
بسم الله الرحمن الرحيم

دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

گزارش کارآموزی در شرکت مهیمن

حدیث غفوری

استاد کارآموزی

سرکار خانم دکتر محمودزاده

تابستان ۱۴۰۱

فهرست مطالب

چکیده

در دوره‌ی کاراموزی در شرکت مهمین تجربه های زیادی کسب کردم. در کارگاه‌های باکیفیت اموزشی شرکت کردم پیاده سازی صفرتاصد یک وب‌سایت را به صورت گروهی انجام دادم و با اصول درست کدنویسی آشنا شدم.

در این گزارش ابتدا به معرفی شرکت مهمین و دستاوردها و زمنیه‌ی فعالیتش پرداختم. سپس مراحل مختلف کاراموزی و فعالیت ها در طول دوره را شرح دادم. درباره‌ی پروژه‌های دوره و فازبندی‌های مختلف اموزش و پروژه صحبت کردم و در انتها به یکی از مهم ترین مباحث زبان جاوااسکریپت یعنی مفهوم غیر همزمانی پرداختم.

فصل اول

معرفی شرکت ارتباطات و فناوری اطلاعات مهیمن

1-1 مقدمه

شرکت فناوری اطلاعات و ارتباطات مهیمن در سال 1388در یک گروه پژوهشی برآمده از دانشگاه صنعتی شریف آغاز به فعالیت نمود. هدف اولیه از تاسیس ارائه ی راهکارهای دانش بنیان برای صنایع کشور بود.

در طی سال های فعالیت ، شرکت دارای چندین رتبه و دستاورد داخلی در صنعت فناوری اطلاعات و ارتباطات است که از بین موارد متعدد می توان به ثبت 9محصول و خدمت دانش بنیان در حوزه فناوری اطلاعات و ارتباطات و گواهینامه نظام ملی مدیریت امنیت اطلاعات از سازمان فناوری اطلاعات اشاره کرد.

2-1 گروه های مختلف فعال در شرکت

در حال حاضر شرکت دارای گروه های توسعه متعدد با تمرکز روی مشکلات و دغدغه های کلان کشور است.

1-2-1 گروه تحلیل کلان داده

ماموریت انی گروه، توسعه فناوری ها و راهکارهای مورد نیاز سازمان های بزرگ در حوزه مدیریت، پردازش و تحلیل داده های انبوده است. راهکارهای متنوع این گروه نظیر هوش تجاری و عملیاتی، کشف تقلب، تضمین درآمد پیش بینی رفتار مشتری، نگهداری و تعمیرات پیش گویانه و ... که در طراحی آن ها از موفق ترین تجارب جهانی الگوبرداری شده، در صنایع مختلفی از جمله مخابرات، بانکداری، حمل و نقل، رسانه و هم چنین نهادهای حاکمیتی با موفقیت به بهره برداری رسیده اند.

1 -2- 2 گروه مدیریت فرآیندهای سازمانی

ماموریت این گروه توسعه محصولاتی برای مدیریت فرآیندهای سازمانی مبتنی بر استانداردهای BPM است. این

گروه موفق به توسعه محصولات بومی قدرمتند BPMS و Case management شده که در شرکت ها و سازمان‌های متعدد برای IT base نمودن فرایند های سازمانی خود، مورد استفاده قرار گرفته است.

3-2-1 گروه ابری

ارائه ی سکوهای نرم افزاری و سخت افزاری در قالب خدمات Database، IaaS، Paasو CI/CDو زیرساخت های مدیریت داده در مقیاس وسیع و تامین برقراری ارتباط با این سکوها از ماموریت های اصلی و خدمات این گروه است.

4-2-1 گروه کنترل دیجیتال

اعمال سیاست های مدیریتی بر روی جریان های داده بزرگ با tps بسیار بالا و ایجاد زیرساخت یک پارچه سازی سامانه های سازمانی، نیاز به بومی سازی تکنولوژی های این حوزه دارد که این مهم ترین ماموریت این گروه است. زیرساخت های توسعه داده شده ی این گروه در پروژه های مختلف تعهدات تا 100k tpsرا عملیاتی نموده است.

5-2-1 گروه امنیت سایبری

خدمات و محصولات قابل ارائه توسط این گروه شامل معماری امنیت سازمان ها و سامانه ها، ارزیابی امنیتی نرم افزار و زیرساخت، مشاوره، پیاده سازی و پشتیبانی سیستم مدیریت امنیت اطلاعات طراحی راهکارهای امنیتی و امن سازی سیستم ها و سامانه ها است که با توجه به نوع خدمات و محصولات، مجوز های مربوطه از سازمان ها و نهادهای تایید کننده اخذ شده است.

3-1 برخی از پروژه های شاخص مهمین

طراحی و پیاده سازی و نگهداری :

- سامانه شناسایی گوشی تلفن همراه و تجهیزات دارای سیم کارت(رجیستری)
- سامانه شبکه احراز هویت کاربران ارتباطی ایران (شاهکار)
- سامانه مدیریت تقلب اپراتورهای دولتی و خصوصی حوزه ارتباطی
- سامانه مدیریت تقلب و تضمنی درآمد شرکت ارتباطات سیار ایران
- سامانه جامع قوانین و نظارت مجلس شورای اسلامی
- و ده ها پروژه دیگر

فصل دوم
شرح فعالیت های زمان کارآموزی

مسابقات کاراموزی کداستار در تابستان هرسال در وب سایت Quera برگزار میشود و پس از سه مرحله مصاحبه و بررسی، افراد منتخب به کاراموزی دعوت می‌شوند.

این مسابقات در زمینه فرانت اند و مهندسی نرم افزار برگزار شده و پس از اعلام نتایج، تیم ها برای انجام پروژه‌های گروهی مشخص می‌شوند.

بنده درکداستار در زمینه فرانت‌اند پذیرفته شدم و دوره‌ی کاراموزی را شروع کردم.

2-1 فرایندهای تیمی

دوره‌ی کاراموزی شامل دو پروژه بود که در اولین پروژه به تیم‌های دونفره و در پروژه‌ی دوم به تیم‌ها هشت نفره شامل ۴ نفر از اعضای فرانت‌اند و ۴نفر از اعضای بک‌اند تقسیم شدیم.

در طول دوره‌ی کاراموزی، اعضای هر دوتیم، دو منتور داشتیم و منتورها در چالش‌های فنی پروژه، ما را راهنمایی میکردند. علاوه بر این هرروز پس از پایان روز کاری با منتورها، جلسه داشتیم و در جلسه چالش‌ها و دغدغه‌های خود را مطرح میکردیم.

منتورها سعی میکردند هرروز نکات جدیدی را در جلسه بیان کنند که این نکات شامل مهارت های فنی و مهارت های نرم بود. به عنوان مثال سایت‌ها و کتاب هایی در زمینه برنامه نویسی معرفی می‌کردند یا یک سری نکات در زمینه توسعه‌ی شخصی که بسیار مفید بود بیان می‌کردند.

2-2 کارگاه‌های اموزشی

منتورها براساس فازهای ارایه شده برای کاراموزی، کارگاه‌های اموزشی برگزار می‌کردند از جمله:

- کارگاه آموزش html,css, scss.
- کارگاه آموزشjavascript javascript,advanced .
- کارگاه آموزش پیاده‌سازی صفحات Responsive.

- کارگاه آموزش Angular.

3-2 فازهای مختلف دوره‌ی کاراموزی

در این بخش قصد دارم موضوعات و مفاهیمی که در دورهی کاراموزی یادگرفتم را شرح دهم.

دورهی کاراموزی شامل فازهایی برای آموزش مفاهیم مختلف فرانت‌اند بود که در کنار شرکت در کارگاه‌های اموزشی و یادگیری این مفاهیم از این اموخته ها در پروژه ها هم استفاده می‌کردیم و با طی کردن گام به گام این فازها به صورت عملی از اموزش ها در پروژه استفاده می‌کردیم.

1-3-2 مرور مفاهیم html

html یک زبان نشانه گذاری و اسکلت اصلی تمامی سایت هایی است که در سراسر دنیا به وجود می آیند است. html یک زبان بسیار ساده است و پیشنیاز ورود به دنیای طراحی وب می باشد.

html یک زبان برنامه نوسی نیست و در اصل تنها برای نمایش محتوا در ساختار های از پیش تایین شده ساخته شده است.

زبان html هیچ ظاهری ندارد و به کمک مکمل خود یعنی CSS، ظاهر و شکل و شمایل می گیرد.

در این فاز با نکات مهم و کاربردی این زبان آشنا شدیم و تمرکز این بخش بر یادگیری نکات فراموش‌شده و کاربردی این زبان بود.

- **یادگیری Semantic Elements**

در HTML تگ‌هایی وجود دارند که صرفاً برای یک کار خاص طراحی شده‌اند و با دیدن نام آن‌ها می‌توان کاربردشان را متوجه شویم. به این تگ‌ها، تگ‌های معنایی گفته می‌شود.

- header
- nav
- main
- aside
- footer
- article
- section
- details
- summary

- **Text Formatting Elements**

بعضی از تگ‌ها مانند b و i برای تغییر ظاهر بخشی از متن استفاده می‌شوند مانند:

- b & strong
- i & em
- mark
- small
- del
- ins
- sub
- sup

کاربرد اصلی این تگ‌ها برای استفادهٔ سریع هنگام نوشتن محتوای سایت است. به عنوان مثال هر وقت خواستیم یک کلمه را bold کنیم به جای اینکه از span در html و یک class در css، استفاده کنیم می‌توانیم از b یا strong در html استفاده کنیم، بدونِ آنکه نیاز باشد تغییری در کد css دهیم.

- **نکات کار با تصاویر در html**

همان طور که می‌دانیم در اکثر وب‌سایت ها از تصاویر استفاده می‌شود بنابراین موضوع بسیار مهم وکاربردی هستند. ما در این بخش با نکات مهم نحوه‌ی استفاده درست و بهینه از تصاویر آشنا شدیم تا بتوانیم وب‌سایت زیباتر به همراه پرفورمنس بالایی داشته باشیم.

در اینجا قصد دارم به برخی از نکات مهم درباره‌ی تصاویر که در دوره یادگرفتم بپردازم:

Alt attribute

alt در اینجا مخفف عبارت Alternate Text یا متن جایگزین می‌باشد که دو کاربرد مهم برای آن می‌توان در نظر گرفت:

- همانطور که از اسم آن مشخص است، جایگزینی برای عکس می‌باشد به طوری که اگر به هر دلیلی امکان نمایش عکس وجود نداشت، این نوشته به کاربر نشان داده می‌شود تا موضوع عکس را بفهمد.
- Screen Readerها متن مورد نظر را می‌خوانند بنابراین کاربرانی که نمی‌توانند عکس را ببینند، متوجهِ موضوع عکس می‌شوند.

بنابراین استفاده از alt برای عکس‌ها بسیار حائز اهمیت است.

Performance

ما تقریباً در تمام سایت‌ها از عکس استفاده می‌کنیم؛ بنابراین همیشه باید به مبحث Performance توجه داشته باشیم، چرا که در غیر این صورت تجربهٔ بدی در انتظار کاربران ما خواهد بود.

یکی از بهترین راه‌ها برای افزایش Performance، استفاده از عکس‌های باکیفیت و در عینِ حال کم‌حجم است. برای این موضوع ما از سایت Squoosh که توسط توسعه‌دهندگان Google ساخته شده، استفاده می‌کنیم. همچنین این سایت قابلیت تبدیل به انواع فرمت‌ها و استفاده از الگوریتم‌های گوناگون را دارد.

همچنین در صورت امکان سعی می‌کنیم حتماً از فرمت svg استفاده کنیم. این فرمت برخلاف فرمت‌هایی مثل png و jpeg، به صورت vector یا برداری است، به طوری که می‌توانیم بدونِ افت کیفیت، به هر اندازه آن را کوچک یا بزرگ کنیم. فرمت svg همواره از باقی فرمت‌ها کم‌حجم‌تر است اما در عوض، جزئیات کمتری را می‌توانیم در آن جای دهیم. بنابراین بهترین استفاده از svg در لوگوها، Illustrationها و آیکن‌هاست.

2-3-2 آشنایی با مفاهیم مهم CSS

سی اس اس مخفف Cascading Style Sheet (CSS) است. زبان css یک زبان طراحی صفحات وب برای ایجاد و ساخت مشخصات ظاهری اسناد و اطلاعات وب سایت می باشد. css یکی از رایج ترین و محبوب ترین ابزارهای طراحی صفحات وب سایت نوشته شده توسط زبان HTML و یا XHTML است.

در کدنویسی با استفاده از CSS می‌توانید استایل سایت مثل رنگ، فونت، تصاویر پس زمینه و ... را بصورت دلخواه تغییر دهید.

هدف از تولید css در واقع جداسازی اطلاعات محتوا (که توسط زبانی مانند HTML نوشته شده اند) از اطلاعات ظاهری مانند صفحه بندی، رنگ و سایز و نوع فونت می باشد. این جداسازی موجب افزایش سرعت در دسترسی به سایت، انعطاف پذیری بیشتر برای کنترل ویژگی های ظاهری، قابلیت طراحی چندین صفحه با یک فرمت یکسان و جلوگیری از پیچیدگی و انجام کارهای تکراری در طراحی وب سایت می گردد.

در این فاز با مفاهیم مهم CSS از جمله موارد زیر آشنا شدیم:

- Box Model چیست و از چه اعضایی تشکیل شده است.
- هر کدام از واحدهای مختلف CSS چه مفهومی دارند و در چه جاهایی باید استفاده شوند.

- انواع Selectorهای CSS چه چیزهایی هستند و چگونه می‌توان از آن‌ها استفاده کرد.
- کلاس‌های Pseudo چه چیزهایی هستند و چه کاربردی دارند.
- نحوه‌ی Responsive کردن وب‌سایت و قابلیت نمایش در صفحه‌های متفاوت

3-3-2 آشنایی با SCSS

SCSS یک Preprocessor برای CSS است که با استفاده از آن کار توسعه‌دهنده به شدت آسان می‌شود. در این فاز با قابلیت‌های معروف SCSS و کاربرد هر یک از آن‌ها آشنا شدیم.

از قابلیت های معروف scss می‌توان به موارد زیر اشاره کرد:

- **Nested Selectors**

شاید بتوان گفت Nested Selection معروف‌ترین و پراستفاده‌ترین مزیت SCSS نسبت به CSS است. به کمک این قابلیت می‌توانیم حجم کد خود را کمتر کنیم.

- **Variables**

در SCSS هم مانند CSS می‌توانیم متغیر داشته باشیم. متغیرهای SCSS به این شکل تعریف می‌شوند:

```
$color-primary: hsl(220, 100%, 55%);
$padding--large: 4rem;
```

شکل (2-1)

البته متغیرهای SCSS با متغیرهای CSS تفاوت‌های اساسی دارند:

- مقادیر اختصاص داده شده به متغیرهای SCSS در زمان Compile جایگزین می‌شوند در صورتی که متغیرهای CSS به همان شکل در کد نهایی استفاده می‌شوند.
- متغیرهای CSS برای المان‌های مختلف می‌توانند دارای مقادیر متفاوت باشند اما متغیرهای SCSS در آنِ واحد تنها یک مقدار دارند.
- مقادیر متغیرهای SCSS به صورت قطعی در نظر گرفته می‌شوند به طوری که اگر از متغیر در جایی از کد استفاده کنیم و در جای دیگر مقدار آن را عوض کنیم، مقدار جدید جایگزین مقدار قبلی نخواهد شد.

- **Mixins**

زمانی که قطعه‌ای از کد را بخواهیم در مکان‌های مختلف استفاده کنیم می‌توانیم از Mixinها کمک بگیریم.

به عنوان مثال فرض کنیم در دو جا بخواهیم یک Layout با سه ستون هم‌عرض داشته باشیم. برای این کار قطعه کد

زیر را می‌نویسیم:

```
.products {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}

.items {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

شکل (2–2)

همان‌طور که می‌بینیم قطعه‌ای از کد عیناً تکرار شده است. می‌توانیم کد بالا را به شکل زیر بازنویسی کنیم:

```
@mixin layout-with-3-equal-columns {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}

.products {
  @include layout-with-3-equal-columns;
}

.items {
  @include layout-with-3-equal-columns;
}
```

شکل (2–3)

به کمک این روش ها می‌توانیم حجم کد را کمتر کنیم و اصول Clean code را هم رعایت کنیم.

4-3-2 آشنایی با مفاهیم مقدماتی JavaScript

مشهورترین زبان برنامه‌نویسی در وب، جاوا اسکریپت(JavaScript) است. با استفاده از این زبان می توانیم صفات استاتیک را پویاسازی کنیم، وب فرم‌های پیچیده را ساده سازی و یا ویژگی‌های تعاملی (interactive) را به وب سایت خود اضافه کنیم.

در این فاز با مفاهیم نسبتاً سادهٔ JavaScript از جمله موارد زیر آشنا شدیم:

- چرا برای تعریف متغیر از سه عبارت var و let و const می‌توان استفاده کرد.
- تفاوت Function معمولی با Arrow Function چیست.
- مفهوم و کاربرد this چیست.
- DOM و راهکارهای مختلف برای اتصال و ایجاد تغییر در آن چه چیزهایی هستند.

5-3-2 آشنایی با مفاهیم Best Practice

در این فاز با بهترین روش‌های توسعهٔ فرانت‌اند آشنا شدیم.

- کد تمیز چه ویژگی‌هایی دارد.
- تفاوت UI و UX چیست.
- کدام رنگ‌ها هم‌نشینی بهتری با یکدیگر دارند.
- چه فونت‌هایی برای استفاده در سایت پیشنهاد می‌شوند.

Clean Code

کدنویسی تمیز(Clean Code) مجموعه اصولی است که به ما کمک می‌کند کدی بنویسیم، که فهم و اصلاح آن توسط دیگران و یا بعد از مدت‌های طولانی برای خودمان سخت نباشد. این اصول استانداردهایی هستند که اگر رعایت شوند، ما را تبدیل به یک برنامه نویس حرفه ای می‌کنند. کدنویسی تمیز در کار گروهی اهمیت بسیار زیادی دارد.

در متدولوژی هایی مانند چابک(Agile) کدنویسی تمیز ضروری به نظر می‌رسد زیرا اعضای تیم، در بسیاری از موارد ناچار به تکمیل و اصلاح کدهای یکدیگر هستند. کد کثیف(Dirty Code) اصطلاحی است که در برابر کدنویسی تمیز وجود دارد. عدم رعایت اصول کدنویسی تمیز، بی نظمی و استفاده از رویه‌های تکراری کد ما را به Dirty Code تبدیل می‌کند.

ما در این بخش یادگرفتیم که چرا تمیز بودن کد و داشتن معماری خوب مهم است و چرا باید وقت و انرژی زیادی صرف طراحی و نوشتن کد تمیز شود وچرا این موضوع در صنعت، بسیار بیشتر از سایر پروژه‌ها اهمیت دارد.

Code Smells

یکی از راه‌های افزایش کیفیت کد، پیدا کردن نشانه‌های کد کثیف در برنامه و سپس Refactor کردن برنامه است به صورتی که نشانه‌های کد کثیف از بین بروند. به این نشانه‌ها Code Smell می‌گوییم.

در زیر لیستی از Code Smellهای مختلف اورده شده:

- Code smell
- Long Method
- Primitive Obsession
- Large Class
- Long Parameter List
- Temporary Field
- Duplicate Code

در این بخش ما با این code smellها آشنا شدیم و تلاش کردیم کدهای کثیفی که در پروژهمان ایجاد شده بود را حذف کنیم.

Design

UI/UX

UIیا User Interface به معنای رابط کاربری، مربوط به المان‌هایی است که کاربر با آن‌ها ارتباط دارد UX یا User Experience به معنای تجربهٔ کاربری، مربوط به کیفیتی است که کاربر هنگام استفاده از محصول تجربه می‌کند.

White Space

المان‌های ما باید فضای کافی برای تنفس داشته باشند. هیچوقت نباید حجم انبوهی از اطلاعات را در یک مکان کوچک جا دهیم چرا که کاربر نمی‌تواند به اندازهٔ کافی تمرکز کند و این مسئله برایش عذاب‌آور خواهد بود.

برای ایجادWhite Space ، راحت‌ترین کار استفاده از margin و padding است. همچنین در flex و grid می‌توانید از gap استفاده کنید.

Color

رنگ‌ها و روانشناسی آن‌ها موضوعی است که سال‌ها طراحان را درگیر خود کرده. هر رنگ نشان‌دهندهٔ مجموعه‌ای از مفاهیم است که به آن روانشناسی رنگ‌ها گفته می‌شود. به عنون مثال رنگ آبی نشانهٔ صلح، آرامش، اعتماد، امنیت و

... می‌باشد؛ به همین خاطر خیلی از شرکت‌های بزرگ دنیا مانند facebook و twitter از این رنگ در محصولات خود استفاده می‌کنند.

همچنین زمانی که بخواهیم بیش از یک رنگ داشته باشیم، باید دقت کنیم که رنگ‌های انتخابی ما یکدگیر را تکمیل کنند.

اگر از دو رنگ استفاده می‌کنیم، معمولاً توصیه می‌شود رنگ‌های متضاد را انتخاب کنیم. به عنوان مثال اگر رنگ اول ما آبی است، رنگ دوم را می‌توانیم نارنجی بگذاریم.

Contrast

کنتراست موضوعی است که به شدت با رنگ‌ها در ارتباط است. کنتراست یعنی تضاد بین رنگ‌ها. به عنوان مثال اگر دکمه‌ای داشته باشیم، می‌توان تضاد بین رنگ نوشته و رنگ پس‌زمینه را در نظر گرفت.

استانداردی به نام WCAG وجود دارد که بیان می‌کند چه نسبتی بین نوشته و پس‌زمینه باید وجود داشته باشد. این استاندارد شامل دو حالت AA و AAA است که حالت دوم رتبه‌بندی بهتری دارد و در صورت امکان باید آن را در نظر گرفت.

همچینن میزان کنتراست به اندازهٔ متن نیز وابسته است. به عنوان مثال در رتبه‌بندی AA اگر اندازهٔ متن معمولی باشد، نسبت کنتراست باید حداقل 4.5 به 1 باشد؛ اما اگر اندازهٔ متن بزرگ باشد، این نسبت می‌تواند تا 3 به 1 کاهش پیدا کند.

Visual Hierarchy

زمانی که کاربر وارد صفحه می‌شود، با چشمانش المان‌های صفحه را به ترتیب بررسی می‌کند که به آن Visual Hierarchy گفته می‌شود. همواره المان‌های مهم‌تر باید رتبهٔ بالاتری در این رده‌بندی داشته باشند. به عنوان مثال تیتر یک صفحه از اهمیت بالایی برخوردار است اما تاریخ نگارش آن ممکن است خیلی مهم نباشد.

برای بالا بردن رتبهٔ یک المان و جلب توجه کاربر از موارد متعددی مانند سایز و وزن فونت، فضای تنفس، رنگ و انیمیشن می‌توانیم استفاده کنیم.

Alignment

Alignment یا تراز بودن، به شدت ارتباط تنگاتنگی با Visual Hierarchy دارد و اگر بتوان از این دو به درستی استفاده کرد، شاهد یک UI و UX حرفه‌ای خواهیم بود.

المان‌هایی که کنتراست بالایی دارند، باید در یک ستون نسبت به المان‌های بالا و پایین خود قرار داشته باشند. به این ترتیب می‌توانیم یک ساختار مشخص ایجاد و به بهبود تمرکز کاربر کمک کنیم.

در اکثر مواقع توصیه نمی‌شود المان‌ها را به صورت وسط‌چین تنظیم کنیم؛ مگر اینکه فضای کوچکی مانند صفحه نمایش تلفن همراه در اختیار داشته باشیم.

Typography

تایپوگرافی چیزی فراتر از انتخاب یک فونت مناسب است. تمام سایت‌ها از متن استفاده می‌کنند. این متن می‌تواند در جایی مثل تیتر، پاراگراف و دکمه استفاده شود. بنابراین تنظیم‌کردن خصوصیات این متون باید به بهترین شکل ممکن انجام شود.

همچنین برای پاراگراف‌ها و متن‌های طولانی توصیه می‌شود با توجه به نوع مطلب، عرض خطوط بین 40 تا 70 کاراکتر باشد. این مسئله به شدت به خوانایی متن کمک می‌کند.

Simplicity

تا جایی که می‌توانیم باید سعی کنیم ذهن کاربر را با افکت‌های عجیب و غریب مشغول نکنیم. باید تا جای ممکن همه چیز کاملاً ساده و گویا باشد.

به کار بردن افکتی که به تازگی آن را یاد گرفته‌ایم ممکن است برای ما بسیار جذّاب باشد، اما برای اکثر کاربران صرفاً باری اضافه و امتیازی منفی محسوب می‌شود.

از سایه‌ها فقط و فقط باید در جایی که به آن احتیاج داریم استفاده کنیم. پیشنهاد می‌شود از border و outline استفاده نکنیم، اما اگر بخواهیم، باید نوع، اندازه و رنگ آن باید طوری انتخاب کنیم که توجه زیادی را از کاربر نگیرد . Gradientها فقط زمانی خوب به نظر می‌آیند که بی‌نقص پیاده‌سازی شده باشند؛ بنابراین در حالت کلی بهتر است از رنگ‌های Solid استفاده کنیم.

از طرفی باید دقت کنیم که طرح ما Over-simplified نباشد. این اتفاق بر سر بسیاری از لوگوها آمده و کاربران از آن بیزارند. همه چیز باید به اندازه باشد تا بهترین UI و UX حاصل شود.

6-3-2 آشنایی با جاوااسکریپت پیشرفته

تا این فاز ، با مفاهیم ابتدایی Java Script آشنا شدیم. در این فاز مباحث پیشرفته‌تر Java Script را یاد گرفتیم تا در انتهای فاز بتوانیم پروژه‌هایمان را به APIآهنگ‌ها متصل کنیم و با سرور در ارتباط باشیم.

حافظه وStorage

در این فاز ما با انواع حافظه در صفحه‌های آشنا شدیم.

آشنایی با اینکه اطلاعات کاربرانی که وارد سامانه می‌شوند چطور در سرتاسر برنامه ذخیره شود؟

چطور اطلاعات کاربران را در طول زمانی که در برنامه هستند نگه داریم و از آن‌ها استفاده کنیم؟

چگونه سلیقه کاربران را به خاطر بسپاریم تا آهنگ‌های مناسب را به آن ها پیشنهاد دهیم؟ برای این کارها، از مفاهیمی به نام حافظه در وب استفاده می‌کنیم. حافظه‌ها می‌توانند انواع مختلفی داشته باشند.

- **Cookie**

کوکی‌ها می‌توانند به دو صورت استفاده شوند:

- سمت سرور :(HTTP Cookie)
- با جاوا اسکریپت و استفاده از document.cookie

- **Session storage**

Session storage نوع دیگری از حافظه در وب است. زمانی که پنجره و تب بسته می شود، اطلاعاتی هم که توسط session storage ذخیره شده هم پاک می شود. همچنین باید توجه داشته باشیم که اطلاعات تب‌های دیگر در session storage قابل استفاده نیست.

- **Local storage**

حافظه لوکال از بسیاری از جهات شبیه session storage است. با این تفاوت که برخلاف session storage، با بستن تب از بین نمی‌رود و هیچ تاریخ انقضایی ندارد. ست کردن و گرفتن دیتا در local storage دقیقا مشابه session storage است.

JSON

جیسون یا جیسان، یک فرمت سبک برای انتقال اطلاعات است. معمولا از JSON برای رد و بدل کردن اطلاعات بین سرور و صفحه‌ی وب استفاده می‌شود .

HTTP

پروتکل HTTP مخفف عبارت Hyper Text Transfer Protocol است و به ارتباط میان سرویس‌دهنده (server) و سرویس‌گیرنده (client) در وب می‌پردازد. ارتباط بین سیستم‌های سرویس‌گیرنده و سرورها از طریق ارسال درخواست‌های HTTP و دریافت پاسخ‌های HTTP انجام می‌شود. به زبان ساده‌تر، HTTP یک نوع قانون است که ارسال و دریافت اطلاعات بین client و server بر اساس آن انجام می‌شود.

RESTful Api

Application Programming Interfaceکه با رابط برنامه‌نویسی کاربردی ترجمه می‌شود یک مجموعه از قواعد و مکانیزم‌ها است که از طریق آن اپلیکیشن‌ها و یا کامپوننت‌های مختلف یک برنامه با همدیگر ارتباط برقرار می‌کنند. نام خود این مکانیزم بیانگر همه چیز است. منظور از رابط چیزی است که دو شئ یا دو موجودیت مختلف را به همدیگر ربط می‌دهد. API می‌تواند داده‌هایی که ما برای اپلیکیشن‌مان نیاز داریم را از طریق یک فرمت مناسب به خروجی بفرستد و یا آن را برگشت دهد. فرمت JSON و XML از این دست فرمت‌ها هستند.

غیر همزمانی‌ها در جاوااسکریپت

- **Promise**
  - Promiseها برای عملکردهای asynchronous به کار ما می‌آیند. با promise ما منتظر نتیجه می‌مانیم و با then() مشخص می‌کنیم که بعد از آن چه اتفاقی رخ بدهد.
- **async, await**
  - در صورتی که پشت نام یک تابع async به کار رود، آن تابع یک Promise به عنوان خروجی برمی‌گرداند.Async و awaitیک راه تمیزتر و ساده‌تر برای استفاده از Promiseها هستند.
- **fetch**
  - fetch Api در جاوا اسکریپت برای ارسال درخواست به سمت سرور و لود کردن اطلاعات و جواب‌های سرور به کار می‌آید.

History API

شی window.history، نگهدارنده‌ی history مرورگر شما است. دو متد اصلی آن:

- history.forward()
- history.back()

npm

در فاز سوم با npmکار کردیم . npm یک package manager برای پکیج‌های javascript است. با دستور npm init فایل package.json برای اپلیکیشن ما ساخته می‌شود. این فایل حاوی متادیتای اختصاصی پروژه همچون نام نویسنده، ایمیل، نام پروژه و اطلاعات مربوط به پکیج‌ها و اپلیکیشن‌هایی است که برنامه ما از آن استفاده می‌کند. در صورتی که فایل package.json در پروژه‌ای وجود داشته باشد، با دریافت پروژه و زدن npm install تمام پکیج‌های مورد نیاز برنامه روی دستگاه ما نصب می‌شود.

Unit Test

در این فاز با Unit Test، مفهوم Code Coverage و مفهوم TDD آشنا شدیم. در این فاز، برای کدی که در فازهای قبلی نوشتیم، تست نوشتیم و با استفاده از مفاهیم Coverageو اصول تست‌نویسی، کیفیت کد خود را بیش از پیش افزایش دادیم.

مفاهیمی که در این فاز اموختیم:

- Unit Test و اهمیت آن
- مفهوم Code Coverage در Unit Testing و اهمیت آن
- تاثیر اصول SOLID بر Unit Tesing
- آشنایی با Mock
  - Mockکردن، یک تکنیک تست‌نویسی است که در آن، قسمتی از کد را با یک پیاده‌سازی دلخواه جایگزین می‌کنیم و از آن برای شبیه‌سازی یک عملیات واقعی استفاده می‌شود. معمولا Mockکردن زمانی استفاده می‌شود که یک متد یا کلاس، وابستگی یا وابستگی‌هایی دارد که در تست ما تداخل نامطلوبی ایجاد می‌کند.
- آشنایی با Karma و Jasmine

Angular 7-3-2

یکی از محبوب‌ترین فریم‌ورک‌های طراحی وب، انگولار است که توسط تیمی در گوگل، طراحی و پیاده‌سازی شده است. این فریم‌ورک به جای جاوااسکریپت، از زبان تایپ اسکریپت استفاده می‌کند. این زبان توسط مایکروسافت طراحی شده است و یک Superset روی جاوااسکریپت است به این معنی که تمام موارد داخل جاوااسکریپت را

دارد و علاوه بر آن، موارد بیشتری را نیز پشتیبانی می‌کند. برای مثال در تایپ‌اسکریپت بر خلاف جاوااسکریپت، تایپ وجود دارد و ما می‌توانیم برای متغیرهای خود تایپ تعیین کنیم.

همچنین در تایپ‌اسکریپت، Interface و Abstract Class پشتیبانی می‌شود که به وسیله‌ی آن‌ها می‌توان کد تمیزتری که از اصول SOLID استفاده می‌کند، ایجاد کرد. در انگولار، کد به زبان Typescript نوشته می‌شود و در انتها به زبان Javascript کامپایل می‌شود، زیرا تنها زبانی که مرورگرها پشتیبانی می‌کنند، زبان Javascript است.

Angular CLI

انگولار به همراه خود CLI مخصوص خود را دارد که ایجاد پروژه، کامپوننت، ماژول و ... را برایDeveloper ها آسان‌تر می‌کند .

Module

ماژول مجموعه‌ای از کامپونت‌ها و سرویس‌ها است که با یکدیگر وابستگی نزدیکی دارند. هر پروژه انگولار حداقل یک ماژول دارد که به آن AppModule می‌گویند و در ابتدا ساخته می‌شود. در این ماژول مشخص می‌شود که کامپونت شروع‌کننده برنامه چیست. این کامپونت در قسمت Bootstrap مشخص می‌شود. اگر در داخل یک ماژول از ماژول دیگری استفاده شود، نام آن در Import می‌آید و نام تمام کامپونت‌های داخل ماژول نیز در Declarations مشخص می‌شود. اگر بالای هر کلاس تایپ‌اسکریپت از دکوراتور @NgModuleاستفاده شود، آن کلاس تبدیل به ماژول می‌شود.

Component

به کوچک ترین واحد قابل نمایش کامپونت گفته می شود که وظیفه ی کپسول سازی فایل های Html , CSS , TS مربوط به یک المان را دارد. هر پروژه‌ی انگولار از چندین کامپونت تشکیل شده است.

هر کامپونت معماری MVVM دارد و شامل یک فایلHTML ، یک فایل Style Sheet و یک فایل تایپ‌اسکریپت است. اگر بالای هر کلاس تایپ‌اسکریپت از دکوراتور @Componentاستفاده شود، آن کلاس تبدیل به کامپونت می‌شود. هر کامپونت می‌تواند تعدادی ورودی و یا خروجی داشته باشد. هر کامپونت باید در داخل Declarations یک ماژول تعریف شده باشد .

Service

برای دریافت داده و یا منطق که مخصوص یک کامپوننت خاص نیست، از Service استفاده می‌شود. در انگولار از Dependency Injection استفاده می‌شود. نحوه‌ی عملکرد آن به این صورت است که سرویس‌های مربوط به یک ماژول در قسمت Providers موجود در ماژول، مشخص می‌شود و در زمان اجرا، آن Service ساخته می‌شود و به کامپوننت‌های داخل ماژول داده می‌شود. بالای هر سرویس از دکوراتور Injectable@استفاده می‌شود. یکی از مهم‌ترین سرویس‌های موجود در انگولار، HttpClientاست. از این سرویس برای ارسال درخواست‌های Get و یا Post به سرور استفاده می‌شود.

8-3-2 آشنایی با انگولار پیشرفته

Routing

برنامه‌های انگولار به صورت SPA یا Single Page Application هستند. به این شکل که ما تنها یک صفحه در سایت خود داریم و محتوا به صورت داینامیک در همان صفحه لود می‌شود. این که محتوای چه قسمتی در صفحه لود شود، به کمک Routing مشخص می‌شود .

Two-Way Binding

مبحث Two-Way Binding به کامپوننت‌های انگولار این اجازه را می‌دهد تا داده بین خودشان به اشتراک گذارند. این اشتراک داده به صورت دوطرفه خواهد بود یعنی هر زمانی که تغییری در مقدار داده ایجاد شود، در همه جا این مقدار تغییر خواهد کرد.

Pipe

ازPipe ها برای تغییر شکل داده در صفحات HTML استفاده می‌کنیم. به طور مثال اگر فرض کنیم یک عدد به صورت 12345 داریم و قصد داریم این عدد را سه رقم، سه رقم جدا کنیم؛ برای این منظور از Pipe استفاده می‌شود. یک سریPipe های آماده در انگولار وجود دارد مانند DatePipe که برای نمایش بهتر تاریخ استفاده می‌شود یا UpperCasePipe که نوشته ما را به صورت UPPER CASE نمایش می‌دهد.

علاوه برPipeهای موجود در انگولار، ما می‌توانیم Pipe جدیدی ایجاد کنیم.

OGMA

Ogma کتابخانه گراف جاوا اسکریپت است. این کتابخانه یک موتور گرافیکی قدرتمند مبتنی بر WebGL ارائه می دهد و از ماشین های قدیمی با HTML5 Canvas پشتیبانی می کند.

Ogma تمام ویژگی های مورد نیاز برای نمایش، کاوش و تعامل با داده های نمودار در برنامه های کاربردی وب را ارائه می دهد.

از ویژگی هایی که این کتابخانه قدرتمند در اختیار ما می‌گذارد میتوان به موارد زیر اشاره کرد:

- اتصال به منابع مختلف برای importکردن و exportکردن داده،
- طرح‌بندی هوشمند داده‌ها
- تعامل غنی با کاربر
- سبک بصری کاملاً قابل تنظیم است.

در این فاز ما نحوه‌ی کار با این کتابخانه را برای پروژه‌ی دوم یادگرفتیم.

3-2 پروژه‌ی وب سایت spotify

در این پروژه ما باید از صفر تا صد یک وب سایت پخش موسیقی و جستجوی آهنگ‌ها را طراحی و پیاده‌سازی میکردیم. این پروژه در فاز های مختلفی تعریف شده بود.

فاز اول

مشخص کردن صفحات مختلف وب‌سایت که طراحی ui,ux وب‌سایت به عهده‌ی خودمان بود.

در این فاز فقط باید بدنه‌ی وب‌سایت را با html پیاده‌سازی میکردیم.

صفحات وب‌سایت شامل

- اصلی
  - وقتی کاربر برای اولین بار وارد سایت می‌شود، صفحهٔ اصلی را می‌بیند. بنابراین باید توجه او را جلب کنید به طوری که تمایل داشته باشد در سایت شما ثبت نام کند.
- ثبت نام
- ورود
- لیست آهنگ‌ها
  - لیستی از تمام آهنگ‌هایی که در سایت وجود دارد. می‌توانید برای بهبود تجربهٔ کاربری از صفحه‌بندی، فیلتر کردن، جستجو و هر مورد دیگری استفاده کید.
- لیست آهنگ‌های مورد علاقه

- آهنگ‌هایی که کاربر به عنوان مورد علاقه انتخاب کرده در این صفحه نمایش داده می‌شود. کاربر باید بتواند هر کدام از این آهنگ‌ها را از لیست حذف کند.

- آهنگ
  - در این صفحه کاربر می‌تواند جزئیات آهنگ مورد نظر را ببیند این جزئیات حداقل شامل موارد زیر است:
    - نام آهنگ
    - عکس آهنگ
    - نام خواننده
    - سال انتشار
  - همچنین کاربر می‌تواند این آهنگ را به لیست آهنگ‌های مورد علاقه اضافه یا از آن حذف کند.

همچنین ما می‌توانستیم علاوه بر این صفحه‌ها موارد بیشتری را طراحی و پیاده‌سازی کنیم و قابلیت های صفحات فعلی را گسترش دهیم. از نظر طراحی وب‌سایت تیم ها می‌توانستند هرگونه که علاقه دارند وب‌سایت خود را طراحی کنند و تعداد صفحه‌ها را زیادتر کنند که دادن این حق انتخاب به تیم‌ها باعث شد در پایان دوره وب‌سایت های فوق العاده متنوع با طرح های خلاقانه‌ای ساخته شود.

فاز دوم

با استفاده از SCSS ساختار HTML که در فاز یک طراحی کردیم را Style دهی کردیم.

در این فاز ما سعی کردیم از اکثر قابلیت‌های SCSS استفاده کنیم.به عنوان مثال با استفاده از Nesting خوانایی کد را افزایش و با استفاده از Mixin حجم آن را کاهش دادیم.

فاز سوم

با استفاده از مفاهیمی که از JavaScript یاد گرفتیم بخشی از منطق سایت را پیاده‌سازی کردیم.

از آنجایی که در این فاز هنوز با مفاهیم ارتباط با سرور آشنا نشده بودیم، مسائلی مانند Signup و Login را در فازهای بعدی انجام دادیم.

فاز چهارم

Code Smells

با توجه به توضیحات این فاز، سعی کردیم Code Smell هایی که در کدمان وجود دارد را شناسایی و اصلاح کنیم.

Design

با توجه به توضیحات این فاز استایل‌ها و ساختار HTML پروژه را به گونه‌ای اصلاح کردیم که موارد ذکر شده در این بخش به بهترین شکل ممکن پیاده‌سازی شده باشند.

Pull Request and Code Review

بعد از اینکه موارد بالا را اصلاح کردیم، یک PR ساختیم و اعضای یکی از تیم‌های دیگر را به عنوان Reviewer انتخاب کردیم و به آن‌ها اطلاع دادیم تا کد ما را بازبینی کنند.

سپس از منتورهای خود خواستیم کد ما را Review کنند و در نهایت آن را روی برنچ master بردیم.

فاز پنجم

در این فاز ما پروژه‌هایمان را به API آهنگ ها متصل کردیم. در این بخش ما نحوه‌ی عملی کارکردن با یک API و اتصال به سرور را اموختیم.

به کمک آموزشی که در بخش های قبلی داشتیم توانستیم اطلاعات کاربران اهنگ ها و اطلاعات دیگر وب‌سایت را در سرور ثبت کنیم و زمانی که کاربران می‌خواهند وارد وب‌سایت شوند آن اطلاعات را به نمایش بگذاریم.

فاز ششم

در این فاز ما برای بخش های مختلف پروژه تست نوشتیم.

فاز هفتم

در این فاز، کدهایی که تا این مرحله نوشته بودیم را به کمک فریم‌ورک انگولار بازنویسی کردیم. ما سعی کردیم تا حد امکان از امکانات انگولار استفاده کنیم و از ساختار جاوااسکریپتی که تا به این لحظه داشتیم فاصله بگیریم.

4-2 پروژه‌ی نرم‌افزار ETL

در این پروژه وب‌سایتی مشابه سایت Talend طراحی کردیم که شامل دو مفهوم اصلی است:

- پردازش (Processor)
- سناریو (Pipeline)

شرح پروژه

روزانه مقادیری داده در قالب جدول‌های SQL و یا فایل‌های CSV به دست ما می‌رسد که نیاز است روی آن‌ها پردازش‌هایی انجام شود و نتیجه‌ی به دست آمده در جداول دیگر و یا فایل‌های CSV ذخیره شود.

جداول یا فایل‌های ورودی و خروجی لزوماً فرمت و ستون‌های یکسانی ندارند و بر حسب نیاز ممکن است ستون‌های متفاوتی داشته باشند.

کاربری که از نرم‌افزار استفاده می‌کند دانش کدنویسی ندارد و صرفاً با مفاهیم جدول و ستون و پردازش‌ها آشناست. کاربر باید بتواند سناریوی دلخواه خود را با استفاده از رابط کاربری ارائه‌شده طراحی کند و آن را با نام دلخواه خود ذخیره کند تا در صورت لزوم بعداً بتواند آن‌ها را مشاهده و یا ویرایش نماید. هم‌چنین باید بتواند طراحی خود را در قالب فایل YML خروجی بگیرد و یا فایل YML خود را به عنوان سناریو در نرم‌افزار بارگذاری کند.

کاربر باید بتواند در هنگام اجرا وضعیت اجرای هر یک از پردازش‌ها رامشاهده کند این که اجرای پردازش تمام شده یا هنوز در حال اجراست یا با خطا مواجه شده یا اصلاً هنوز دستور اجرایی نیامده و یا این که توسط کاربر کنسل شده است.

فصل سوم
غیر همزمانی‌ها در جاوا اسکریپت

3-1غیر همزمانی چه زمانی اتفاق می‌افتد؟

ممکن است زمانی پیش بیاید که در برنامه‌تان بخواهید در بخشی با استفاده از تابع setTimeout، مقداری صبر کنید و بعد از آن برنامه را ادامه دهید یا در مثال پروژه‌ی اسپاتیفای، شما لازم دارید که در ابتدای لود صفحه، اطلاعات آهنگ‌ها و خواننده‌ها را از APIدریافت کنید و تا زمانی که این اطلاعات از APIگرفته نشده‌اند، برنامه منتظر دریافت دیتا بماند.

کد زیر را در نظر بگیرید:

```
let text = 'text one;'
let setDisplayTimeOut = () => {
  setTimeout(() => {
      text = 'text two';
  }, 1000);
  let output = document.getElementById("output");
  output.innerText = text;
}
```

شکل (3-1)

طبق این کد، می‌خواهیم بعد از 1 ثانیه مقدار متغیر textعوض شود و text twoدر صفحه نمایش داده شود. اما در حالت عادی، جاوا اسکریپت مقدار نمایش داده شده را همانtext one نشان می‌دهد. فرض کنید می‌خواهیم از سرور اطلاعات کاربران را دریافت کنیم و بعد از دریافت، تغییراتی را روی آن‌ها انجام بدهیم. اگر مانند مثال بالا، اطلاعات را دریافت کرده و تغییرات را انجام بدهیم، دچار خطا می‌شویم. زیرا روند دریافت اطلاعات کاربران کامل نشده. برنامه

باید صبر کند تا دریافت اطلاعات کامل شود و بعد تغییرات را ایجاد کند.

در این شرایط است که Promiseها به کار ما می‌آیند!

Promise 1-1-3

Promiseها برای عملکردهای asynchronous مانند مثال‌های بالا به کار ما می‌آیند. با promise ما منتظر نتیجه می‌مانیم و با then() مشخص می‌کنیم که بعد از آن چه اتفاقی رخ بدهد. مثال بالا را با Promiseبازنویسی می‌کنیم:

```
let text = 'text one;'
const timePromise = new Promise((resolve, reject) => {
  setTimeout(() => {
      text = 'text two';
      if (text === 'text two')
          resolve();
      else
          reject('wrong data!');
  }, 5000);

});

const setText = () => {
  let output = document.getElementById("output");
  output.innerText = text;
};

function setDisplayTimeOut() {
  timePromise.then(setText)
  .catch((error) => {
      console.log(error);
   });
}
```

شکل (3-2)

در Promise، می‌توانید resolve()و reject()را بسته به موفقیت‌آمیز بودن و یا نبودن Promiseصدا بزنید. در این تابع، در صورتی که مقدار text، برابر با text twoباشد، Promiseموفقیت‌آمیز بوده است و باید resolve()صدا شود و در غیر این صورت، موفقیت‌آمیز نبوده و reject()با پیغام مناسب باید صدا زده شود. تابع then()، دو تابع ورودی می‌گیرد اولی هندل‌کننده‌ی موفقیت‌آمیز بودن Promiseو دومی هندل‌کننده‌ی موفقیت‌آمیز نبودن Promiseاست. به جای تابع دوم از .catch()هم می‌توان استفاده کرد.

async, await 2-1-3

در صورتی که پشت نام یک تابع async به کار رود ، آن تابع یک Promise به عنوان خروجی برمی‌گرداند.

async و await یک راه تمیزتر و ساده‌تر برای استفاده از Promiseها هستند.

مسئله بالا را با استفاده از async و await حل می‌کنیم:

```
let text = 'text one;'
async function getTimePromise() {
 return new Promise((resolve) => {
     setTimeout(() => {
         text = 'text two';
         resolve();
     }, 5000);
   });
 }

async function setText() {
 let output = document.getElementById("output");
 await getTimePromise();
 output.innerText = text;
}
```

شکل (3-3)

ساختار بالا بسیار ساده است. با فراخوانی تابع setText()، منتظر می‌شویم تا نتیجه‌ی getTimePromise() برگردانده شود. بعد از آن که نتیجه موفقیت‌آمیز بود، مقدار خروجی را ست می‌کنیم. توجه داشته باشید در صورتی که در تابعی از await استفاده می‌شود، باید خود آن تابع هم async تعریف شود. مانند تابع setText که خودش async تعریف شده است.

Fetch 3-1-3

fetch Api در جاوا اسکریپت برای ارسال درخواست به سمت سرور و لود کردن اطلاعات و جواب‌های سرور به کار می‌آید.

به کد زیر که برای گرفتن اطلاعات کاربران نوشته شده توجه کنید:

```
async function getData() {
  let response = await fetch('https://jsonplaceholder.typicode.com/users');
  if (response.ok) {
      let json = await response.json();
      let users = JSON.stringify(json);
      let output = document.getElementById('output');
      output.innerText = users;
  } else if (response.status == 500) {
      console.log("server error");
  }
}
```

شکل (3-4)

در حالت کلی ورودی‌های تابع url، fetchو option هستند و خروجی آن یک Promise است.

option پارامترهای دلخواهی است که در ارسال درخواست به سرور می‌توانیم در نظر بگیریم. به طور مثال تغییر متد ارسال، header و غیره. در صورتی که option به تابع پاس داده نشود، متد GET ساده در نظر گرفته می‌شود. در خط دوم برنامه‌ی بالا، با صدا زدن تابع fetch، منتظر نتیجه آن می‌مانیم و بعد از دریافت کامل اطلاعات، نتیجه در response ریخته می‌شود. با استفاده از response می‌توان نتیجه‌ی دریافت اطلاعات را مشاهده کرد. اگر response.ok=true باشد، یعنی دریافت اطلاعات درستی انجام شده . response.status همان Status code این عملیات است.

در صورتی که Status code برابر با 500 باشد، یعنی خطایی سمت سرور رخ داده .