گروه پنجم سوالات جاوااسکریپت

objects

توی این تمرین میخوایم کار با آبجکت‌ها توی جاوااسکریپت رو مرور و بعد تمرین کنیم.

اول لینک‌های زیر رو بخونید، این لینک‌ها، نحوه تعریف آبجکت ها، دسترسی به پراپرتی های اون، تغییر و حذف یک پراپرتی رو پوشش میده. همچنین روش هایی رو اشاره میکنه که باهاش بتونیم روی کلید ها (همون پراپرتی ها) و مقادیرشون حلقه بزنیم. همچنین در مورد کپی کردن آبجکت ها و call by reference بودنشون توضیح میده. در ادامه هم راجع optional chaining حرف میزنه. بحث مهم بعدی، this در جاوااسکریپت هست که لینک آخر رو کاویان ربانی از همکارای سابق من که الان رفته هلند نوشته و بسیار کاربردی و مهمه.

https://javascript.info/object
https://javascript.info/object-copy
https://javascript.info/optional-chaining
https://medium.com/@kavianrabbani/learn-this-keyword-once-and-forever-81c59ec805ba

بعد از خوندن این لینک‌ها، باید بدونیم آبجکت چی هست، چطوری مقدار کلید ها (پراپرتی ها) رو بخونیم، اونارو تغییر بدیم و یه کلید رو حذف کنیم. چطوری چک کنیم آیا کلیدی توی آبجکت مون وجود داره یا نه، چطوری روی کلید ها و مقادیرشون حلقه بزنیم، رفرنسی بودن آبجکت ها ینی چی و بخاطرش چطوری آبجکت هارو کپی بکنیم، optional chaining چیه و چه استفاده هایی داره و سینتکسش چطوریه و در نهایت this چیه و مقدارش توی شرایط مختلف چطوری تنظیم میشه.

بریم سراغ تمرین ها:

این تمرین 3 بخش داره.
توی بخش اول این تمرین، میخوایم از اون لیست 1000 تایی کاربرا که توی تمرین آرایه داشتیم، دوباره استفاده کنیم، پس مثل قبل، اون لیست رو بالا کدمون به اسم متغیر users قرار میدیم.

1 - تابع groupByCountry. این تابع، ورودی نداره. به عنوان خروجی، باید بهمون یه آبجکت بده که کلیدای این آبجکت، اسم کشورا هستن و مقدار هر کلید، لیست یوزرایی هست که توی اون کشوره. فرض کنید از قبل نمیدونیم چه کشورایی وجود دارن.

2 - تابع groupByGender. این تابع، ورودی نداره. به عنوان خروجی، باید بهمون یه آبجکت بده که کلیدای این آبجکت، جنسیت های موجود هستند (که از قبل نمیدونیم چیا هستن) و مقدار هر کلید، باید لیست کاربرای اون جنسیت باشه.

3 - حالا میخوایم یه تابع groupBy کلی بنویسیم که اسم یه کلید موجود توی آبجکت کاربر رو بهش میدیم و این بر اساس اون کلید، لیست کاربرا رو در قالب یه آبجکت، گروه بندی میکنه. مثلا:

```
groupBy('country')
groupBy('gender')
groupBy('birthDate')
```

مثلا برای وقتی که کلیدی که بهش میدیم، country هست، باید دقیقا جواب سوال 1 و وقتی بهش gender میدیم، باید دقیقا جواب سوال 2 رو بهمون برگردونه. مثلا در سناریوی country، یه همچین چیزی باید خروجی بده بهمون:

```
{
  Iran: [{}, {}, …],
  'United States': [{}, …],
  …
}
```

4 - حالا میخوایم تابعی قوی تر از سوال قبلم بنویسیم. اسم این تابع، groupByWithKeyGenerator هست. این تابع، جای اینکه بهش بگیم میخوایم بر اساس کدوم کلید گروه بندی کنیم، بهش یه تابع ورودی میدیم. این تابع رو باید صدا بزنیم تا کلیدهای آبجکت گروه بندی رو حساب کنیم. مثلا:

یه تابع داریم به اسم getAge که یه کاربر رو میگیره و سنشو برمیگردونه به این شکل:

```
const getAge = (user) => 2023 - (new Date(user.birthDate)).getFullYear()
```

حالا میخوایم بر اساس سن، کاربرا رو گروه بندی کنیم. تابع groupByWithKeyGenerator رو باید اینطوری صداش کنیم:

```
groupByWithKeyGenerator(getAge)
```

و باید خروجی ای شبیه زیر داشته باشیم:

```
{
  18: [{}, {}, …]],
  20: [{}, {}, {}, …],
  ….
}
```

توی بخش دوم، میخوایم توابع map و filter و reduce رو برای آبجکت ها بنویسیم.

5 - تابع map: این تابع، یه آبجکت و یک تابع به عنوان ورودی میگیره و یه آبجکت جدید بهمون برمیگردونه. این آبجکت جدید، کلید هاش همون آبجکت ورودی هستن، ولی مقدار هر کلید، خروجی تابع ورودی به map به ازای مقدار قبلی هست. تابعی ورودی که قراره برامون صدا زده باشه، باید کلید و مقدار هر عضو رو به اسم key و value دریافت کنه و انتظار داشته باشه که مقدار جدید value رو بهمون برگردونه. مثلا:

```
Let inputObj = {
  A:  10,
  B: 20
}
Function multiplyByTwo(key, value) { return key + a * 2 }
objectMap(inputObj, multiPlyByTwo)
```

خروجی:

```
{
  A: “A20”,
  B: “B40”
}
```

5 - تابع filter، این تابع، مشابه تابع map قبلی و همینطور فیلتر روی آرایه ها عمل میکنه. 2 تا ورودی میگیره و در خروجی بهمون یه آبجکت برمیگردونه که یه سری کلیدا داخلش نیست و فیلتر شدن. این فیلتر شدن رو با استفاده از تابع دوم میفهمه که کلید و مقدار هر پراپرتی رو بهش میده و انتظار داره خروجی اش یه boolean باشه که نشون بده توی آبجکت خروجی، اون کلید و مقدارش باید بیان یا نه.

6 - تابع reduce، این تابع هم مشخصا شبیه تابع reduce روی آرایه ها کار میکنه، ولی ورودی اش یه آبجکت و یه تابع reducer هست. به ازای هر پراپرتی توی آبجکتمون، عملیات reduce رو انجام میده. در نهایت آخرین خروجی عملیات reduce رو برمیگردونه.

تمرین بیشتر:

ساختار داده ای داریم به اسم درخت (tree) تو برنامه نویسی. با تو در تو کردن آبجکت ها تو جاوااسکریپت، به یه ساختار داده درخت میرسیم. الگوریتمای Depth First Search و Breadth First Search (به اختصار DFS و BFS) برای پیمایش و جستجو روی درخت ها با عمق نامعلوم استفاده میشن. تفاوت پیمایش و جستجوشون، صرفا در اینه که توی جستجو، همون پیمایش داره انجام میشه، فقط اگه یه مقداری برامون مناسب بود، به عنوان نتیجه جستجو بر میگردونیم و دیگه پیمایش رو ادامه نمیدیم.

https://javascript.plainenglish.io/tree-traversal-in-javascript-9b1e92e15abb
https://dev.to/anishkumar/tree-data-structure-in-javascript-1o99

در کل مبحث پیچیده ای هست. ممکنه نیاز باشه برای خودتون رو کاغذ نقاشی بکشید تا بفهمید چی میشه.

7 - تابع deepLog. این تابع، یه آبجکت با عمق **نامعلوم** میگیره و تمام مقادیر غیر آبجکتی (number و string و date و ...) که توی آبجکت وجود داره رو برامون لاگ میکنه.:

```
Let inputObj = {
  foo: 'bar',
  baz: {
    a: 'b',
    c: 'd',
    e: {},
    f: { g: 'h' }
  }
}
```

مثلا اگه این آبجکت رو بهش ورودی بدیم، برامون مینویسه 'bar' و 'b' و 'd' و 'h'.