ابوالفضل دانیالی 401110256

در این تمرین هدف ما ایجاد یک صفحه‌ی وب است که بتواند برای یک فرمول (و به‌طور کلی هر فرمولی) مجموعه‌ای از ورودی‌ها را از خود صفحه دریافت کند و به‌صورت ریسپانسیو، با تغییر ورودی‌ها، خروجی حاصل از فرمول را در لحظه به‌روزرسانی نماید.

این تمرین از سه بخش تشکیل شده است:

- **h.html:**نمایش المان‌ها در صفحه
- **Eval.js:**پیاده‌سازی منطق
- **Style.css:**تزیین المان‌ها و صفحه

h.html:

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Dynamic Formula Evaluator</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h2>🧾 Basic Price Calculation</h2>
  <p>Formula: <code>count * fee - discount</code></p>
  <input type="text" id="fee" placeholder="قیمت واحد" />
  <input type="text" id="count" placeholder="تعداد" />
  <input type="text" id="discount" placeholder="تخفیف" />
  <div class="formula" evaluator="count * fee - discount"></div>


  <h2>🔵 Area and Perimeter of a Circle</h2>
  <p>Formula: <code>Math.PI * radius * radius</code></p>
  <input type="text" id="radius" placeholder="Radius" />
  <div class="formula" evaluator="Math.PI * radius * radius"></div>

  <p>Formula: <code>2 * Math.PI * radius</code></p>
  <div class="formula" evaluator="2 * Math.PI * radius"></div>

  <hr>

  <h2>⚠️ Invalid Formula Example</h2>
  <p>Formula: <code>radius **</code></p>
  <div class="formula" evaluator="radius **"></div>

  <hr>

  <h2>💰 Salary Calculations</h2>
  <p>Formula: <code>salary ** 2</code></p>
  <input type="text" id="salary" placeholder="Salary" />
  <div class="formula" evaluator="salary ** 2"></div>

  <p>Formula: <code>salary + 2 ** (monthEmployed) * bonus</code></p>
  <input type="text" id="monthEmployed" placeholder="Months Employed" />
  <input type="text" id="bonus" placeholder="Bonus" />
  <div class="formula" evaluator="salary + 2 ** (monthEmployed) * bonus"></div>

  <p>Formula: <code>salary + 2 ** (monthEmployed) * bonus + gift</code>
  (no gift input → will show error)</p>
  <div class="formula" evaluator="salary + 2 ** (monthEmployed) * bonus + gift"
></div>

  <script src="eval.js"></script>
</body>
</html>
```

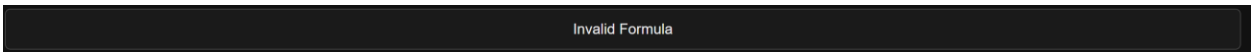
برای هر فرمول، مجموعه‌ای از ورودی‌ها در نظر گرفته شده است. هر ورودی با یک **textbox** که دارای کلاس input و یک **id** منحصر به‌فرد است، نمایش داده می‌شود. این id نشان‌دهنده‌ی نام مورد استفاده در فرمول است (در صورتی که باکس خالی باشد، این نام با استفاده از placeholder نمایش داده می‌شود). در کد جاوااسکریپت نیز از همین نام استفاده می‌کنیم. نتیجه‌ی هر فرمول در یک div قرار می‌گیرد که به آن کلاس formula اختصاص داده شده، و خود فرمول به‌صورت attribute با نام evaluator ذخیره می‌شود.

به‌عنوان مثال، چندین فرمول از جمله مثال صورت سوال تمرین، محیط و مساحت دایره، و حقوق یک کارمند نمایش داده شده‌اند (فرمت نوشتاری فرمول‌ها در خود صفحه مشخص شده است).

تنها نکته‌ی باقی‌مانده، بحث **خطاها (errors)** است. در مثال‌ها، تست‌کیس‌های نادرست نیز برای شرایطی که فرمول معتبر است اما خطا دارد، در نظر گرفته شده‌اند. در اینجا دو نوع خطا بررسی شده است:

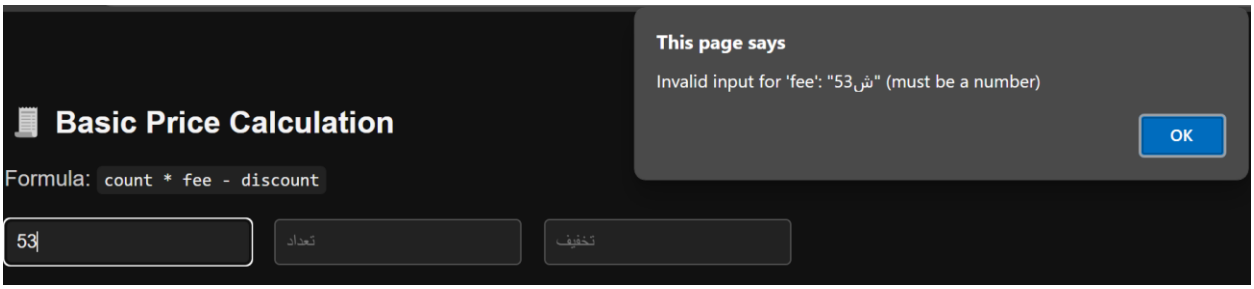
1. **فرمول نامعتبر است و معنا ندارد** ← مثل** radius :
2. **یکی از ورودی‌ها وجود ندارد** ← مانند نبودن ورودی gift در تست‌کیس آخر

در هر دو حالت، پیام **Invalid Formula** نمایش داده می‌شود.

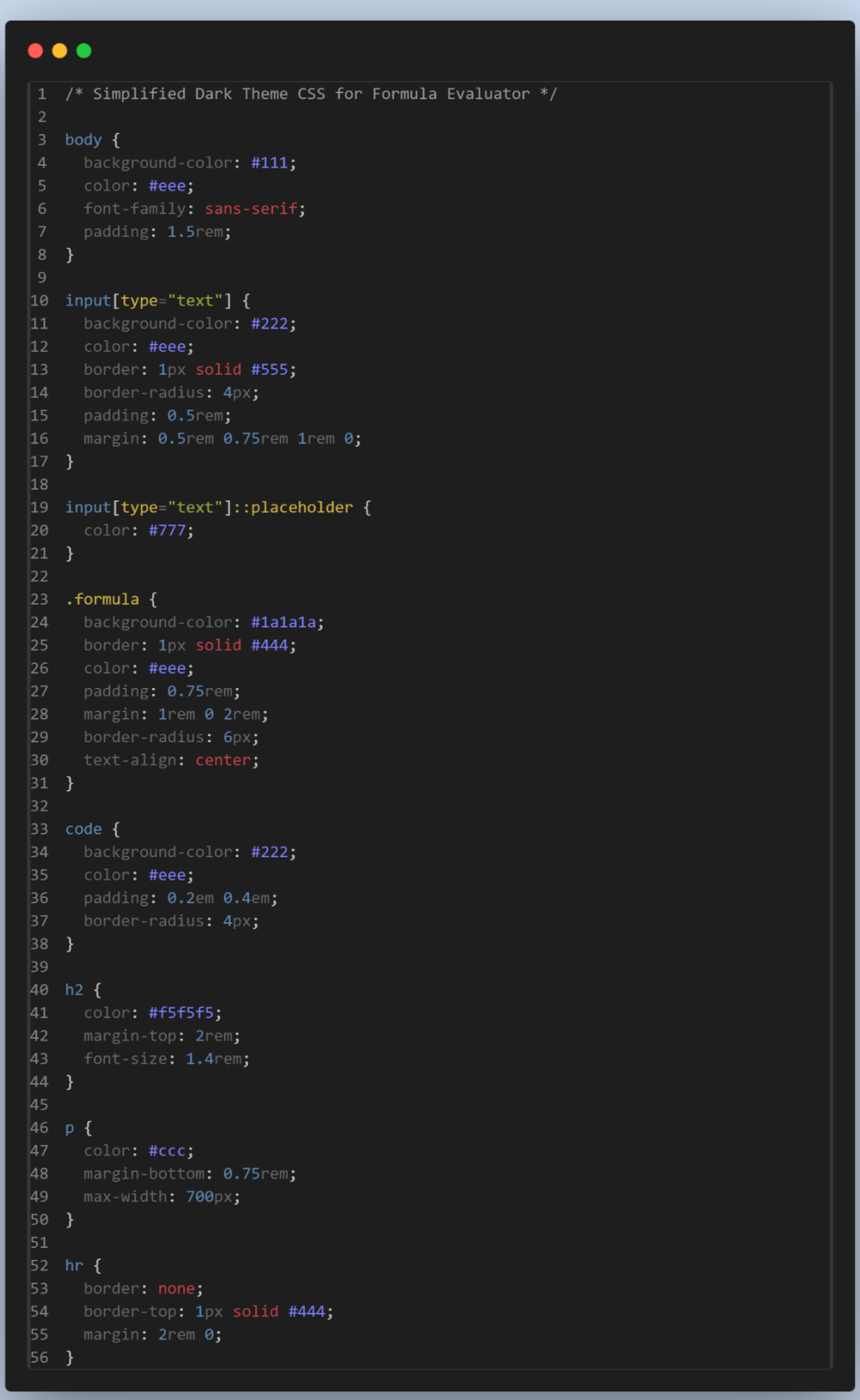


خطای دیگر این است که اگر برای یک فرمول، یکی از ورودی‌ها خالی باشد، پیام **ورودی‌ها ناقص هستند** نمایش داده می‌شود.

در صورتی که مقدار غیر عددی در یکی از ورودی‌ها وارد شود، به کاربر هشدار داده می‌شود که مقدار واردشده نامعتبر است.



Style.css



```css
/* Simplified Dark Theme CSS for Formula Evaluator */

body {
  background-color: #111;
  color: #eee;
  font-family: sans-serif;
  padding: 1.5rem;
}

input[type="text"] {
  background-color: #222;
  color: #eee;
  border: 1px solid #555;
  border-radius: 4px;
  padding: 0.5rem;
  margin: 0.5rem 0.75rem 1rem 0;
}

input[type="text"]::placeholder {
  color: #777;
}

.formula {
  background-color: #1a1a1a;
  border: 1px solid #444;
  color: #eee;
  padding: 0.75rem;
  margin: 1rem 0 2rem;
  border-radius: 6px;
  text-align: center;
}

code {
  background-color: #222;
  color: #eee;
  padding: 0.2em 0.4em;
  border-radius: 4px;
}

h2 {
  color: #f5f5f5;
  margin-top: 2rem;
  font-size: 1.4rem;
}

p {
  color: #ccc;
  margin-bottom: 0.75rem;
  max-width: 700px;
}

hr {
  border: none;
  border-top: 1px solid #444;
  margin: 2rem 0;
}
```

از یک تمپلت ساده استفاده شده و برخی مقادیر برای paddingو ... از chatgpt کمک گرفته شده.

Eval.js

```
function evaluateAllFormulas() {
    document.querySelectorAll(".formula").forEach(formulaBox => {
      const expr = formulaBox.getAttribute("evaluator");

      try {
        const allWords = expr.match(/\b[a-zA-Z_]\w*\b/g) || [];
        const matchDotProps = expr.matchAll(/\b\w+\.(\w+)\b/g);
        const dotProps = Array.from(matchDotProps, m => m[1]);

        const variables = [];

        for (const name of allWords) {
          if (name === "Math" || dotProps.includes(name)) continue;
          variables.push(name);
        }


        let scope = {};
        let hasEmpty = false;

        for (const variable of variables) {
          const input = document.getElementById(variable);
          const value = input.value.trim();

          if (value === "") {
            hasEmpty = true;
            continue;
          }

          if (!/^[-+]?\d*(\.\d+)?$/.test(value)) {
            alert(`Invalid input for '${variable}': "${value}" (must be a number)`
);
            formulaBox.textContent = "Invalid Input";
            return;
          }

          scope[variable] = parseFloat(value);
        }

        if (hasEmpty) {
          formulaBox.textContent = "Please fill all inputs";
          return;
        }

        const func = new Function(...variables, `return ${expr};`);
        const result = func(...variables.map(v => scope[v]), Math);

        formulaBox.textContent = result;

      } catch (err) {
        formulaBox.textContent = "Invalid Formula";
      }
    });
  }

  document.querySelectorAll("input").forEach(input => {
    input.addEventListener("input", evaluateAllFormulas);
  });

  evaluateAllFormulas();
```

در ابتدا تابع evaluateAllFormuals قرار دارد که وظیفه آن محاسبه فرمول ها بر حسب ورودی ها می باشد و به این صورت عمل می کند :

با get attribute همه فرمول ها را با استفاده از کلاس formula می یابیم به ازای هر عضوی که کلاس formula را داراست خود شکل فرکول را که در attribute evaluate را درexpr ذخیره می کنیم

سپس کلماتی که فقط دارایحروف هستند را با استفاده از عبارت منظم \b[a-zA-Z_]\w*\b جدا می کنیم که نتیجه همان اسم و idورودی هاست

در فرمول ممکن است فقط از ورودی ها اتستفاده نشده باشد به طور مثال math.pi را داریم که نباید با متغیر اشتباه گرفته شده باشد و در dotprops این جداسازی رخ داده و pi و امثال آن را در dotprops ذخیره می کنیم

سپس کلمه هایی که اسم متعیر هستند(در بند بالا صدق نمی کنند) را در varaiblesذخیره می کنیم .

در مرحله بعدی error handling را انجام می دهیم و مقتدیر متغیر ها از باکس مربوطه استخراج می کنیم و در scope ذخیره می کنیم.

رو اسم متغیر ها حرکت می کنیم و متناظر با id برابر با ان ها داده نوشته رو بکس متناظر را دریافت کرده، وایت اسپیس ها را از آن حذف نموده و در value ذخیره می کنیم و سپس آن را بررسی می کنیم

اگر مقدرا آن تهی بوده و hasemptyرا true قرار می دهیم که یعنی یک ورودی خالی است و باید ارور خالی بودن ورودی را نمایش دهیم.

سپس با استفاده از ریجکس /^[-+]?\d*(\.\d+)?$/ چک کینیم آیا رشته value فرمت یک عدد را دراست یا خیر(اگر صدق کرد یک عدد است) و اگر یک عدد نبود این موضوع را alert می دهیم و از تابع خارج می شویم

اگر هیچ یک از error ها را رخ نداد value را به float کست می کنیم و در scope ذخیره می کنیم.

در نهایت تابع func می سازیم که ورودی آن متغیر هاست و خروجی فرمول استخراج شده در expr می باشد. و این تابع مقادیر متغیر ها که در scope ذخیره شده را به به تابع پاس می دهیم و نتیجه را در باکس فرمول اولیه می ریزیم و در این میانه اگر خطایی یافت شد آن را catch می کنیم و به عنوان فرمول دارای مشکل پاس می دهیم

در نهایت صفحه را responsive می کنیم بدین شکل که به هر input box یک eventlistner اضافه می کنیم که هر گاه کاربر یه خانه را تغییر داد تابع evaluateAllFormuals صدا زده می شود و به طور دستی نیز بعد از load شده پیج یه دور تابع را صدا می زنیم تا مقادیر اولیه فرمول ها معلوم شود.