برنامه‌سازی وب

کلید آزمون میانترم – نیم‌سال پاییز ۱۴۰۴

نام استاد: علی ابریشمی | دانشگاه صنعتی شریف | زمان آزمون: ۱۲۰ دقیقه

۱. به کدهای HTML و CSS صفحه‌ی بعد دقت کرده، سپس به سوالات زیر جواب دهید:

(الف) تعدادی مورد در کدهای داده شده وجود دارند که کد را ملزم به ریفکتور شدن می‌کنند. ۲ مورد از آن‌ها را نام ببرید. حداقل یک مورد از مواردی که می‌نویسید باید روی بهینه‌سازی موتور جستجو تاثیر داشته باشد و آن را به طور خاص مشخص کنید. (۶ نمره)

هر ۲ مورد معقولی که باعث بهبود کلی کیفیت کد یا خروجی آن شود، مورد قبول واقع می‌شود. برای مثال می‌توان پیشنهادهای زیر را در نظر گرفت:

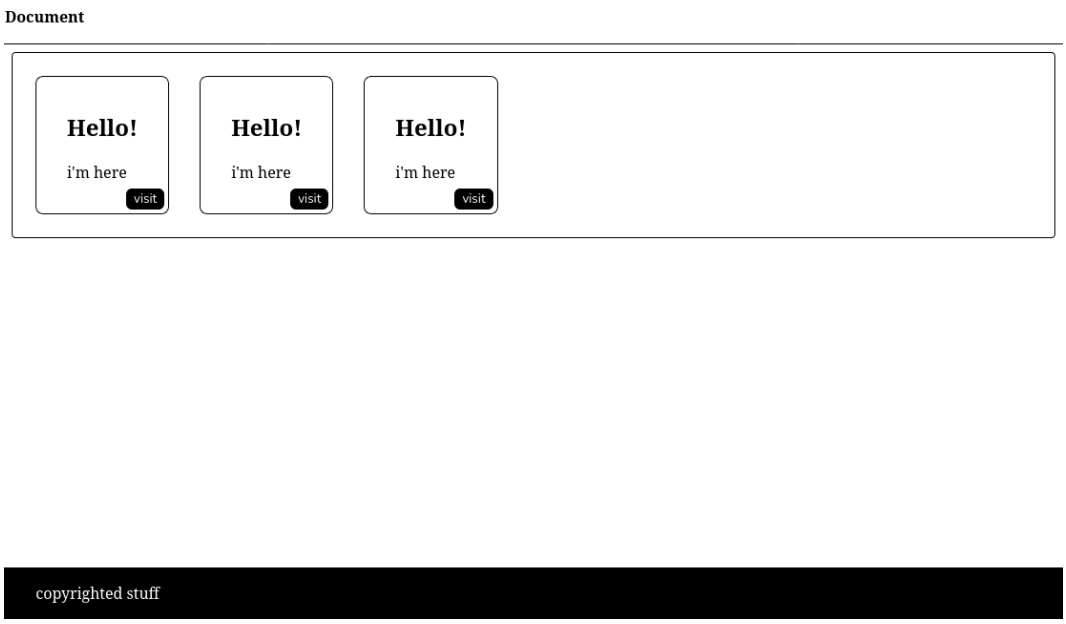
- استفاده از semantic tags به جای `div`های فعلی
- حذف `rule`های استفاده نشده CSS
- افزودن تگ‌های meta
- واکنش‌گرا یا Responsive کردن طراحی
- قرار دادن رنگ‌ها و اندازه‌ها در متغیرهای `:root` در CSS

منابع:

- اسلاید ۹ HTML
- اسلاید ۱۰ HTML
- اسلاید ۳۵ HTML
- اسلایدهای ۳۱ تا ۳۳ CSS
- اسلاید ۳۵ CSS

(ب) چینش کلی صفحه یا Layout آن را با توجه به CSS اعمال شده رسم کنید. (۱۴ نمره)

شکل دقیق آن به صورت زیر است، با این وجود هر شکلی که با تقریب خوبی درک شما را از قوانین و شیوه‌ی اعمال آن‌ها در نشان دهد مورد پذیرش واقع شده است. (در واقع نشان دادن کلی layout کافیست و نیازی به نشان دادن جزئیات نیست)



منابع: کل اسلایدهای HTML و CSS و تمرین اول درس

(ج) یک نمونه استفاده برای هر مورد از حالات `display` زیر بنویسید. مثال‌ها باید مستقل از کد آورده شده باشند. (۱۵ نمره)

به طور کلی هر کاربرد مشخص و قابل تمیز از بقیه کاربردها مورد قبول است. با این حال یک یا دو مثال برای هرکدام آورده شده.

`display`	مورد استفاده
`block`	ساختاربندی header و footer که کل خط را در بر میگیرند.
`flex`	ردیف کارت‌های تخفیف فروشگاه آنلاین، Navbar یک وب‌سایت و ...
`inline`	قرمز کردن بخشی از متن یک پاراگراف، قرار دادن تعدادی آیکن داخل متن بدون شکستن آن و ...
`inline-block`	قرار دادن کارت تبلیغاتی با طول و عرض مشخص در میان متن یک پاراگراف
`grid`	طراحی Quick Access تمرین اول، نمایش دسته‌بندی‌های کالاهای اسنپ مارکت و ...

منابع:

- تمرین اول درس
- اسلایدهای ۲۳ تا ۲۹ CSS

HTML

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Document</title>
    <link href="styles.css" rel="stylesheet" />
  </head>
  <body>
    <div class="header">
      <p><b>Document</b></p>
    </div>
    <div class="main">
      <div class="section">
        <h2>Hello!</h2>
        <p>i'm here</p>
        <button>visit</button>
      </div>
      <div class="section">
        <h2>Hello!</h2>
        <p>i'm here</p>
        <button>visit</button>
      </div>
      <div class="section">
        <h2>Hello!</h2>
        <p>i'm here</p>
        <button>visit</button>
      </div>
    </div>
    <div class="footer">
      <p>copyrighted stuff</p>
    </div>
  </body>
</html>
```

CSS

```css
body {
    margin: 0;
}

.header {
  padding: 1px;
  border-bottom: #000 solid 1px;
}

.main {
  padding: 8px;
  margin: 8px;
  border-radius: 4px;
  border: #000 solid 1px;
  display: flex;
  flex-direction: row;
}

#main {
  padding: 8px;
  margin: 8px;
  border-radius: 4px;
  border: #000 solid 1px;
  display: flex;
  flex-direction: column;
  gap: 8px;
}

.section {
  display: block;
  position: relative;
  padding: 1rem 2rem;
  margin: 1em;
  border: #000 solid 1px;
  border-radius: 8px;
}

.section button {
  position: absolute;
  bottom: 0;
  right: 0;
  margin: 4px;
  border-radius: 6px;
  padding: 2px 8px;
  border: 0;
  background-color: #000;
  color: #fff;
}

.section #button {
  margin: 12px;
  border-radius: 6px;
  padding: 2px 8px;
  justify-items: center;
  border: 0;
  background-color: #000;
  color: #fff;
}

.footer {
  position: absolute;
  bottom: 0;
  background-color: #000;
  color: #fff;
  width: 97%;
  padding-left: 3%;
}
```

۲. کد جاوااسکریپت زیر را در نظر بگیرید:

```
const a = async () => {await b(); await c(); console.log('o')}
async function b() { setTimeout(() => {console.log('f')}, (5*'2'*1+'0'+0)) }
async function c() {await b(); setTimeout(() => {d(); e(); e();}, 2000); console.log('c')}
const d = async () => {setTimeout(() => {console.log('!')}, 3000)}
const e = () => console.log('e')
```

(الف) اجرای تابع b چند ثانیه طول می‌کشد؟ چرا؟ (۵ نمره)

به ترتیب پیش می‌رویم:

```
5*'2'=5*2=10
10*1=10
10+'0'='10'+'0'='100'
'100'+0='100'+'0'='1000'
```

زمان چاپ شدن کاراکتر f در واقع ۱۰۰۰ میلی‌ثانیه یا ۱ ثانیه است.

منبع: اسلاید ۲۲ JavaScript

(ب) روی یک محور زمانی، زمان چاپ شدن هر کاراکتر پس از اجرای تابع a را نشان دهید. محور شما باید بر حسب ثانیه باشد و از ۰ شروع شود. همچنین قید کنید که چاپ کل خروجی، چند ثانیه طول خواهد کشید. (۵ نمره)

نکته سوال این بود که await تاثیری در زمان اجرای setTimeout ندارد. کال شدن این تابع صرفا باعث میشود عملیاتی برای n میلی‌ثانیه بعد زمان‌بندی شود.

از a شروع می‌کنیم. تابع b صرفا setTimeout را با بدنه‌اش کال می‌کند و به پایان می‌رسد. این تابع تمام شده و میتوانیم به سراغ تابع بعد برویم. تابع c یک بار دیگر b را فراخوانی کرده و بابت اجرای بدنه‌ی آن صبر می‌کند. سپس یک setTimeout دیگر را زمان‌بندی کرده و نهایتا حرف 'c' را چاپ می‌کند. پس در زمان ۰ یک حرف c چاپ شده. حال اجرای بدنه‌ی c تمام شده و می‌توانیم به سراغ قسمت پایانی بدنه‌ی a برویم. در همان زمان ۰ حرف 'o' چاپ می‌شود. (توجه کنید زمان اجرای هر قطعه را داریم ۰ در نظر میگیریم. صرفا زمانبندی‌ها اهمیت دارند.) حال دو کاراکتر 'f' که برای یک ثانیه بعد از زمان اجرای a و c زمانبندی شده بودند (یعنی در ثانیه ۱) چاپ می‌شوند. سپس تابع تعریف شده داخل setTimeout تابع c بعد از ۲ ثانیه (یعنی در ثانیه ۲) اجرا می‌شود. این تابع در همان لحظه اجرا ابتدا یک '!' برای ۳ ثانیه بعد (یعنی در ثانیه ۵) زمانبندی برای چاپ می‌شود. سپس در همان ثانیه ۲، ۲ عدد 'e' چاپ می‌کند.

به طور خلاصه زمانبندی شما باید به صورت زیر نوشته شده باشد:

- ثانیه صفرم: c و o
- ثانیه اول: f و f
- ثانیه دوم: e و e
- ثانیه پنجم: !

که خروجی !coffee بوده و چاپ تمام حروفش ۵ ثانیه طول می‌کشد.

برای آزمایش نیز کد را به صورت زیر تغییر می‌دهیم:

```
const init = Date.now()
const a = async () => {
  await b(); await c(); console.log('o')
  console.log(` :'o' PRINTED${Date.now() - init}`)
}
async function b() {
  setTimeout(() => {
    console.log('f')
    console.log(` :'f' PRINTED${Date.now() - init}`)
```

```
  }, (5*'2'*1+'0'+0))
}
async function c() {
  await b(); setTimeout(() => {d(); e(); e();}, 2000);
  console.log('c')
  console.log(` :'c' PRINTED${Date.now() - init}`)
}
const d = async () => {
  setTimeout(() => {
    console.log('!')
    console.log(` :'!' PRINTED${Date.now() - init}`)
  }, 3000)
}
const e = () => {
  console.log('e')
  console.log(` :'e' PRINTED${Date.now() - init}`)
}

a()
```

که خروجی‌اش به صورت زیر است:

```
c
PRINTED 'c': 10
o
PRINTED 'o': 10
f
PRINTED 'f': 1001
f
PRINTED 'f': 1001
e
PRINTED 'e': 2003
e
PRINTED 'e': 2004
!
PRINTED '!': 5006
```

منابع:

- فایل‌های promises.js و promises-ex1.js و promises-ex2.js و promises-ex3.js و varproblems.js
- اسلایدهای ۵۳ تا ۶۰ JavaScript

(ج) توضیح دهید چگونه می‌توانستیم به جای console.log هر کاراکتر را به یک آرایه اضافه کنیم. حتماً متدهای Array لازمه را در پاسخ خود قید کنید (۵ نمره)

صرفا قید کردن ایجاد یک آرایه و استفاده از متد push روی آن کفایت می‌کرد.

منبع: اسلایدهای ۴۵ و ۴۶ JavaScript

۳. کد HTML زیر را در نظر بگیرید:

```
<html>
<head></head>
<body>
    <div class="abc">
        <p class="pr">Hey</p>
    </div>
    <button id="vbt">bop</button>
    <script>
        document
            .getElementById('vbt')
            .addEventListener('click',
            () => {
                let lmnt = document.querySelectorAll(".abc")
                for (let i = 0; i < lmnt.length; i++) {
                    p = document.createElement("p")
                    p.className = "pr"
                    p.innerText = "Hey"
                    curr = lmnt[i].cloneNode()
                    curr.appendChild(p)
                    lmnt[i].appendChild(curr)
                }
            });
    </script>
</body>
</html>
```

(الف) توضیح دهید چگونه می‌توان حلقه for موجود را به یک حلقه forEach تبدیل کرد؟ مزیت این کار چیست؟ نوشتن کد دقیق نیاز نیست، صرفا ساختار جدید را مشخص یا توصیف کنید. (۶ نمره)

کد کامل و دقیق آن به صورت زیر است، با این حال توصیف کلی از ساختار forEach کافی بود.

```
document
    .getElementById('vbt')
    .addEventListener('click', () => {
        let lmnt = document.querySelectorAll(".abc");

        lmnt.forEach(item => {
            const p = document.createElement("p");
            p.className = "pr";
            p.innerText = "Hey";

            const curr = item.cloneNode();
            curr.appendChild(p);
```

```
            item.appendChild(curr);
        });
    });
```

منابع: اسلاید ۴۶ و ۴۹ JavaScript و کد کلاسی arraymethods.js از کدهای کلاسی JavaScript

(ب) درخت DOM را در ابتدا، یک و دو بار پس از فشردن دکمه d رسم کنید. (۹ نمره)

نکته سوال این بود که حین اضافه شدن فرزندان جدید، لیست مربوطه بدون تغییر می‌ماند.

به منظور آزمایش گزاره بالا، کد را به صورت زیر تغییر می‌دهیم تا بعد از هر تغییر وضعیت لیست lmnt را ببینیم:

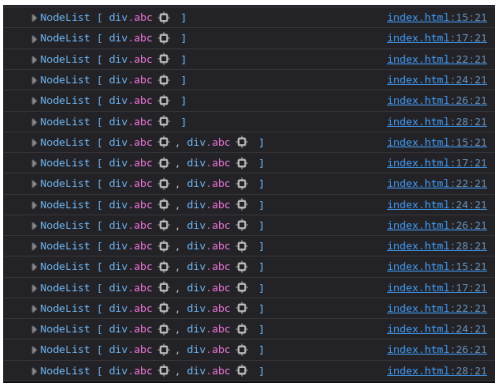
```
document
    .getElementById('vbt')
    .addEventListener('click', () => {
        let lmnt = document.querySelectorAll(".abc");

        lmnt.forEach(item => {
            console.log(lmnt)
            const p = document.createElement("p");
            console.log(lmnt)
            p.className = "pr";
            p.innerText = "Hey";

            const curr = item.cloneNode();
            console.log(lmnt)
            curr.appendChild(p);
            console.log(lmnt)

            console.log(lmnt)
            item.appendChild(curr);
            console.log(lmnt)
        });
    });
```

که خروجی آن پس از یک کلید و دو کلیک به صورت زیر است:

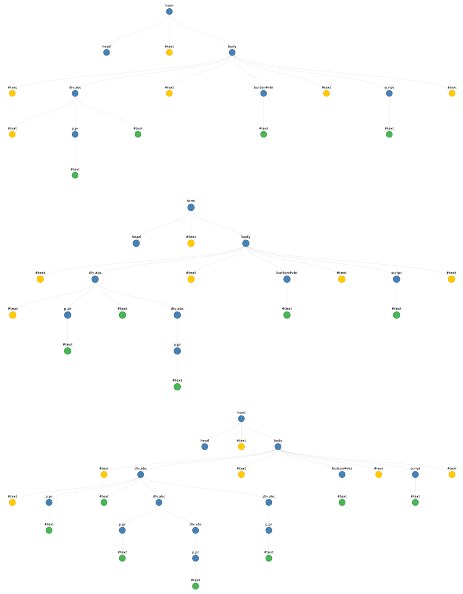


که صحت گزاره گفته شده را تایید می‌کند.

در واقع در مرحله ابتدایی که بدیهتا یک `div` شامل یک پاراگراف داریم. در مرحله دوم، یعنی پس از اولین کلیک، به `div` مربوطه یک `div` جدید شامل یک پاراگراف اضافه می‌شود. حال در مرحله سوم، یعنی پس از دومین کلیک، به هر دو `div` موجود، به هرکدام یک `div` شامل یک پاراگراف اضافه می‌شود.

یعنی پس از nامین کلیک، ما باید 2^n عدد `div` و `p` داشته باشیم.

در نتیجه درخت‌های DOM از ابتدا تا بعد از کلیک دوم به ترتیب به صورت زیر می‌شوند:



که البته کافی بود فقط خود المان ها را از `body` به بعد نشان دهید. (رئوس آبی جز HTML و Head)

با اینکه خواسته سوال نبوده، ولی تعداد المان‌ها را هم مورد بررسی قرار می‌دهیم. از هر مرحله از ابتدا تا پس از دومین کلیک یک تصویر به ترتیب آمده:

Hey

Hey

Hey

Hey

bop

که یعنی گزاره گفته شده در مورد تعدادها نیز درست است.

منابع:

- تمرین اول
- اسلایدهای ۹ و ۱۰ و ۱۶ و ۱۸ و ۲۲ تا ۳۰ مبحث DOM

۴. کاربرد متدهای آرایه `map` و `reduce` را در جاوااسکریپت با ذکر یک مثال توضیح دهید. (۵ نمره)

برای `map`: روی یک آرایه می‌چرخد، روی هر عنصر یک تابع اعمال می‌کند و یک آرایهٔ جدید برمی‌گرداند. مثلا زمانی که می‌خواهیم روی لیستی از قیمت‌ها ۹ درصد مالیات اعمال کنیم.

برای `reduce`: کل آرایه را به یک مقدار نهایی تبدیل می‌کند. برای مثال زمانی که می‌خواهیم حاصل جمع تمام قیمت‌های درون یک لیست (فرضاً سبد خرید) را ببینیم.

منبع: اسلاید ۴۷ JavaScript

۵. (الف) به کد زیر دقت کنید:

```python
# file: sample.py
import m
import importlib

ab = m.f(0.1, 0.2)
c1 = float(ab) == (0.3)
bb1 = "meat is good"
bb1.replace("meat", "egg")
bb = bb1[:3]
c2 = float(m.f(0.1, 0.2)) == (0.3)
```

```python
# file: m.py
f = 0.3

def f(a, b):
    return str(a+b)
```

سپس درستی یا نادرستی عبارات زیر را به همراه دلیل درستی یا نادرستی هر کدام بنویسید. (۹ نمره)

گزاره	د/ن	دلیل
در کد بالا اگر متغیر c1 را چاپ کنیم، مقدار چاپ شده برابر با False خواهد بود.	د	رجوع به قسمت اول فایل جوپیتر در کدهای کلاسی Python
اگر متغیر bb را چاپ کنیم، مقدار چاپ شده برابر با egg خواهد بود	ن	رجوع به اسلاید ۳۱ Python
اگر حین اجرای برنامه قبل از رسیدن به خط آخر تابع f را عوض کنیم، مقادیر متغیرهای c1 و c2 قطعاً برابر خواهند بود.	د	رجوع به اسلاید ۴۵ Python

(ب) الگوی طراحی Decorator را توصیف کرده (۳ نمره) و بگویید چگونه می‌توان در پایتون یک دکوراتور برای یک تابع تعریف و از آن استفاده کرد (۵ نمره). (جمعاً ۸ نمره)

عینا در اسلاید آخر پایتون قید شده است:

A decorator is a design pattern that allows you to modify or extend the behavior of a function or method without changing its actual code.

کد تعریف و استفاده هم دقیقا در همان اسلاید آمده:

```python
def my_decorator(func):
    def wrapper():
        print("before...")
        func()
        print("after...")
    return wrapper

@my_decorator
```

```python
def say_hello():
    print("hello")

say_hello()
```

البته نیازی به نوشتن کد دقیق نبود و همین که ساختار کلی توصیف میشد کفایت میکرد. (مثلاً تابعی به نام `f` که یک تابع `g` را به عنوان پارامتر میگیرد و در خود یک تابع به نام `wrapper` دارد و ...)

منبع: اسلاید ۶۲ Python

(ج) توضیح دهید چرا نوشتن وابستگی‌های یک پروژه‌ی پایتون (ماژول‌های خارجی) در یک فایل جدا مناسب‌تر است. نام این فایل معمولاً چیست؟ برای مثال، بنویسید چگونه می‌توان مشخص کرد پروژه‌ی پایتون ما نیاز به نسخه‌ی `0.1.2` ماژول `somemod` دارد؟ چگونه می‌توان تمام ماژول‌های موجود در فایل مربوطه را با یک دستور نصب کرد؟ (۸ نمره)

در اسلایدهای مبحث مقدمات پایتون دقیقا آمده:

It's best to keep your dependencies listed in a requirements.txt file, alongside their versions (e.g. `pandas==2.3.0`). Then, you can use: `pip install -r requirements.txt`

همچنین مزیت این کار نگهداری وابستگی‌ها برای اجراپذیری و نصب راحت بر روی سایر ماشین‌ها (خواه کاربران، توسعه‌دهندگان دیگر یا ...) است. مانند آنچه در درس AP در `pom.xml` دیدید.

منبع: اسلاید ۴۷ Python

(د) فرض کنید قرار است یک برنامه پایتون را به چند درگاه پرداخت متصل کنید. در مورد اطلاعات پرداخت‌کننده، درگاه پرداخت اول شماره کارت، کد ملی صاحب کارت و رمز کارت را دریافت می‌کند. درگاه پرداخت دوم فقط ایمیل صاحب حساب را دریافت می‌کند. درگاه پرداخت سوم نیز یک کد هگزادسیمال با ۲۰ کاراکتر دریافت می‌کند. درگاه پرداخت اول مبلغ را به ریال، درگاه دوم مبلغ را به دلار و درگاه سوم مبلغ را به تتر (ارز دیجیتال) می‌پذیرد. از شما می‌خواهیم تعدادی کلاس (حداقل `PaymentProcessor` و `PaymentMethod`) را طوری پیاده کنید که صرفا با فراخوانی `PaymentProcessor.pay(...)` عملیات پرداخت متناسب با موارد گفته شده انجام شود. دقت کنید که نیازی به نوشتن کد پایتون نیست. صرفا موارد زیر را بنویسید:

- نام کلاس‌های مورد نیاز
- نام فیلدها یا اتریبیوت‌های هر کلاس
- نام متدهای هر کلاس
- تایپ فیلدها (اتریبیوت‌ها)، پارامترها و خروجی متدهای هر کلاس

در صورت نیاز می‌توانید توضیحات تکمیلی نیز بنویسید. مجدداً تاکید می‌شود که نیاز به نوشتن کد دقیق با سینتکس بی‌نقص نیست و هدف سوال تنها بررسی درک شما از طراحی شی‌گرا در زبان پایتون و مسلط بودن بر تایپ‌های موجود در این زبان است. (۱۰ نمره)

تقریبا هر طراحی معقولی مورد پذیرش است و نمره کامل دریافت می‌کند. مهم است حتما تعدادی تایپ از زبان پایتون در سوال مشخص شود (۲ بار در سوال قید شده) و همچنین دو کلاس حداقلی گفته شده تعریف و استفاده شوند. اشاره و استفاده به `staticmethod` یا `abc` یا موارد مشابه می‌تواند مناسب باشد. همچنین ارتباط میان کلاس‌های گفته شده می‌بایست به دقت توصیف شوند.

منابع:

- اسلاید ۵۷ تا ۶۱ Python
- بخش Classes (۷) در فایل جوپیتر کدهای کلاسی بخش Python