گروه ششم سوالات جاوااسکریپت

DOM manipulation

توی این گروه، میخوایم با dom در مرورگر کار کنیم.

اول لینک‌های زیر رو بخونید، این لینک‌ها، اول به معرفی Document Object Model تو مرورگر میپردازن. اینکه ساختارش چیه و چه دیتاتایپ هایی توش وجود دارن و ارتباطش با تگ های html توی صفحه چیه. بعدش در مورد پیدا کردن تگ(ها) صحبت میکنه و بعدش هم پیمایش و ارتباط node ها توی dom باهم دیگه. در نهایت هم در مورد ساختن هر node و یا تگ html و تنظیم محتوا و attribute هاش صحبت میکنه.
لینک آخر هم در مورد event ها توی مرورگر هست. این لینک خودش 5 تا زیرموضوع داره که سعی کنید همشو بخونید.

https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
https://developer.mozilla.org/en-US/docs/Web/API/Document_object_model/Locating_DOM_elements_using_selectors
https://javascript.info/dom-nodes
https://javascript.info/dom-navigation
https://javascript.info/searching-elements-dom
https://javascript.info/basic-dom-node-properties
https://javascript.info/dom-attributes-and-properties
https://javascript.info/modifying-document
https://javascript.info/events

بعد از خوندن این لینک‌ها، با بدونیم که Document Object Model چی هست و چه دیتاتایپ هایی داخلش وجود داره، ارتباط دیتاتایپ ها با تگ های html چیه. با استفاده از جاوااسکریپت چطوری میتونیم DOM رو تغییر بدیم، المانی رو ازش حذف یا بهش المانی اضافه کنیم. چطوری المانی که وجود داره رو تغییر بدیم. چطوری محتوا و attribute برای تگ های html قرار بدیم و در نهایت چطوری event ست کنیم و چه تکنیک هایی داره.

بریم سراغ تمرین ها:

1 - میخوایم یه ابزاری بنویسیم که کار مارو هنگام کار با dom راحت بکنه. اسم این ابزار رو میذاریم elementor. میخوایم با استفاده از این ابزاری که مینویسیم، کد کمتر و خوانا تری بنویسیم و توسعه و کار با dom رو راحت تر کنیم.

میخوایم بتونیم با فراخوانی تابعی به اسم elementor و دادن یک selector با تایپ string، روی المانی که اون selector انتخابش میکنه، با یه تعدادی تابع دیگه، کارهایی انجام بدیم.

نکاتی که باید رعایت کنیم:
تابع elementor فقط روی یک المان عمل میکنه. پس اگه سلکتوری پاس داده شد که چند تا المان رو انتخاب میکرد، باید اولین المانی که با اون سلکتور انتخاب شده رو استفاده کنیم. همچنین اگه هیچ المانی با اون سلکتور انتخاب نشد، باید خطا برگردونیم.

تابع المنتور بهمون یه object برمیگردونه. این آبجکت، دارای متدهای attribute و click و on و text و class و removeClass هست. (بعد از کاراکتر دو نقطه ":" ، تایپی که انتظار داریم اون مقدار داشته باشه نوشته شده و برای شفاف شدن نوع ورودی هاست و کاربرد دیگه ای نداره .برای نمونه [string] به معنی آرایه ای از string هاست.)

```
function attribute(key: string, value: string) {}
function click(handler: function) {}
function on(eventName: string, handler: function) {}
function text(content: string) {}
function class(className: string | [string]) {}
function removeClass(className: string) {}
```

برای راحتی استفاده، باید بتونیم این توابع رو پشت سر هم صدا بزنیم. به این کار در برنامه نویسی، Builder Pattern گفته میشه. برای اینکه روی خروجی هر تابع بتونیم دوباره تابعی از همون آبجکت رو صدا کنیم، میتونیم this رو ریترن کنیم در انتهای هر تابع.

نمونه:

```
elementor('#some-id')
  .click(() => console.log('HI')
  .text('Click Me')
  .attribute('title', 'a click button')
```

```
  .on('mouseenter', () => console.log('Mouse Entered')
  .class(['text-bold', 'class2'])
```

راهنمایی:

اگر در پیاده‌سازی ساختار اولیه به مشکل خوردید، میتونید از قطعه کد زیر استفاده کنید و جاهایی که کامنت گذاشته شده رو پر کنید.

```
function elementor(selector) {
    // TODO
    return {
        setAttribute(key, value) {
            // TODO
            return this
        },
        on(eventName, handler) {
            // TODO
            return this
        },
        click(handler) {
            // TODO
            return this
        },
        text(content) {
            // TODO
            return this
        },
        class(className) {
            // TODO
            return this
```

```
        },
        removeClass(className) {
            // TODO
            return this
        }
    }
}
```

2 - فرض کنید میخوایم تابعی بنویسیم که یک آبجکت تو در تو میگیره و ساختار html صفحه متناظر با اون رو برامون توی صفحه میسازه. اسم این تابع رو render میذاریم.

هر تگ رو با یه آبجکت مدل می کنیم. پس هر تگ یک آبجکت هست که به فرم زیره:

```
{
  Name: ‘div’,
  Attributes: [{key: ‘’, value: ‘’}],
  Children: [{}]
}
```

کلید children، فرزندای اون تگ رو نشون میده. فرزندا میتونن خودشن یه آبجکت باشن یا اینکه یه string ساده باشن. عمق این آبجکت ها هم نامعلومه و باید تابع رو به صورت بازگشتی پیاده‌سازی کنید.
کلید attributes، لیستی از attribute های اون تگ رو نشون میده. هر attribute، یک آبجکت هست که یک کلید key داره و یک کلید value, که به ترتیب اسم و مقدار attribute هستن.

فرض کنید یه همچین آبجکتی داشته باشیم و به تابع رندر ورودی بدیمش:

```
{
  name: ‘div’,
  attributes: [{key: ‘class’, value: ‘container’}, {key: ‘id’, value:
‘some-id’ }],
  children: [
    “Salam”,
    {
      name: ‘p’,
      attributes: [{ key: ‘class’, value: ‘text-bold’ }],
      children: [“Some Text”]
    }
  ]
}
```

تابع رندر، باید صفحه html رو به این صورت بسازه:

```
<html>
    <body>
        <div class="container" id="some-id">
            Salam
            <p class="text-bold">
                Some Text
            </p>
        </div>
    </body>
</html>
```

3 - فرض کنید یه سایتی داریم که داخلش انواع فرم‌های مختلف رو داریم و این فرما، داینامیک هستن و برای موضوعات مختلف، فرم های مختلف داریم. مثلا توی ثبت آگهی در دیوار، برای هر دسته بندی، یه فرم متفاوت داریم. یا برای ثبت یه محصول تو دیجی کالا، فرم با فیلدهای متفاوت داریم که بر اساس نوع محصول مشخص میشن. نمیتونیم به ازای تمام این فرم‌ها کلی html بنویسیم. میخوایم یه تابع فرم ساز بنویسیم. که یه نمایشی از تگ های فرم رو بهش بدیم و html اون رو توی صفحه برامون بسازه.

برای اینکه خیلی سختش نکنیم، فقط فیلدهای متنی ساده و checkbox و select رو میتونیم بسازیم.

هر فیلد رو با یه آبجکت مدل میکنیم. این آبجکت، اول یه تایپ داره که میگه text هست یا checkbox هست یا select هست.

هر نوع از اینپوت، یه سری کلید دیگه هم میتونه داشته باشه. در ادامه میگیم که هر اینپوت، آبجکتش دقیقا چه کلیدایی داره. مقادیری که به این کلیدا میدیم، بر اساس نوع نیاز محصول متفاوت هستن و اینجا به عنوان مثال پر شدن.

```
Text => {
  id: “name“
  Type: “text”,
  label: “نام:”,
  placeholder: “نام خود را وارد کنید…”,
  inputType: “text”|”number”|”password”|”email”
}

Checkbox => {
  id: “isNew”,
  type: “checkbox”,
  label: “کالا نو است.”,
}

Select => {
  id: “state”,
  type: “select”,
  label: “وضعیت کالا”,
  options: [
    { value: “new”, “label”: “نو” },
    { value: “second-hand”, label: “دست دوم”}
```

```
  ]
}
```

تابعی که میخوایم بنویسیم، قراره لیستی از این اینپوت هارو بگیره و ساختار html اش رو توی صفحه بسازه.
مثلا اگه یه همچین لیستی از اون فیلدها داشته باشیم و به تابع‌مون ورودی بدیم:

```
[{
    id: "firstName",
    Type: "text",
    label: "نام:",
    placeholder: "نام خود را وارد کنید...",
    inputType: "text"
},
{
    id: "lastName",
    Type: "text",
    label: "نام خانوادگی:",
    placeholder: "نام خانوادگی خود را وارد کنید...",
    inputType: "text"
},
{
    id: "email",
    Type: "text",
    label: "ایمیل:",
    placeholder: "ایمیل خود را وارد کنید...",
    inputType: "email"
},
{
    id: "experiences",
    Type: "select",
    label: "میزان سابقه:",
    options: [
        { value: '1', label: '1 سال'},
        { value: '2', label: '2 سال'},
        { value: '3', label: '3 سال'},
        { value: '4', label: '4 سال'},
        { value: 'more-than-4', label: 'بیش از 4 سال'}
    ]
```

}]

خروجی داخل صفحه میشه این:

```
<body>
    <form action="">
        <label for="firstName">نام:</label>
        <input placeholder="نام خود را وارد کنید..." id="firstName"
type="text" />

        <label for="lastName">نام خانوادگی:</label>
        <input placeholder="نام خانوادگی خود را وارد کنید..." id="lastName"
type="text" />

        <label for="email">ایمیل:</label>
        <input placeholder="ایمیل خود را وارد کنید..." id="email"
type="email" />

        <label for="experiences">میزان سابقه : </label>
        <select id="experiences">
            <option value="1">1 سال</option>
            <option value="2">2 سال</option>
            <option value="3">3 سال</option>
            <option value="4">4 سال</option>
            <option value="more-than-4">بیش از 4 سال</option>
        </select>
    </form>
</body>
```