای امروز تا زمان کنونی را برمی گرداند.

به عنوان مثال، اگر اکنون ساعت 10 صبح بود و هیچ تغییری وجود نداشت، پس:

getSecondsToday() == 36000 // (3600 \* 10)

1-8- یک تابع formatDate(date) بنویسید که باید تاریخ را به صورت زیر قالب بندی کند:

اگر از تاریخ و زمان کنونی کمتر از 1 ثانیه گذشته است، "در حال حاضر".

در غیر این صورت، اگر از تاریخ کمتر از 1 دقیقه گذشته باشد، " n ثانیه پیش".

در غیر این صورت، اگر کمتر از یک ساعت باشد، " n دقیقه قبل"

در غیر این صورت، تاریخ کامل در قالب "DD.MM.YY HH:mm" برگرداند.

به عنوان مثال:

alert( formatDate(new Date(new Date - 1)) ); // "right now"

alert( formatDate(new Date(new Date - 30 \* 1000)) ); // "30 sec. ago"

alert( formatDate(new Date(new Date - 5 \* 60 \* 1000)) ); // "5 min. ago"

// yesterday's date like 31.12.16 20:00

alert( formatDate(new Date(new Date - 86400 \* 1000)) );

1-9- یک تابع printNumbers(from, to) بنویسید که در هر ثانیه یک عدد را با شروع از و ختم به نمایش می دهد.

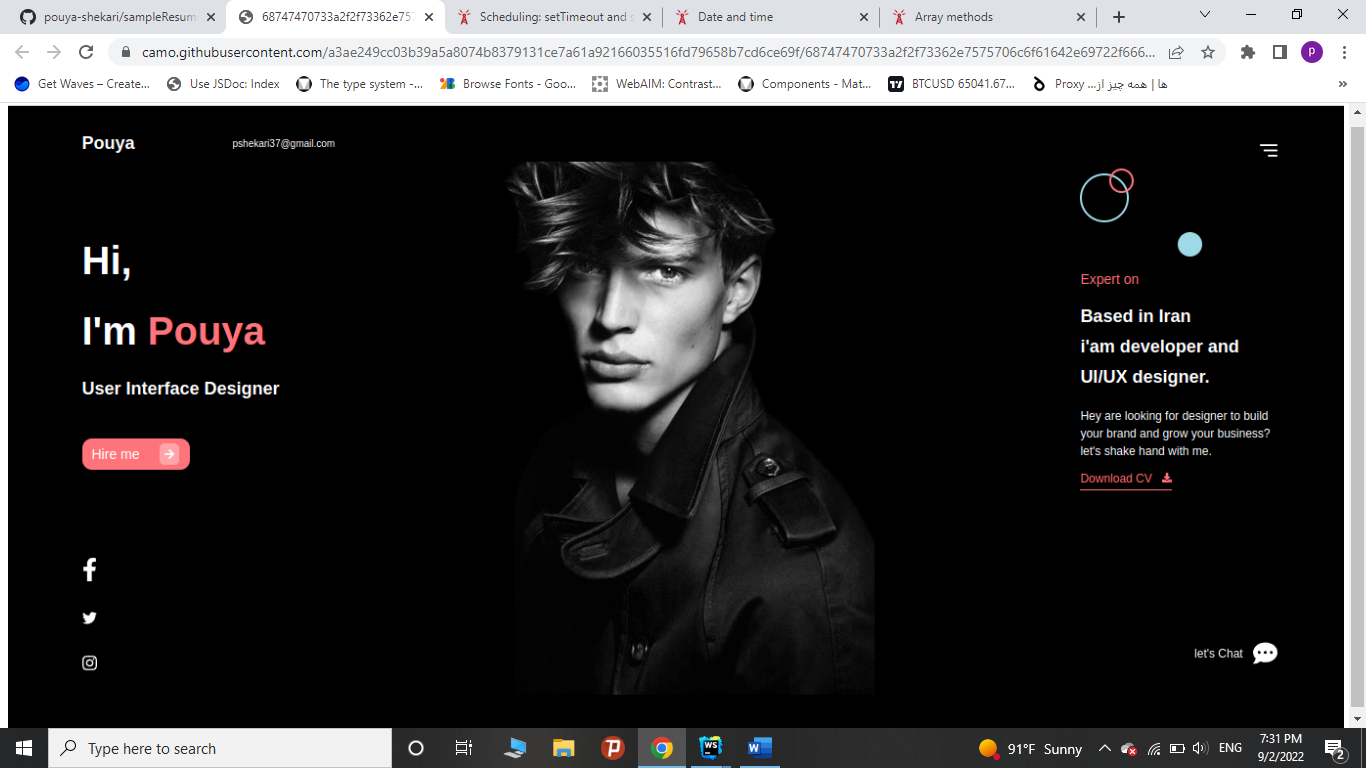
پیاده سازی را به دو صورت انجام دهید:

با استفاده از setInterval

استفاده از setTimeout تو در تو.

**۲- بخش HTML و CSS**

2-1- طرح زیر را در تمامی نماها ( دسکتاپ، موبایل و تبلت) با استفاده از اطلاعات خود پیاده سازی کنید.



**نکات**

* **مهلت ارسال تمرین تا پایان روز شنبه 12/10/98 است. پاسخ تمرین را در کارتابل آموزشی خود ارسال کنید.**
* **نام فایل و ایمیل ارسالی خود را به این صورت قرار دهید: Name\_hw1\_maktab51 به عنوان مثال Mohammad\_Ali\_Kargar\_hw1\_maktab48.**
* **تصاویر تمارین بخش HTML در کنار این فایل ضمیمه شده است.**
* **ارسال پاسخ تمارینی که در کلاس تحویل داده‌شده، الزامی است.**
* **در صورت لزوم یک فایل word به عنوان توضیح در کنار کدهای خود قرار دهید.**
* **در صورتی‌که تمرین شامل چند فایل و فولدر می­باشد حتماً آن­ها را قالب یک فایل فشرده‌شده تجمیع کنید.**
* **به دلیل بروز برخی مشکلات در ارسال فایل‌ها با پسوند rar، درصورتی که قصد ارسال فایل نهایی به صورت فایل rar دارید، پسوند آن را از rar به ra و درصورت عدم ارسال مجدد به pdf یا ... تغییر دهید. از ارسال شدن ایمیل خود اطمینان حاصل فرمایید.**
* **در صورتی‌که سوالی دارید در گروه تلگرامی بپرسید.**