**پروژه تیلویند FrontHooks**

» از zeplin.io میتوانیم پروژه فیگما را باز کرده و سورس css و اندازه های دقیق را با هاور ماوس به ما می دهد.

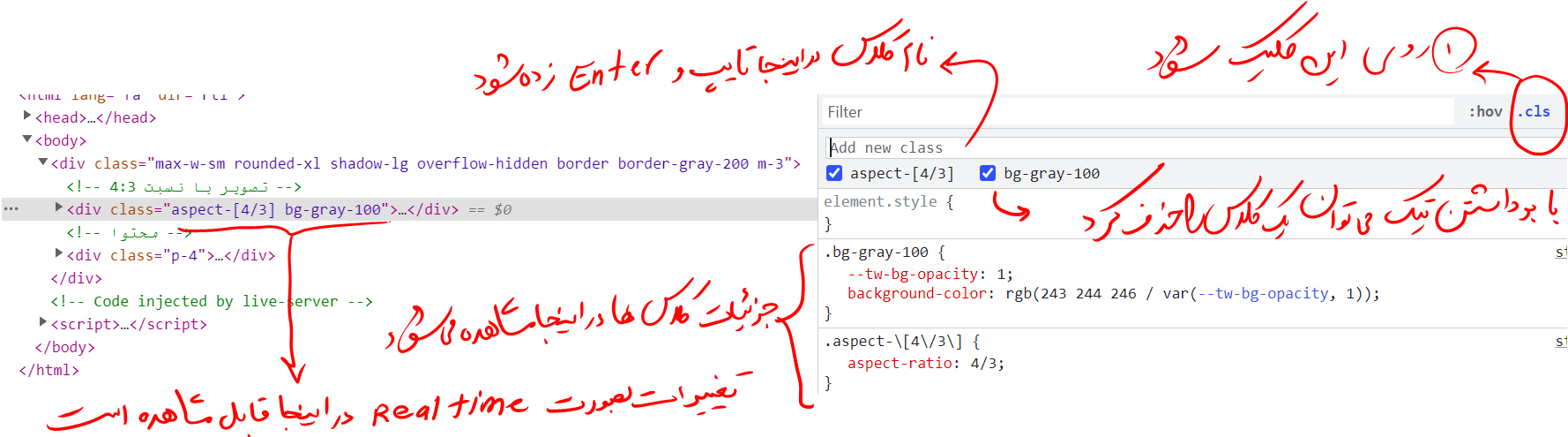
» پس از نصب تیلویند، اولین کارمان اضافه کردن فونت پروژه است. یک پوشه fonts در پوشه assets ایجاد میکنیم و فونت های پروژه را در این پوشه قرار میدهیم.

» در سند اولیه، قرار دادن خاصیت های lang="fa" و dir="rtl" فراموش نشود.

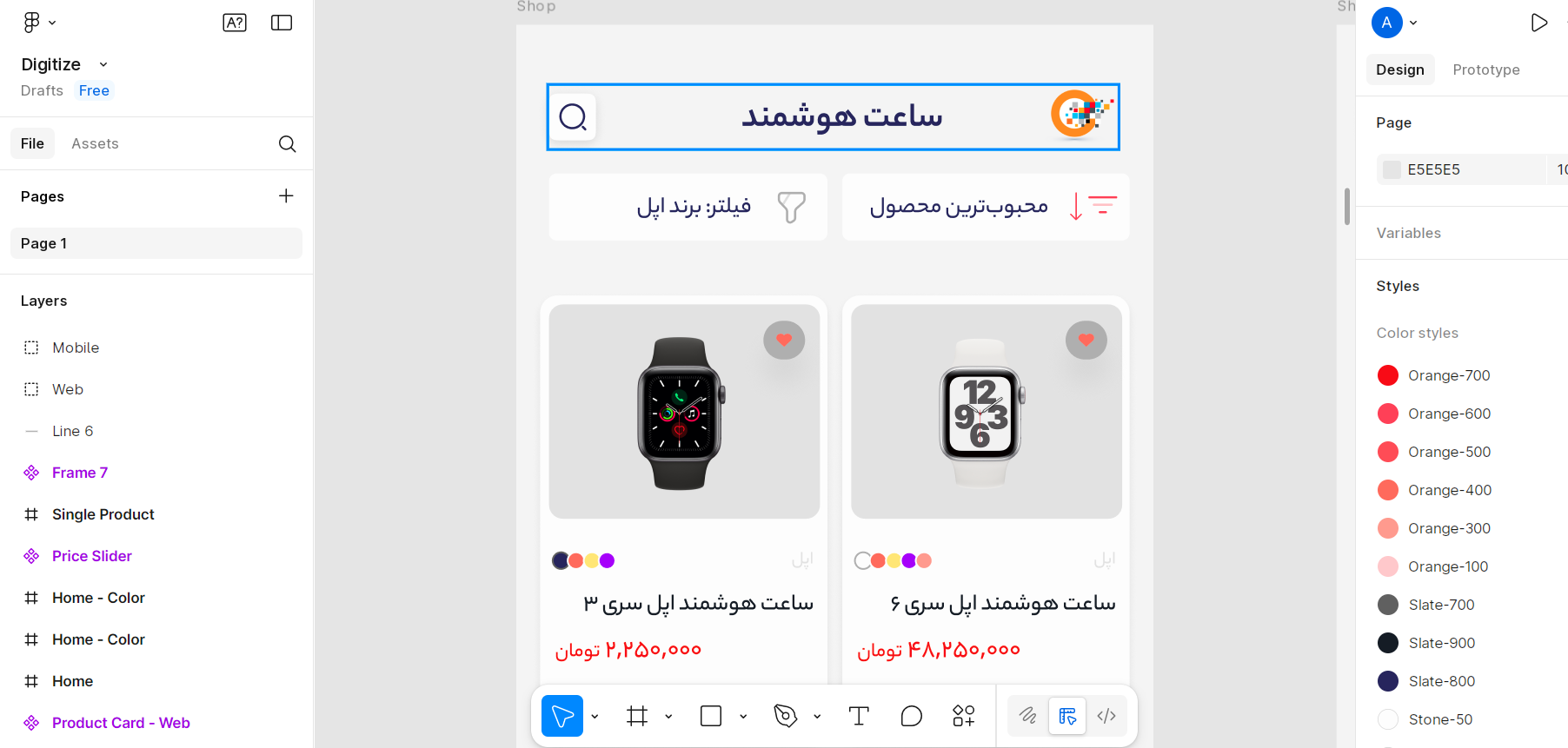


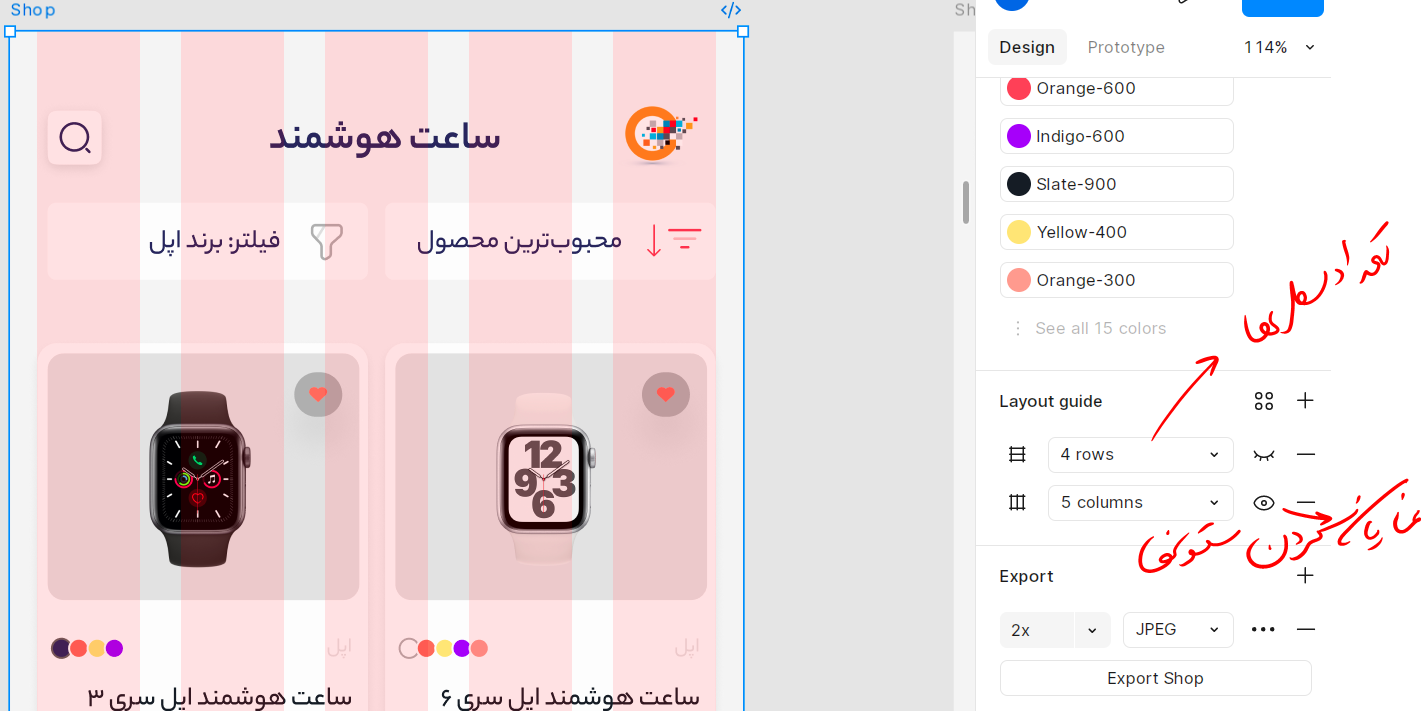
|  |  |
| --- | --- |
|  |  |

» یک پوشه هم به نام images در پوشه public ایجاد میکنیم و عکس های پروژه را در این پوشه قرار می دهیم.



» طراحی را از صفحه Shop مربوط به حالت موبایل (mobile first approach) شروع میکنیم:

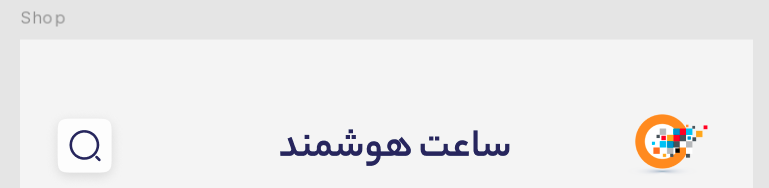


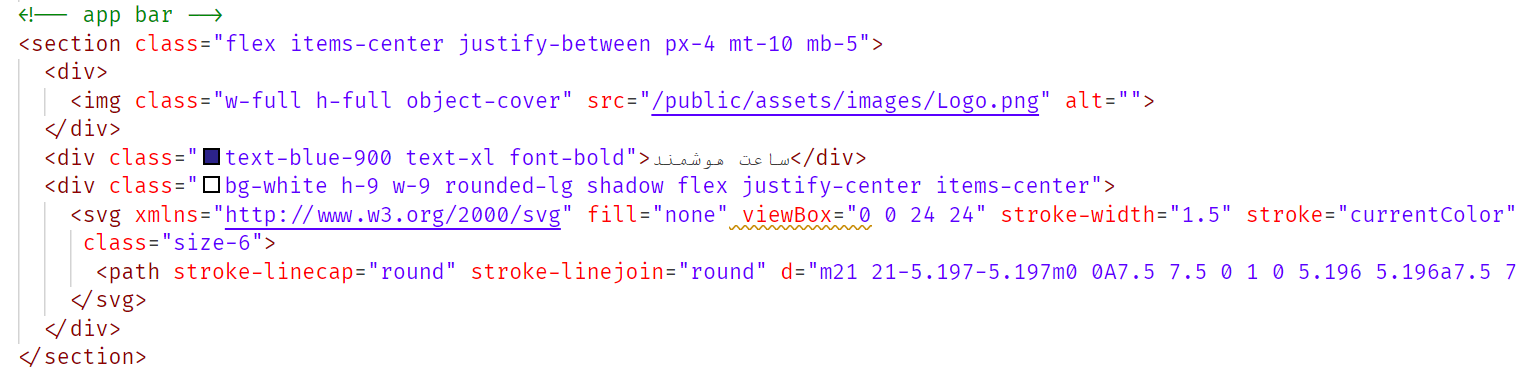


» تنظیم فونت و رنگ

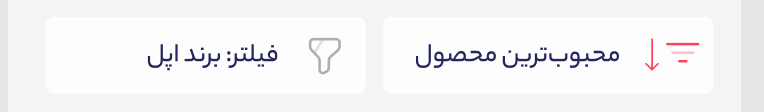


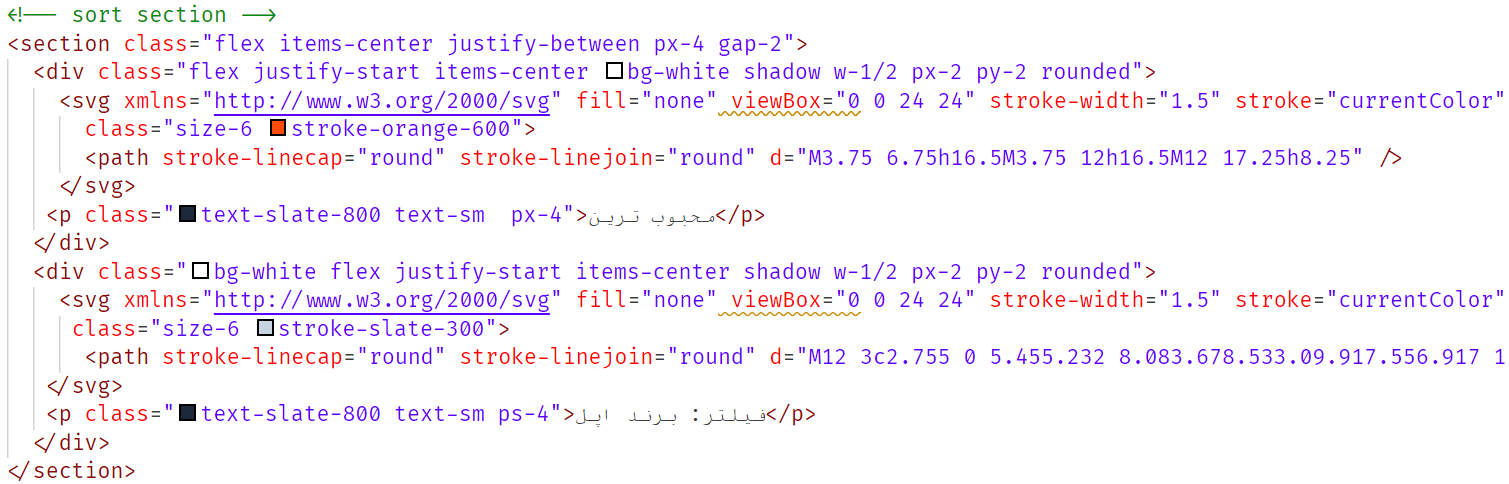
طراحی قسمت :





طراحی قسمت :







» از w-1/2 جهت عرض دو تا باکس فوق استفاه کردیم. و از stroke-orange-600 رنگ آیکون سمت راستی را تعیین کردیم.

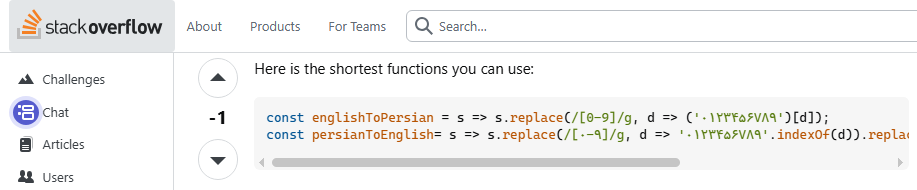
پیاده سازی بخش محصولات:

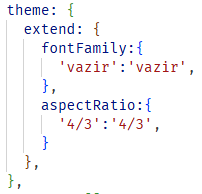
|  |  |
| --- | --- |
| در حالت دسکتاپ | در حالت موبایل |
|  |  |

اشتباهی که بعضی انجام میدهند این است که برای کارت ها عرض ثابت در مدیاکوئری های مختلف در نظر میگیرند که اشتباه است و باید با grid هندل شود.

» برای نمایش اعداد فارسی سرچ کنید: persian number javascript

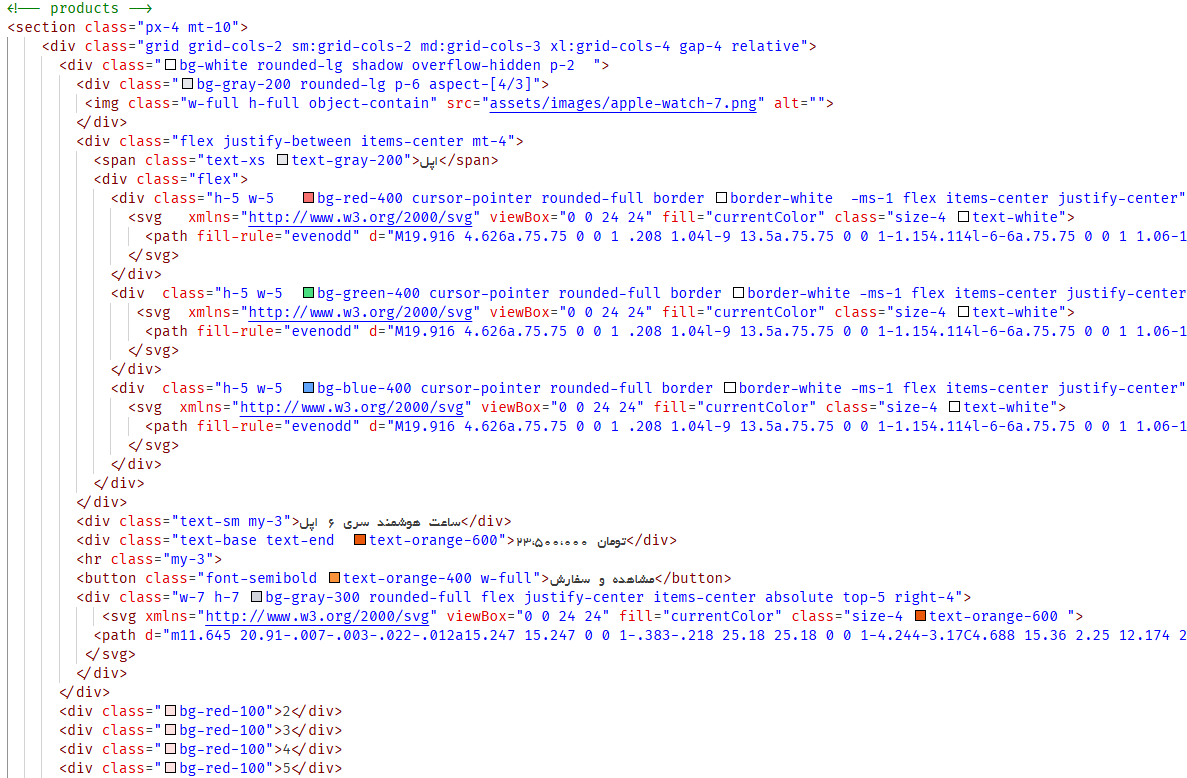
سپس اعداد فارسی را از تصویر زیر کپی کرده و در پروژه استفاده کنید. بعداً باید توسط یک تابع جاوا اسکریپتی اعداد را به فارسی تبدیل کرد:





» نسبت 4/3 را در فایل config قرار میدهیم چون میخواهیم عکس محصولات با این نسبت

در سایت مشاهده شوند.



برای احترام به div تشکیل دهنده ساختار، رنگ سفید را به div داخلی تر میدهیم.

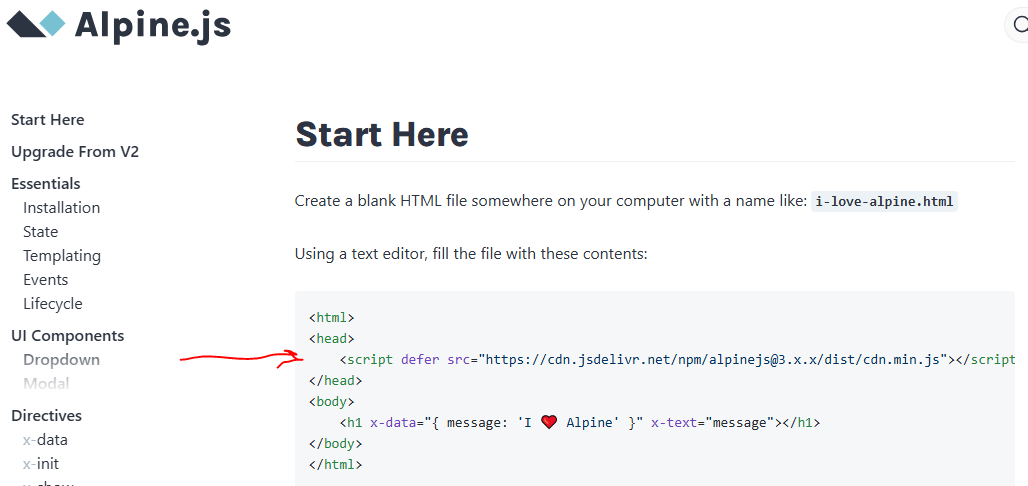
» میخواهیم با کلیک روی رنگ محصول، یک تیک سفید رنگ روی رنگ انتخاب شده و یک ring هم دور رنگ انتخاب شده نمایش داده شود:



برای داینامیک سازی های کدهای قبل بجای استفاده از جاوا اسکریپت و جی کوئری از کتابخانه alpine.js استفاده میکنیم:

1.ابتدا وارد سایت alpine.js شوید و روی لینک docs کلیک کنید و cdn این کتابخانه را برداشته و قبل از تگ بسته شدن body قرار دهید:







2.برای والد هرسه تا div مربوط به رنگها، متغیر color را با مقدار اولیه red تعریف میکنیم:



3.برای هر div مربوط به هر رنگ، رویداد کلیک را تعریف میکنیم که مقدار متغیر color را با هر کلیک برابر همان رنگ مربوطه خودش قرار میدهد:

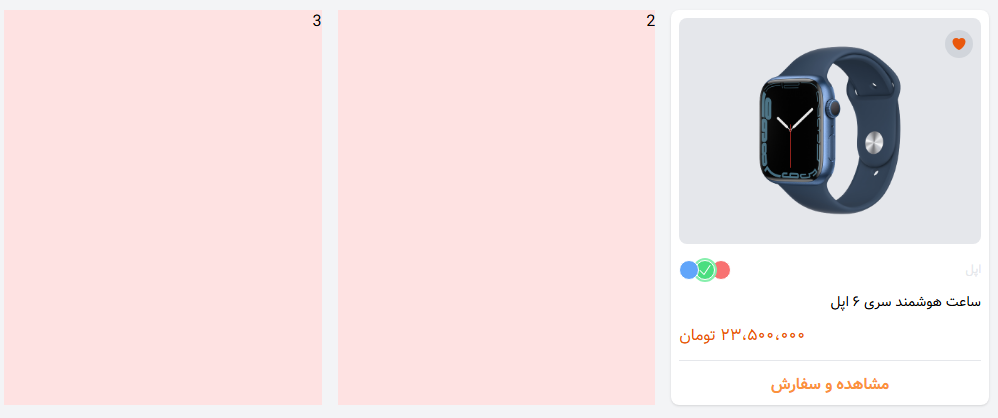
|  |  |  |
| --- | --- | --- |
| برای رنگ سوم | برای رنگ دوم | برای رنگ اول |
|  |  |  |

4.برای هر svg آیکونها شرطی قرار میدهیم که درصورتی این آیکون svg نمایش داده شود که متغیر color برابر رنگ مربوطه باشد.

|  |  |  |
| --- | --- | --- |
| برای رنگ سوم | برای رنگ دوم | برای رنگ اول |
|  |  |  |

5.برای هر div مربوط به هر رنگ، یک کلاس آلپاینی تعریف میکنیم که این کلاس شامل یک شرط است که در صورتی که شرطش برقرار باشد این کلاس اعمال می شود که در این مثال شرط این است که اگر برای هر div ، متغیر color برابر رنگ انتخاب شده باشد آگاه ring به div اضافه می شود در غیر اینصورت هیچ کار خاصی نمی کند ( یک کوتیشن خالی بعد از علامت کالن)

|  |  |
| --- | --- |
|  | برای رنگ اول |
|  | برای رنگ دوم |
|  | برای رنگ سوم |



» حال بجای اینکه تمام کدهای قبلی را برای مثلاً 10 محصول تکرار کنیم میتوانیم اینکار را بطور داینامیک انجام دهیم و می توانیم از loop زدن در Alpine.js استفاده کنیم. دستور x-for در Alpine.js برای تکرار یک بلوک از کد HTML به ازای هر آیتم در یک آرایه یا شیء استفاده می‌شود. برای استفاده از x-for، یک عنصر (معمولاً template ) را در نظر بگیرید و دستور x-for را روی آن اعمال کنید. در داخل x-for، یک متغیر برای دسترسی به آیتم فعلی در حلقه تعریف می‌کنید.

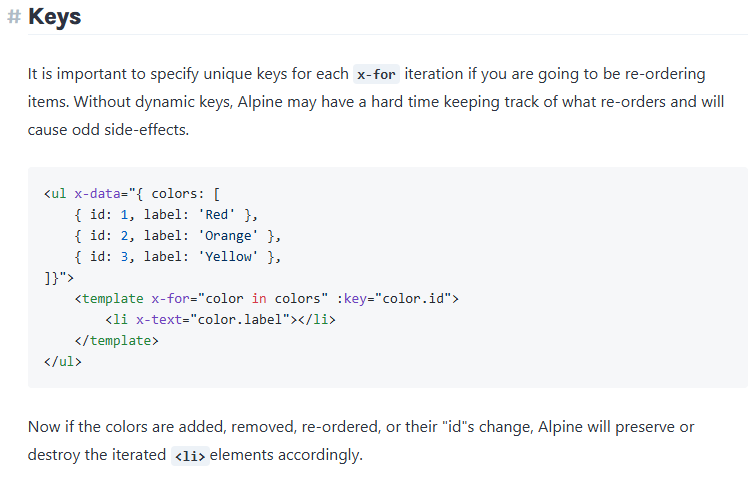
<template x-for="item in items" :key="item">

<div x-text="item"></div>

</template>

برای نمایش یک متن درونی از x-text استفاده میکنیم. مثال: <span x-text="title"></span>





در مثال فوق دیتای ما یک آبجکت است و مقداری که این آبجکت دارد color نام دارد و color هم یک آرایه ای از آبجکت هاست. در تمرین ما هم هر محصول به عنوان یک آبجکت از این آرایه تعریف می شود. البته این دیتا بعداً از دیتابیس گرفته و جایگزین می شود .

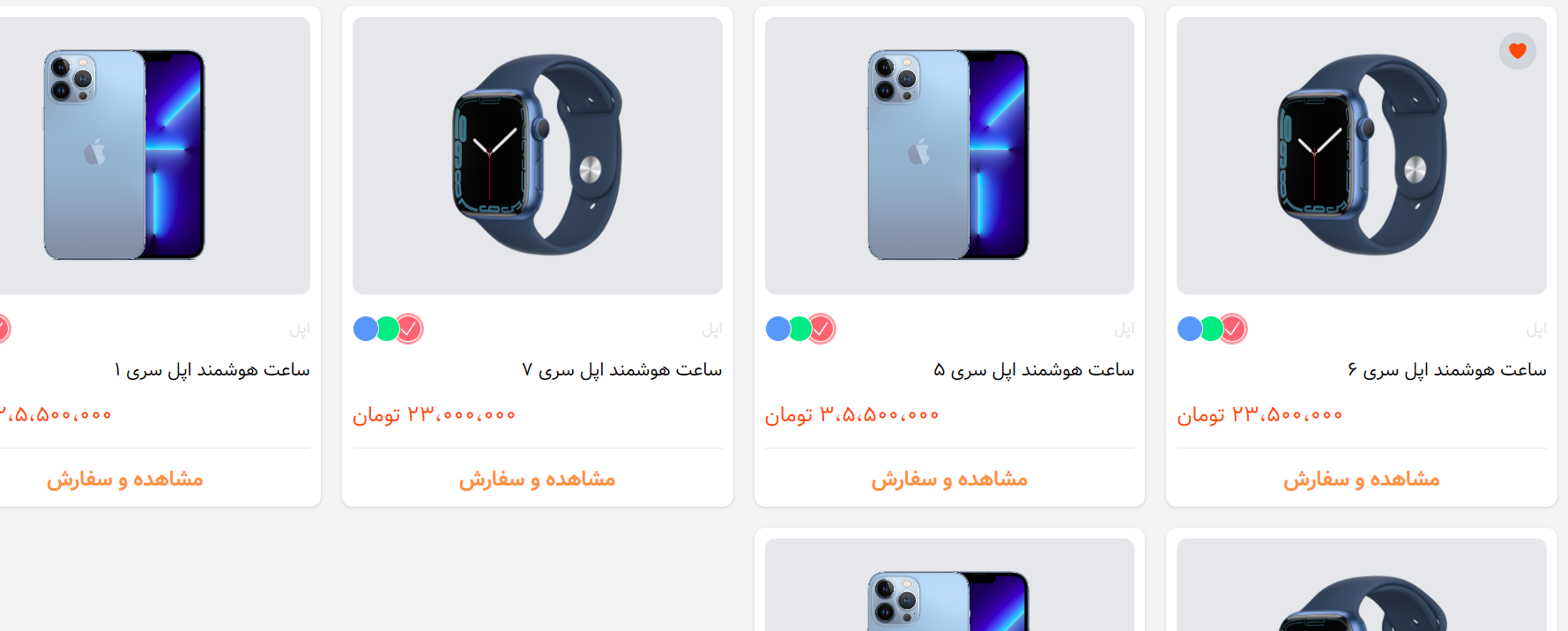
» عبارت x-data در مثال فوق را کپی کرده و به Div مربوط به والد div کارت محصولات یعنی همان div گرید بندی اضافه میکنیم و div های 2 تا 10 را پاک میکنیم یعنی فقط div شماره 1 که درمرحله قبل طراحی کردیم را نگه می داریم.



» اگر در کد فوق دقت کنید در خط 82 و 83 عناوین محصول و قیمت آن ها حذف شده است و از عبارت های x-text="product.title" و x-text="product.price" برای خواندن این مقادیر و نمایش در خروجی استفاه شده.

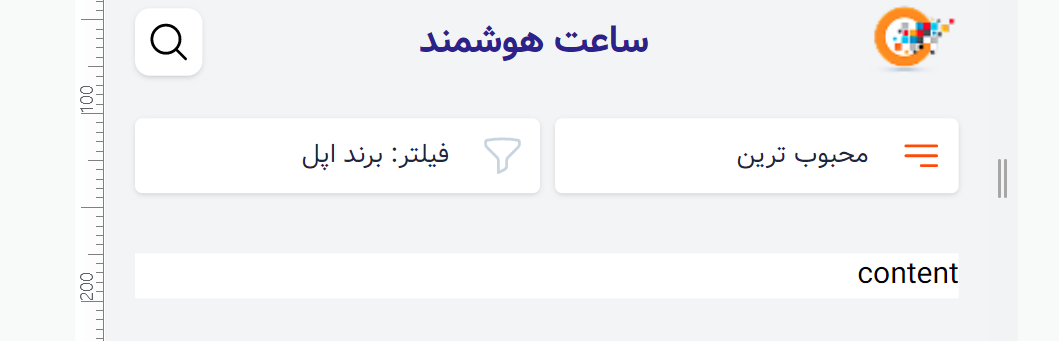
اگر بخواهیم تصاویر هم به همین نحو داینامیک خوانده شود باید کد به شکل زیر اصلاح شود. در این حالت نمی توان روی تگ img از x-text استفاده کرد. x-text متن داخلی عنصر را عوض می‌کند، اما src را تنظیم نمی‌کند. برای bind صحیح از **src="product.img":** یا x-bind:src="product.img" استفاده کنید:

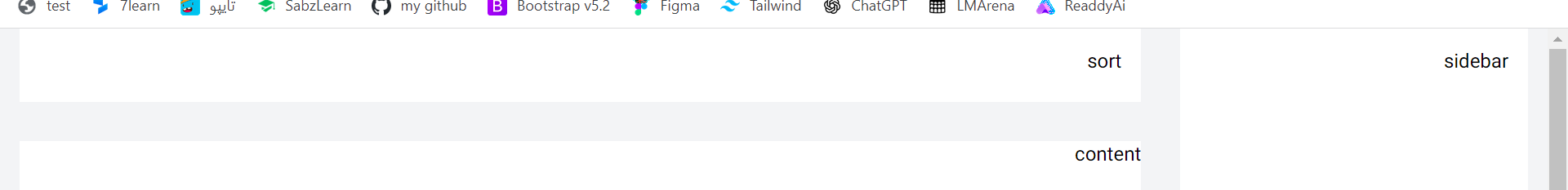
  
خروجی نهایی به شکل زیر خواهد بود:



طراحی ساختار اصلی:

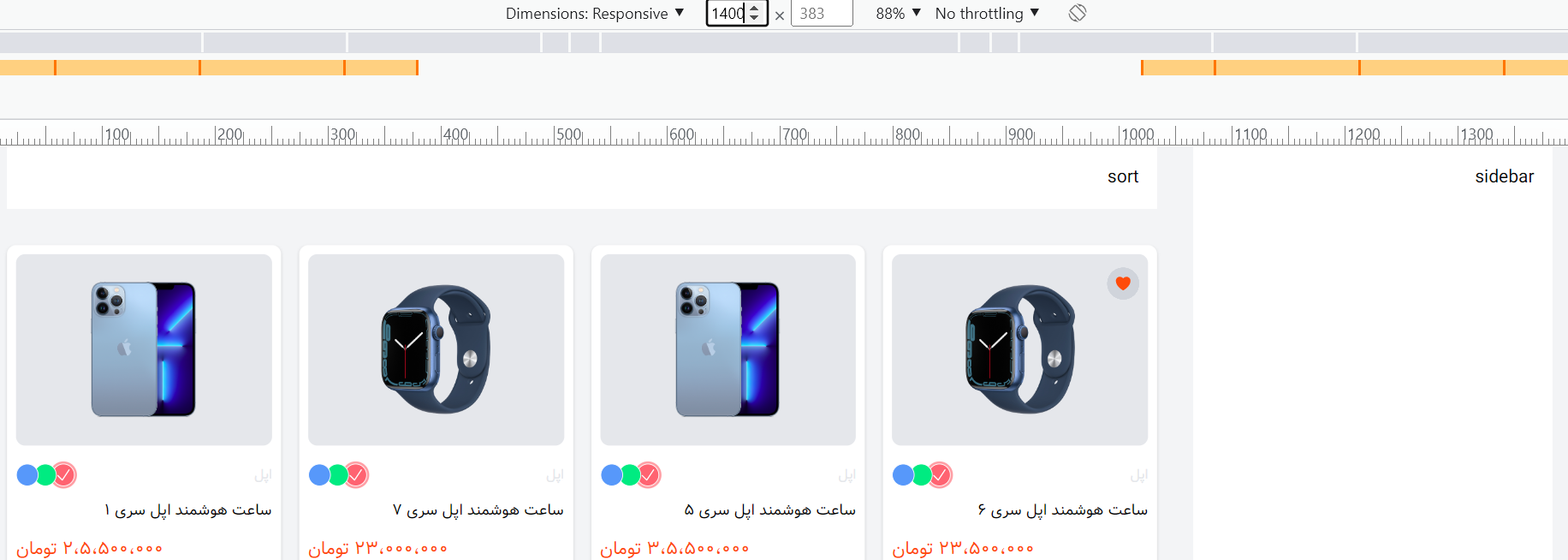






جایگزینی محتوای کارت محصولات طراحی شده در قسمت های قبل با عبارت content :

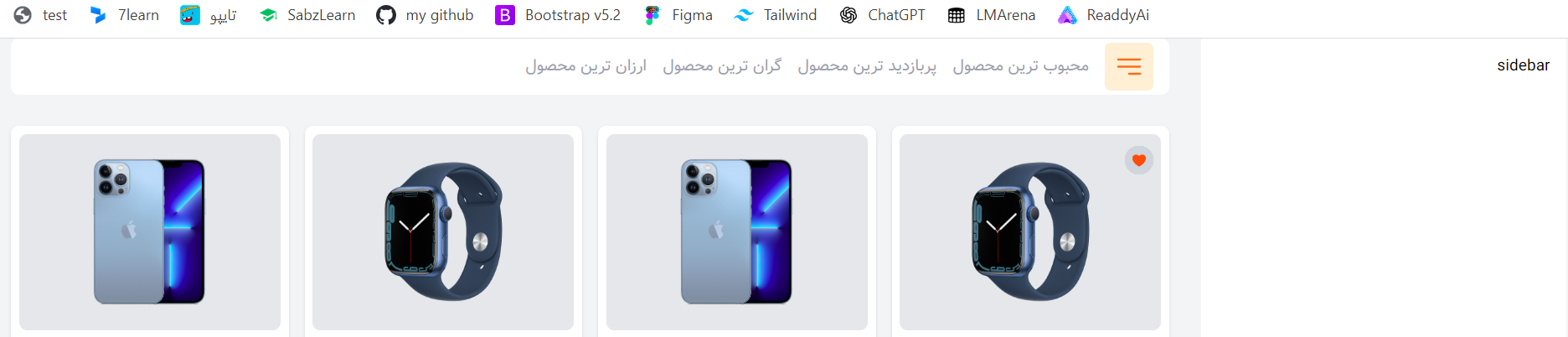


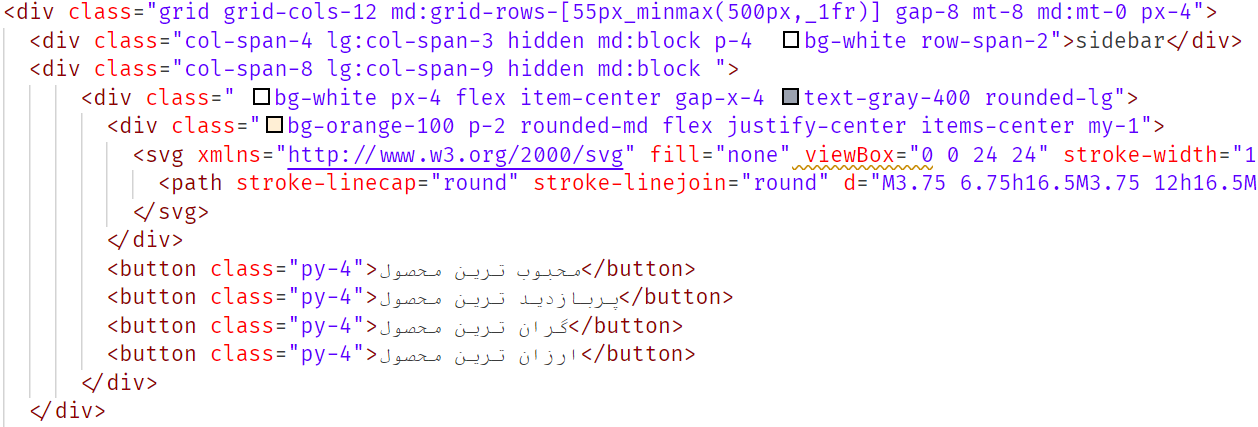
یک 

پیاده سازی قسمت بالای کارت محصولات:

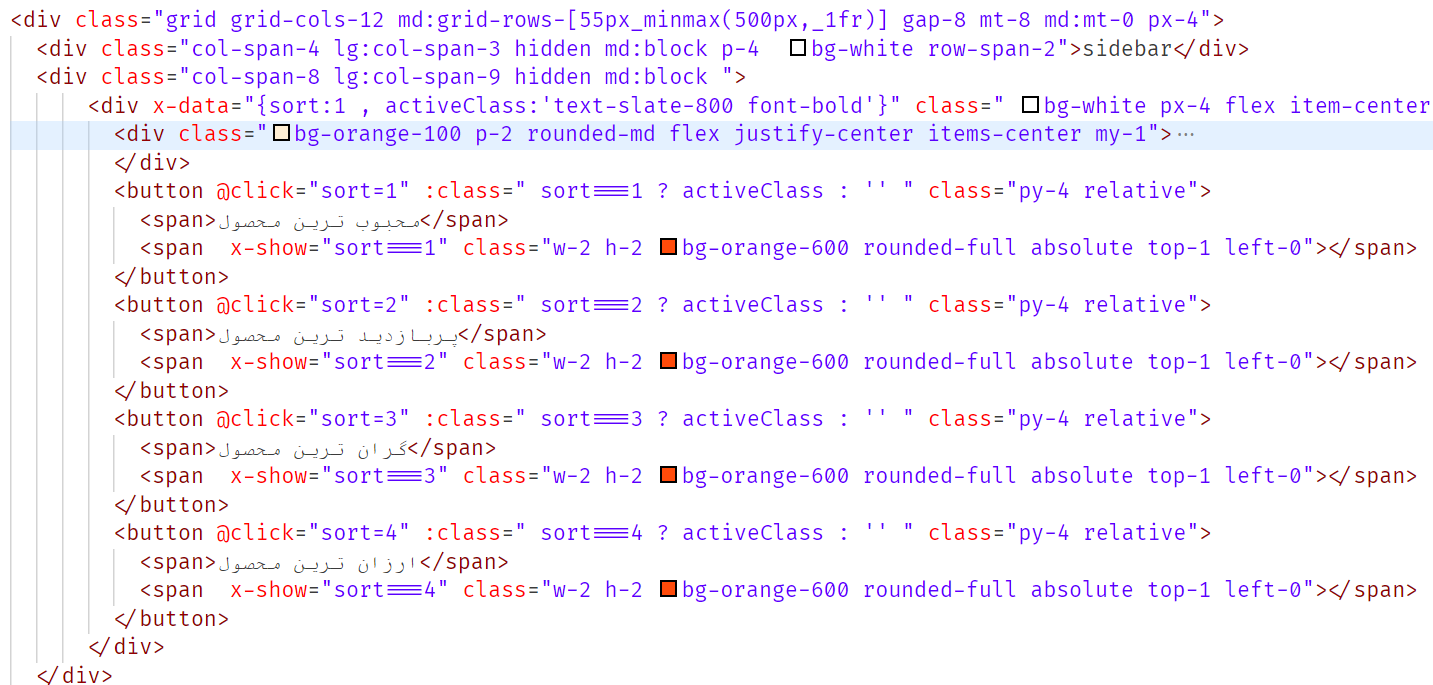
از button ها برای پیاده سازی استفاه میکنیم.و بعد توسط alpine.js حالت تب مانند را ایجاد میکنیم.

برای احترام به div تشکیل دهنده ساختار، یک div داخلی اضافه می کنیم و رنگ سفید را به این div میدهیم و پدینگ py-4 را به button ها میدهیم تا فضای قابل کلیک بهتر و بیشتری در اختیار داشته باشیم.





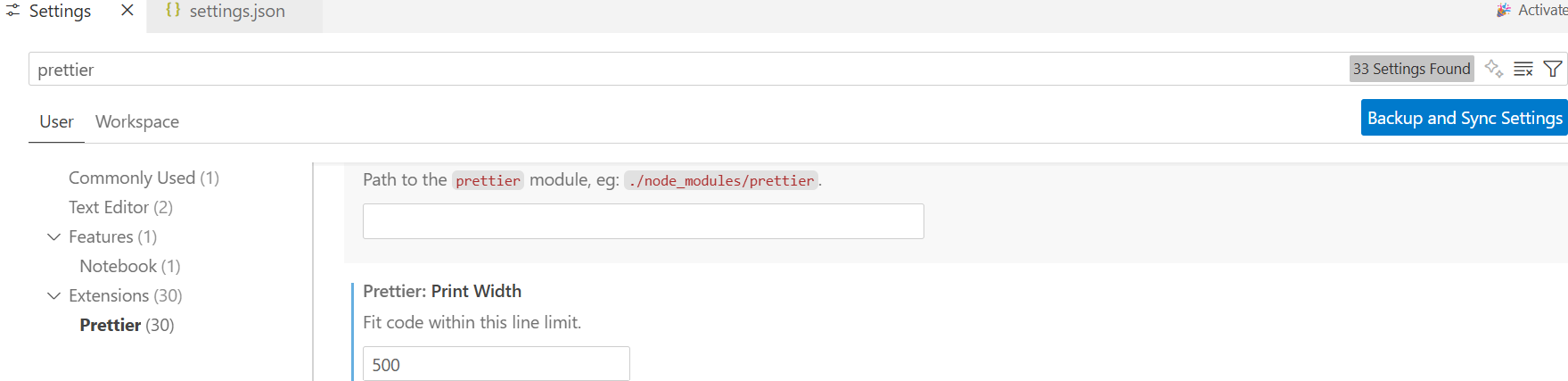
میخواهیم با کلیک روی هر button رنگ آن پر رنگ و فونت آن bold شود همچنین یک دایره کوچک نارنجی رنگ هم در گوشه بالای سمت چپ هر button نمایان شود برای اینکار به والد button ها که bg-white دادیم عبارت x-data را اختصاص دهیم:

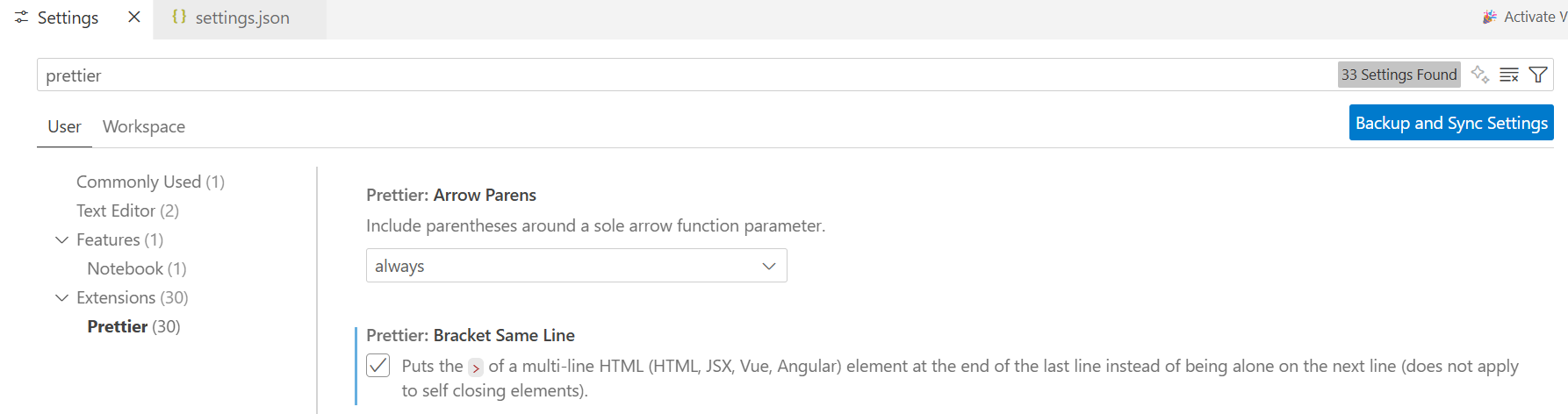




در تگ والد ، در x-data مقدار متغیر دلخواه با نام Sort را برابر 1 و کلاسهای دلخواه را نیز در متغیر activeClass اختصاص دادیم. با کلیک روی هرbutton توسط رویداد @click مقدار متغیر sort مقدار دهی می شود و توسط کلاس آلپاینی :class چک می شود که اگر شرط برقرار باشد کلاسactiveClass اعمال شود و اگر برقرار نبود هیچ کار خاصی انجام نمی شود ( بعد از علامت کالن ، تک کوتیشن خالی قرار داده شده). x-show هم در صورت برقرار بودن شرطش عملیات نمایان شدن دایره کوچک قرمز را انجام میدهد.

» برای کارکرد بهتر prettier عدد زیر از 80 به 500 تغییر داده شد و تیک گزینه puts the … نیز زده شد.

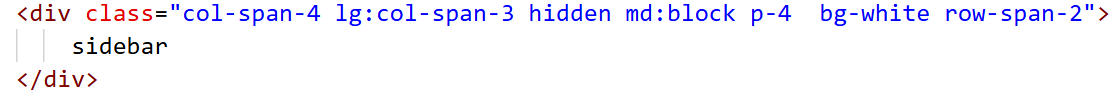


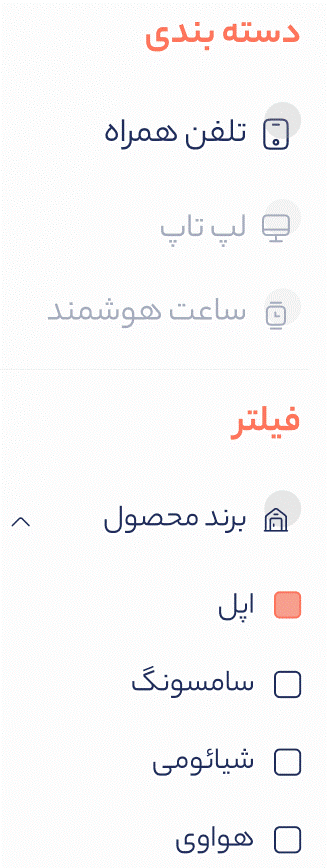


» در طرح فیگمای ما ستون سایدبار تا انتها کشیده نشده و فقط به اندازه محتوای درون خودش ارتفاع گرفته ولی در کدهای ما، سایدبار که پس زمینه سفید دارد به اندازه ارتفاع ردیف محصولات، ارتفاع می گیرد برای حل این قضیه پس زمینه سفید را از div اولیه سایدبار می گیریم و به یک div داخلی جدید می دهیم.

» آیکون ها را بصورت Svg از فایل فیگما استخراج میکنیم، برای اینکار روی آیکون در فیگما کلیک راست کرده و گزینه copy و سپس گزینه copy as svg را انتخاب می کنیم حال برای کاهش حجم این آیکونها وارد سایت svgomg شوید و گزینه paste markup را انتخاب کنید و سپس آیکون را کلیک میکنیم تا فایل کاهش یافته در Cilpboard قرار گیرد و این آیکون را در فایل htmlدرج میکنیم.

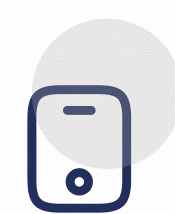


 کد پیش فرض بصورت فوق است که آن را تغییر میدهیم:





چالشی که در کار با این آیکون‌ها داریم این است که این آیکون‌ها دارای پس زمینه خاکستری رنگ هستند که هنگامی که روی این آیکون و متن کنارش هاور شود متن کنار و خود آیکون تغییر رنگ می‌دهد ولی پس زمینه رنگ ثابت دارد. برای این کار از پس زمینه، خروجی svg نمی‌گیریم فقط از آیکون خروجی svg میگیریم و پس زمینه را توسط یک span ایجاد میکنیم.



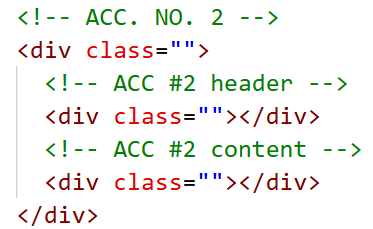
|  |  |
| --- | --- |
|  |  |

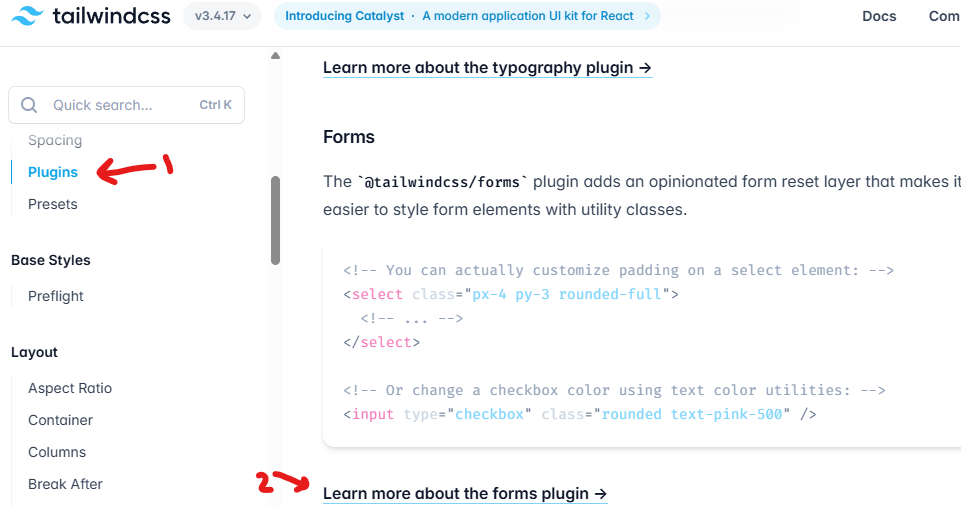
» اگر گزینه paste markup جواب نداد کافی است کد svg از فایل فیگما را درون یک فایل نوت پد paste کرده و سپس آن را با پسوند svg ذخیره کنیم و بعد درون سایت svgmog ایمپورت کنیم .

» برای اینکه آیکون Svg هنگام انتخاب شدن تغییر رنگ بدهد کد رنگی که در قسمت pathدارد را پاک میکنیم و دستور stroke="currentColor" را در تگ Svg آن می نویسیم.

» برای پیاده سازی از ul – li استفاده میکنیم و از تگ a درون li استفاده میکنیم.

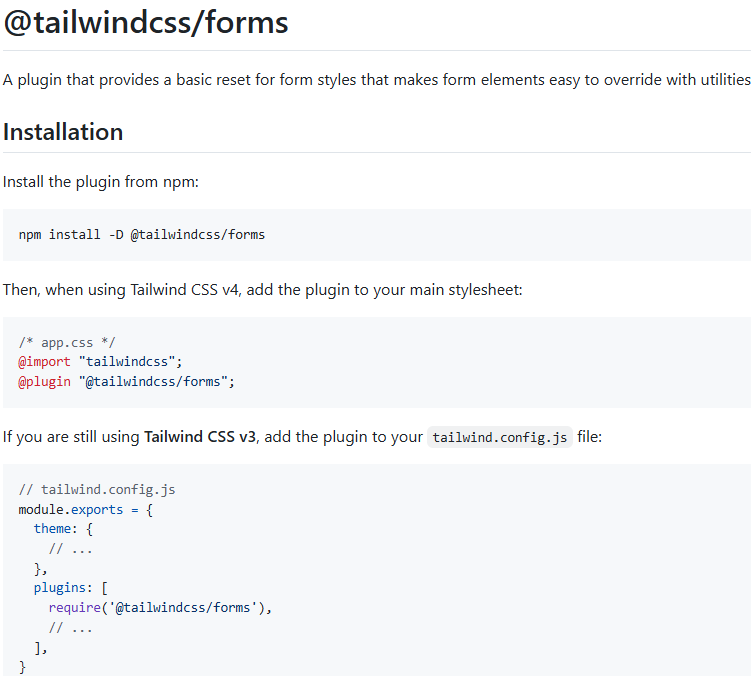
در قسمت فیلتر سه تا div برای پیاده سازی هر سه آکاردئون (برند محصول، رنگ محصول، محدوده قیمت) قرار میدهیم. هرآکاردئون یک هدر دارد و یک content .



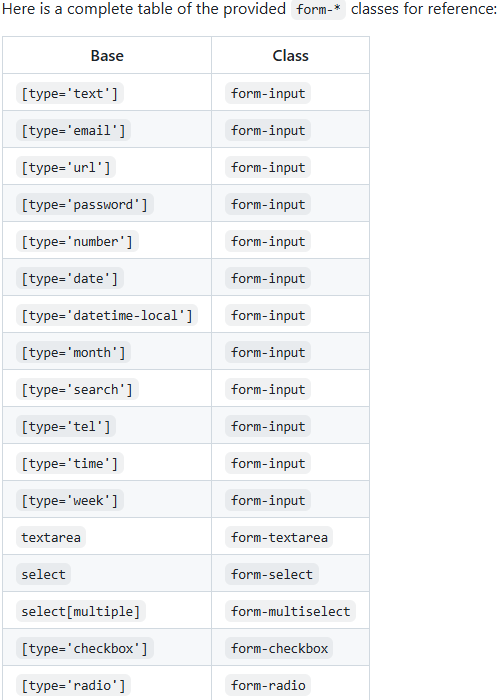


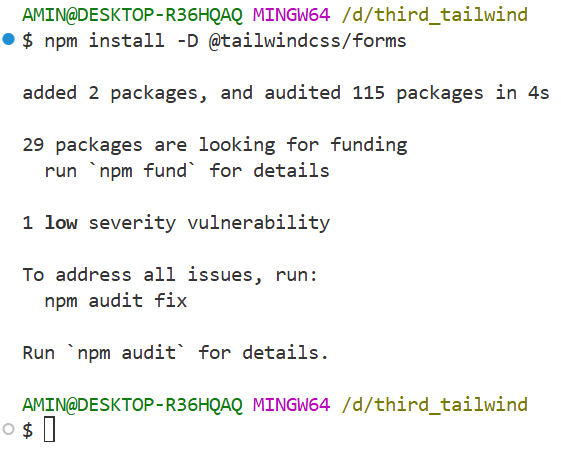
برای زیباتر کردن چک باکس ها از پلاگین های عمومی تیلویند استفاده میکنیم.

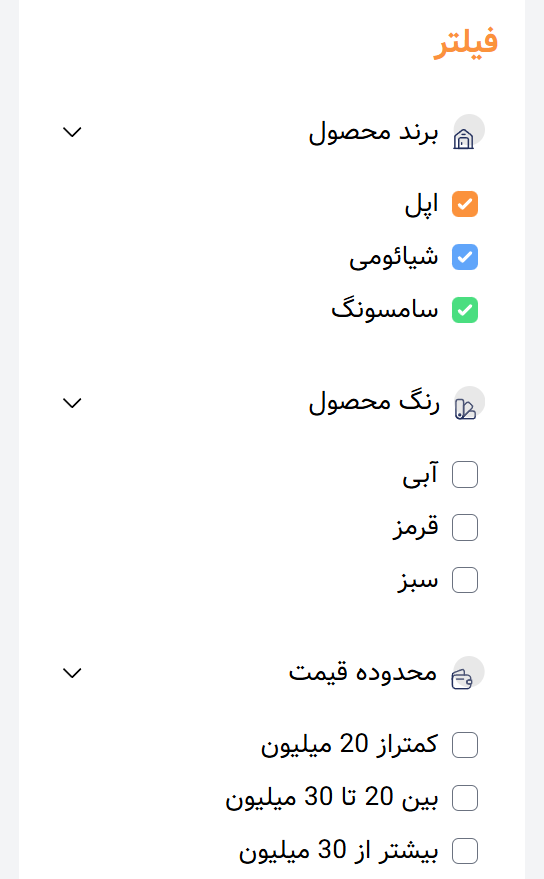
با کلیک روی قسمت 2 به گیت هاب راهنمای نصب منتقل می شویم.

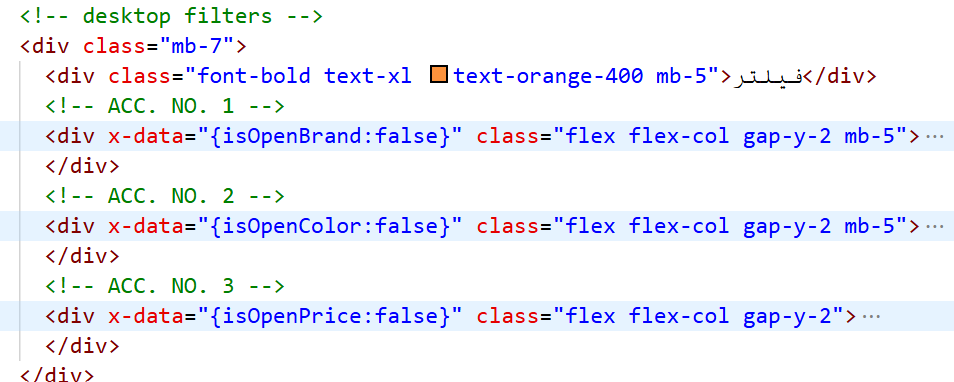


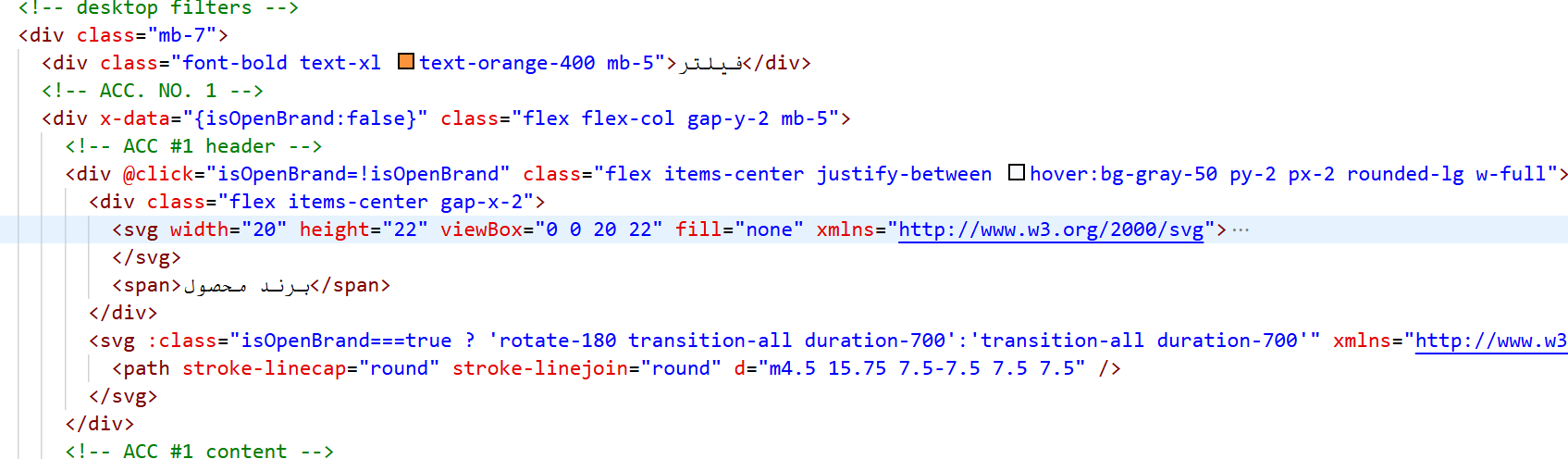
از کلاسهای جدول فوق برای استایل دادن به اینپوت ها استفاده می کنیم.





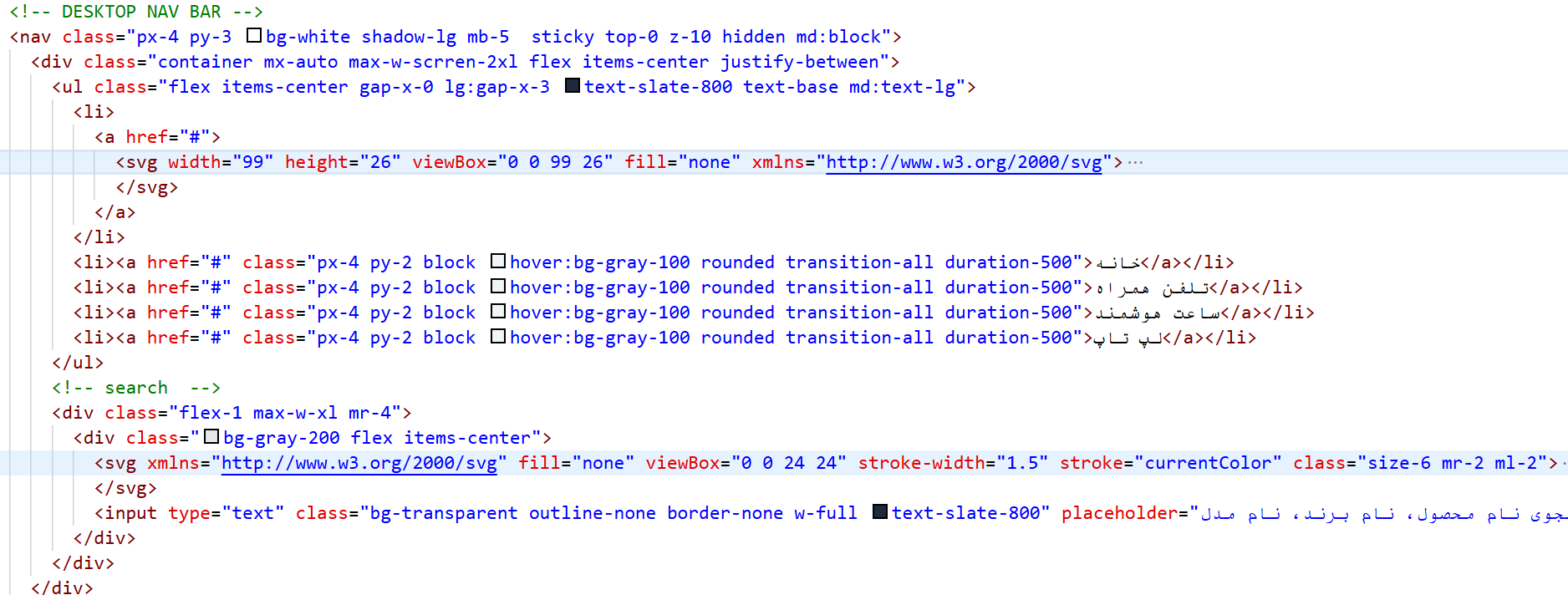




» در کدهای فوق از alpine.js برای باز و بسته شدن آکاردئون استفاده می کنیم.

» در کدهای فوق در صورت برقرار بودن شرط isOpenBrand===true علامت فلش با انیمیشن نرمی 180 درجه میچرخد.

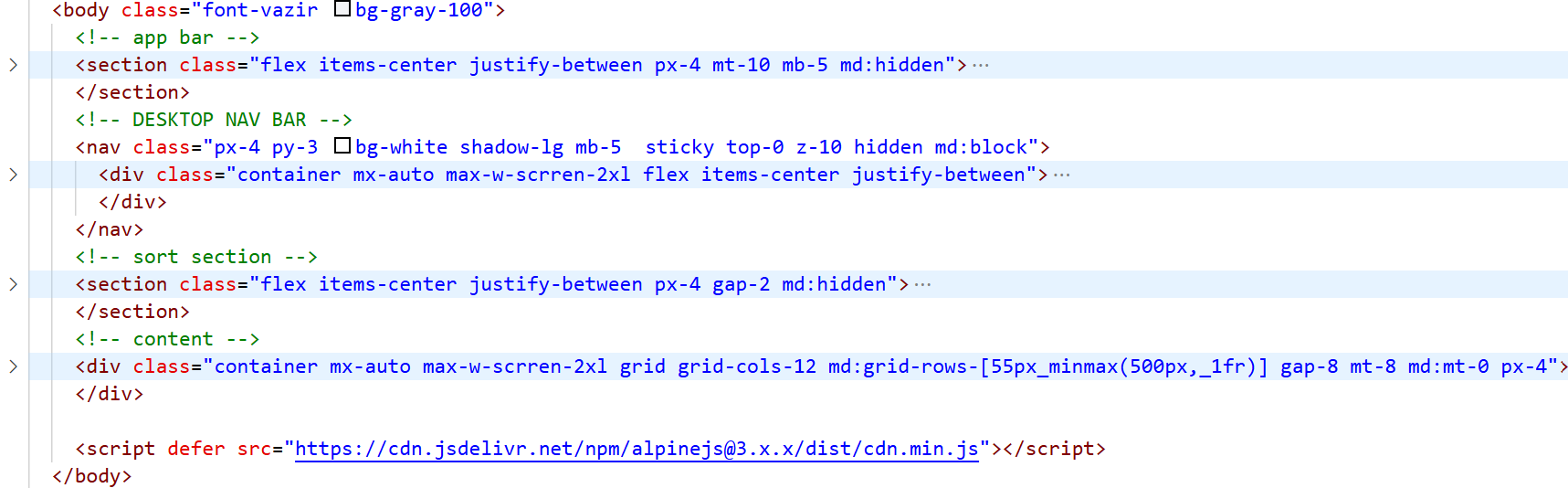
**پیاده سازی منو و سرچ:**

****

|  |  |
| --- | --- |
|  | <div class="flex"> |
|  | <div class="flex-1"> |

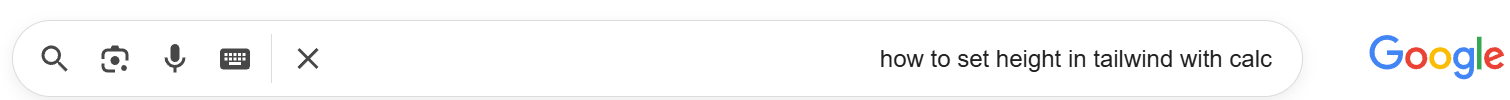
دادن flex-1 باعث میشود فرزندش shrink or grow شود.

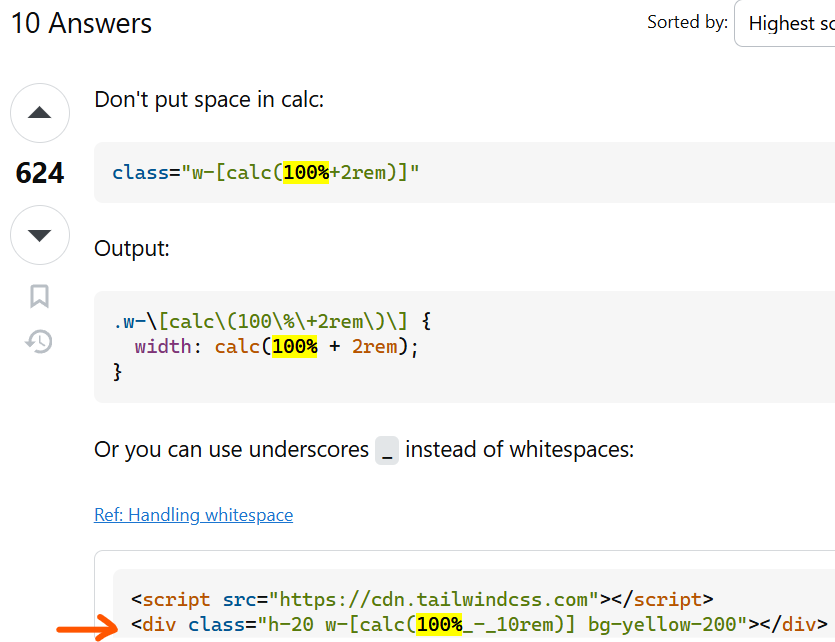
از کلاس های container mx-auto max-w-scrren-2xlدر قسمت منو و content استفاده میکنیم تا منو و محتوای سایت در سایزهای بزرگ کش نیاید.



» اگر از max-w-2xl بجای max-w-scrren-2xl استفاده کنیم، نتیجه دلخواه حاصل نمی شود.

» اگر در قسمت سایدبار، کاربر آکاردئون ها را باز کند، کاربر مجبور است اسکرول کند لذا میخواهیم یک ماکسیمم ارتفاع به این قسمت اعمال کنیم. به نحوی که این مقدار تفاضل ارتفاع viewport از مقدار ارتفاع نوار منو باشد. همچنین میخواهیم سایدبار بصورت چسبان در سمت راست باقی بماند.





در پروژه ما باید دستور زیر را

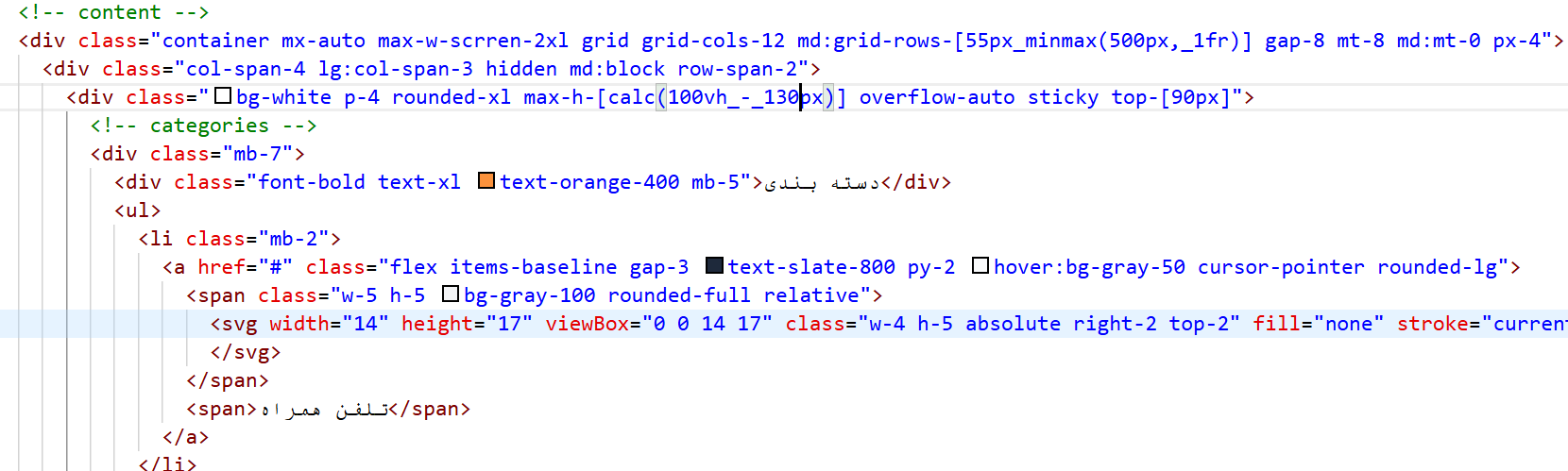
w-[calc(100%\_-\_10rem)]

با دستور جدید جایگزین کنیم.

max-h-[calc(100vh\_-\_130px)]

130px عرض نوار منو با احتساب margin آن است.

|  |  |
| --- | --- |
| دستور overflow-auto با اعمال اسکرول مانع از سرریز میشود | بدون استفاده از دستور overflow-auto |
|  |  |



**پیاده سازی navigation bottom :**

» برای اینکه با کلیک روی هر آیتم فقط متن همان آیتم نمایش داده شود و متن بقیه آیتم ها مخفی شود از alpone.js استفاده میکنیم.

» برای اینکه از warp شدن متن آیتم ها جلوگیری کنیم باید هر آیتم بتواند expand شود. این expand شدن باید روی نگهدارنده تگ های a یعنی تگ های li صورت گیرد.

» از w-full و نیز از flex-1 استفاده کردیم ولی نتیجه حاصل نشد سپس از flex-auto استفاده کردیم.

|  |  |
| --- | --- |
| تأثیر flex-auto | حالت پیش فرض |
|  |  |

» به li ها رنگ text-gray-400 میدهیم و مقدار fillمربوط به Svg ها را به currentcolor تغییر می دهیم تا به این نحو svg از li ارث بری کند.

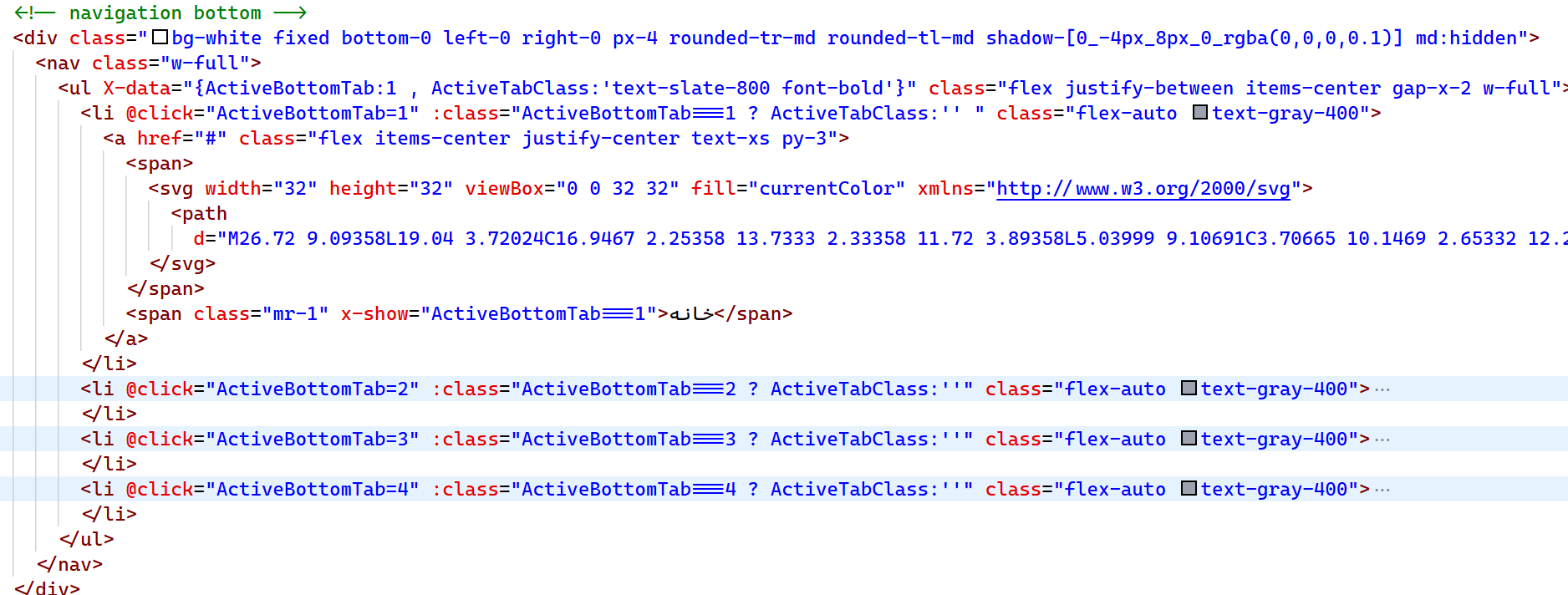
|  |
| --- |
| قبل |
|  |

|  |
| --- |
| بعد |
|  |

|  |  |
| --- | --- |
| » اگر svg داری opacity بود باید این خاصیت را نیز حذف کنیم در غیر اینصورت رنگ اعمال نمی شود. |  |
| » یک svg ممکن است بنا به شکلی که دارد چند fill داشته باشد که باید همه آنها حذف شوند. |  |

از box-shadow سفارشی در تیلویند استفاده میکنیم.

|  |
| --- |
|  |



جزئیات li ها :

