Abstract

Author: Anna Abajyan

Web technologies are so fast that it is becoming a tough job for businesses as well as developers to keep up the pace. New tools pop up every year letting the old ones go sideways and frontend development is a major part of the web. Businesses are focusing more on front-end development to enhance user interaction, site efficiency, interactivity and look & feel. And of course the popularity of front-end development tools is on the rise due to ever changing web techs. So developing my project I'm using some front-end development tools which are included in 2018 top list: Chrome Developer Tools, NPM Node JS, SASS and CodePen. Besides I'm using some CSS modern technics which are not supported by all browsers. So my goal is to show the power of CSS, what effects and what can be done with modern technics. And further you will see the final result of the website for a fictional company using only CSS. (No JavaScript and no Bootstrap).

Keywords: web development, frontend, CSS, SASS, website development, web tools

Licenses for Software and Content

Software Copyright License (to be distributed with software developed for masters project)

(This license is known as “The MIT License” and can be found at http://opensource.org/licenses/mit-license.php)

Content Copyright License (to be included with Technical Report)

Brief introduction

HTML(Hyper Text Markup Language) is the language in which most websites are written. HTML is used to create pages and make them functional. The code used to make them visually appealing is known as *CSS* (Cascading Stylesheet) - language allows a browser engine to paint elements of the page with specific features, like colors, positioning, or decorations.
SASS is a CSS preprocessor, an extension of CSS that adds power and elegance to the basic language. Sometimes you need a tool that will help you write maintainable, future-proof code, all while reducing the amount of CSS you have to write (keeping it DRY).Sass, one of the most popular front-end development tools in this category is a nine-year-old open-source project which pretty much defined the genre of modern CSS preprocessors. Although a little tricky to get to grips with initially, Sass's combination of variables, nesting, and mixins will render simple CSS when compiled, meaning your stylesheets will be more readable and (most importantly) DRY.
CodePen is a web development environment for front-end designers and developers. It is all about faster and smoother development. It is considered to be one of the top front-end development tools when it comes to better coding environment.

Node JS allows developers to write and run JavaScript applications on the server. Developers started using node.js to also write tools to help them with local web development.
NPM is a simple command line interface that allows developers to install and manage packages on their local computers. There are all kinds of open source tools, libraries and frameworks needed for modern development. Modern web development could simply not exist without a package manager.
Chrome Developer tools It is one of the top front-end development tools when it comes to hands-on debugging. You are going to run into many issues when building a new feature, adding a new page or fixing an existing issue. The Chrome Developer Tools are a set of debugging tools built into Chrome. These tools allow you to do a wide variety of development testing in your browser, which saves a ton of development time.
Using the 'Device mode' you can test how responsive the website will be. 'Sources panel' is used to debug your JavaScript using breakpoints. 'Timeline' helps you identify run-time performance issues.

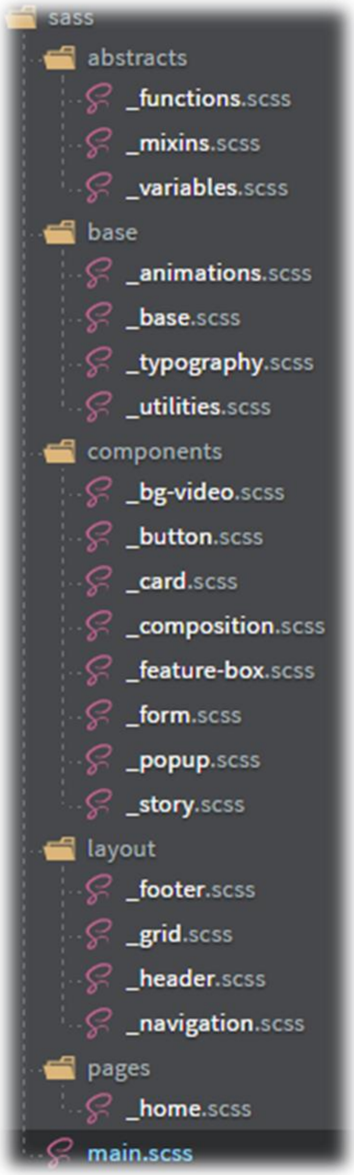
BEM (Block, Element and Modifier) methodology which gives our CSS code a solid structure that remains simple and easy to understand.

Considerations

Before starting we should take in mind 3 main principles while building the website:

1. Responsive design (responsive images…)
2. Maintainable and scalable code
3. Web performance (size of images…)

Maintainable and scalable code considered to be clean, modular, reusable and ready for growth. So we have to **think** about the layout of our webpage or web app before writing code (modular building blocks that make up interfaces; held together by the layout of the page; re-usable across a project, and between different projects; independent, allowing us to use them anywhere on the page). Next is to **build** our layout in HTML and CSS with a consistent structure for naming classes (Block Element Modifier: BLOCK: standalone component that is meaningful on its own; ELEMENT: part of a block that has no standalone meaning; MODIFIER: a different version of a block or an element). And the last is to create a logical **architecture** for our CSS with files and folders. For huge projects it's best to use the 7-1 pattern, means to have 7 different folders for partial Sass files, and 1 main Sass file to import all other files into a compiled CSS stylesheet. But considering that my project is not so huge I used only 5 folders and 1 main Sass file.



```
sass
  abstracts
    _functions.scss
    _mixins.scss
    _variables.scss
  base
    _animations.scss
    _base.scss
    _typography.scss
    _utilities.scss
  components
    _bg-video.scss
    _button.scss
    _card.scss
    _composition.scss
    _feature-box.scss
    _form.scss
    _popup.scss
    _story.scss
  layout
    _footer.scss
    _grid.scss
    _header.scss
    _navigation.scss
  pages
    _home.scss
  main.scss
```

- base/
- components/
- layout/
- pages/
- ~~themes/~~

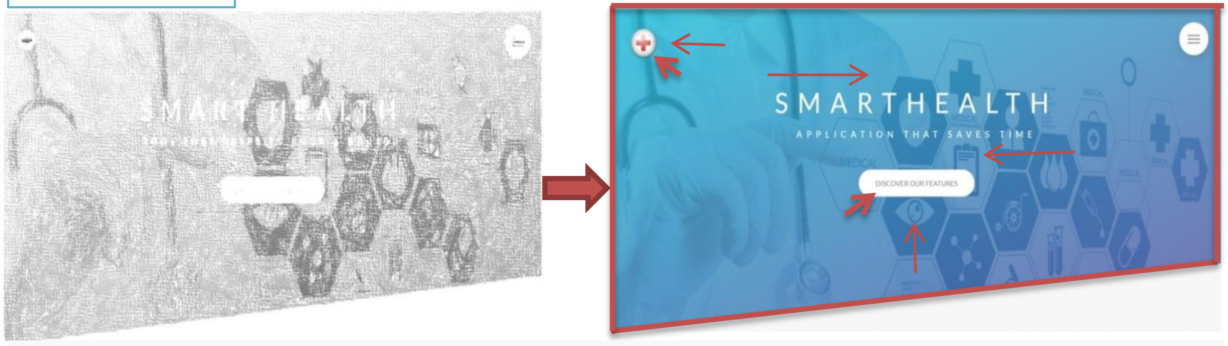
- abstracts/
- ~~vendors/~~

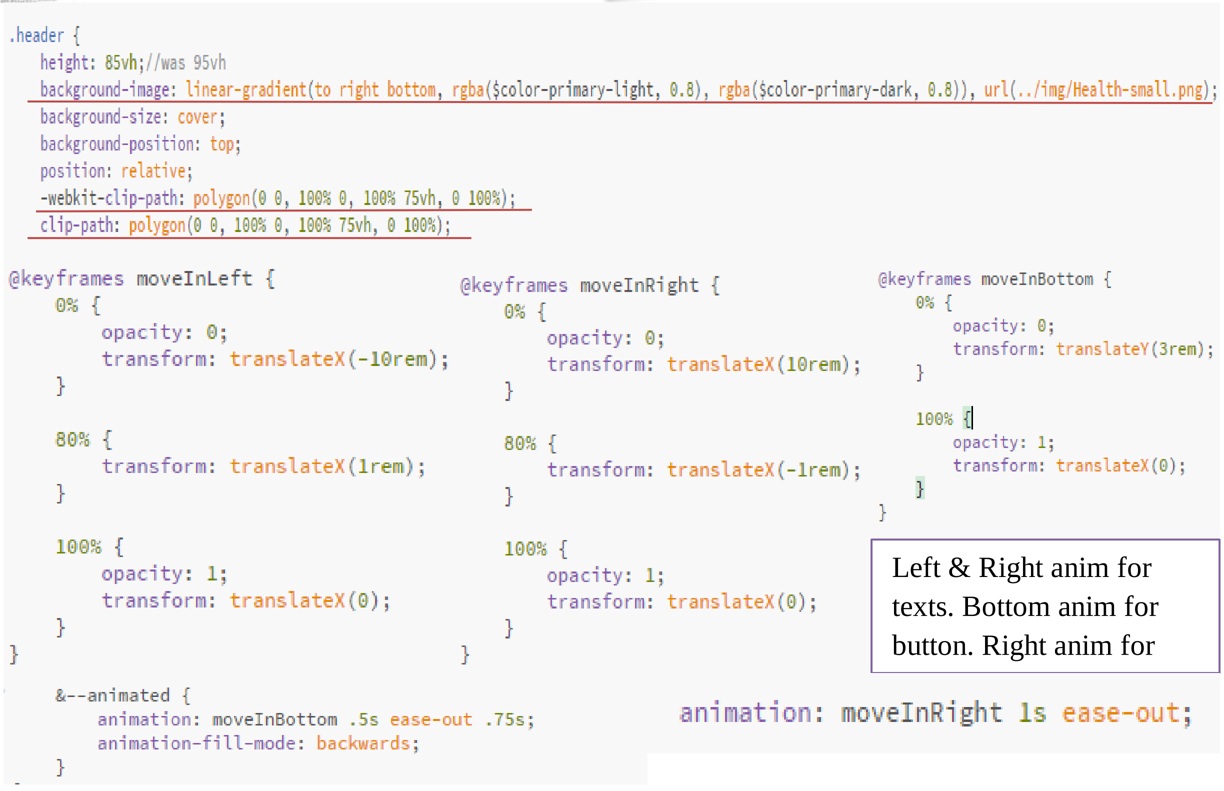
As to responsive design we have to consider 3 basic rules:
Fluid Grids and Layouts which allows content to easily adapt to the current viewport width used to browse. And using in website %-s rather than pixels(px) for all layout-related lengths.
Flexible/Responsive images
Images behave differently than text content, and so we need to ensure that they also adapt nicely to the current viewport.
Media Queries
To change styles on certain viewport widths (breakpoints), allowing us to create different version of our website for different widths.

The Process

Step1. Well at the beginning before developing my project I searched and read many articles and tutorials related to my project and chose what tools and methods should be used. Also looked through many designs of different websites and sketched the idea of how the website should look like, as without having proper idea we can't code. And the first was building the header. I will bring examples of more important parts of code in my project to my opinion, to show how it was implemented. So the first was developing the header of my website, which is generally the most important part of the website. In the header I used the **clip-path** CSS property that creates a clipping region that defines what part of an element should be displayed. More specifically, those portions that are inside the region are shown, while those outside are hidden. So as you can see (in picture1) I made polygon shape.
The background used the linear-gradient(), CSS function that creates an image consisting of a progressive transition between two or more colors along a straight line. Its result is an object of the <gradient> data type, which is a special kind of <image>. After implemented animation on texts, buttons and logo using a @keyframe rule with a name that is then used by the animation-name property to match an animation to its keyframe declaration. And each @keyframes rule contains a style list of keyframe selectors, which specify percentages along the animation when the keyframe occurs, and a block containing the styles for that keyframe.

Picture1.





```
.header {
    height: 85vh;//was 95vh
    background-image: linear-gradient(to right bottom, rgba($color-primary-light, 0.8), rgba($color-primary-dark, 0.8)), url(../img/Health-small.png);
    background-size: cover;
    background-position: top;
    position: relative;
    -webkit-clip-path: polygon(0 0, 100% 0, 100% 75vh, 0 100%);
    clip-path: polygon(0 0, 100% 0, 100% 75vh, 0 100%);
```

```
@keyframes moveInLeft {
    0% {
        opacity: 0;
        transform: translateX(-10rem);
    }

    80% {
        transform: translateX(1rem);
    }

    100% {
        opacity: 1;
        transform: translateX(0);
    }
}
```

```
@keyframes moveInRight {
    0% {
        opacity: 0;
        transform: translateX(10rem);
    }

    80% {
        transform: translateX(-1rem);
    }

    100% {
        opacity: 1;
        transform: translateX(0);
    }
}
```

```
@keyframes moveInBottom {
    0% {
        opacity: 0;
        transform: translateY(3rem);
    }

    100% {
        opacity: 1;
        transform: translateX(0);
    }
}
```

Left & Right anim for texts. Bottom anim for button. Right anim for

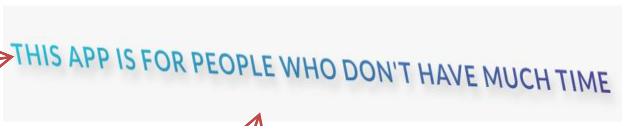
```
    &--animated {
        animation: moveInBottom .5s ease-out .75s;
        animation-fill-mode: backwards;
    }
```

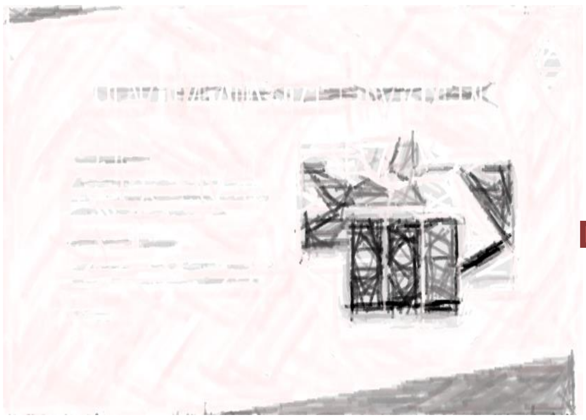
```
animation: moveInRight 1s ease-out;
```

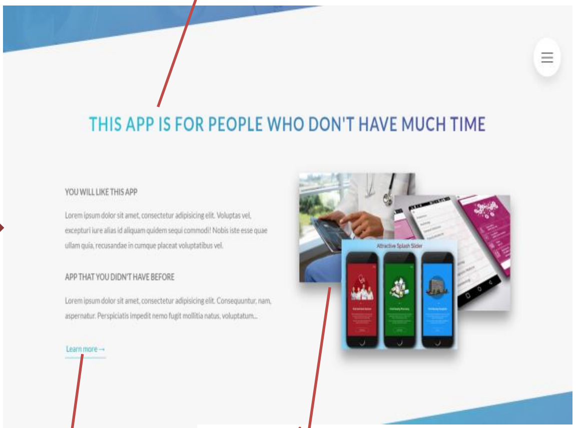
Step2.

Building the About section. I made the heading text gradient using background-clip, and later other headings are applied too except the main header. The background-clip CSS property specifies if an element's background, whether a <color> or an <image>, extends underneath its border. Background-clip: text means the background is painted within (clipped to) the foreground text. This property works only with prefix(-webkit-). Also used outline-offset property with outline property in images when hovering, this sets the amount of space between an outline and the edge or border of an element.

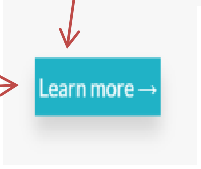
```
&:hover {
    transform: skewY(2deg) skewX(15deg) scale(1.1);
    text-shadow: .5rem 1rem 2rem rgba($color-black, .2);
}
```

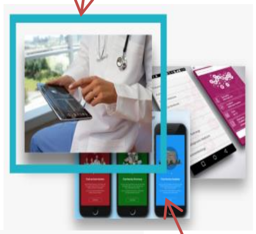





THIS APP IS FOR PEOPLE WHO DON'T HAVE MUCH TIME
YOU WILL LIKE THIS APP
APP THAT YOU DIDN'T HAVE BEFORE
Learn more →

```
&:hover {
    background-color: $color-primary;
    color: $color-white;
    box-shadow: 0 1rem 2rem rgba($color-black, .15);
    transform: translateY(-2px);// button up
}
```





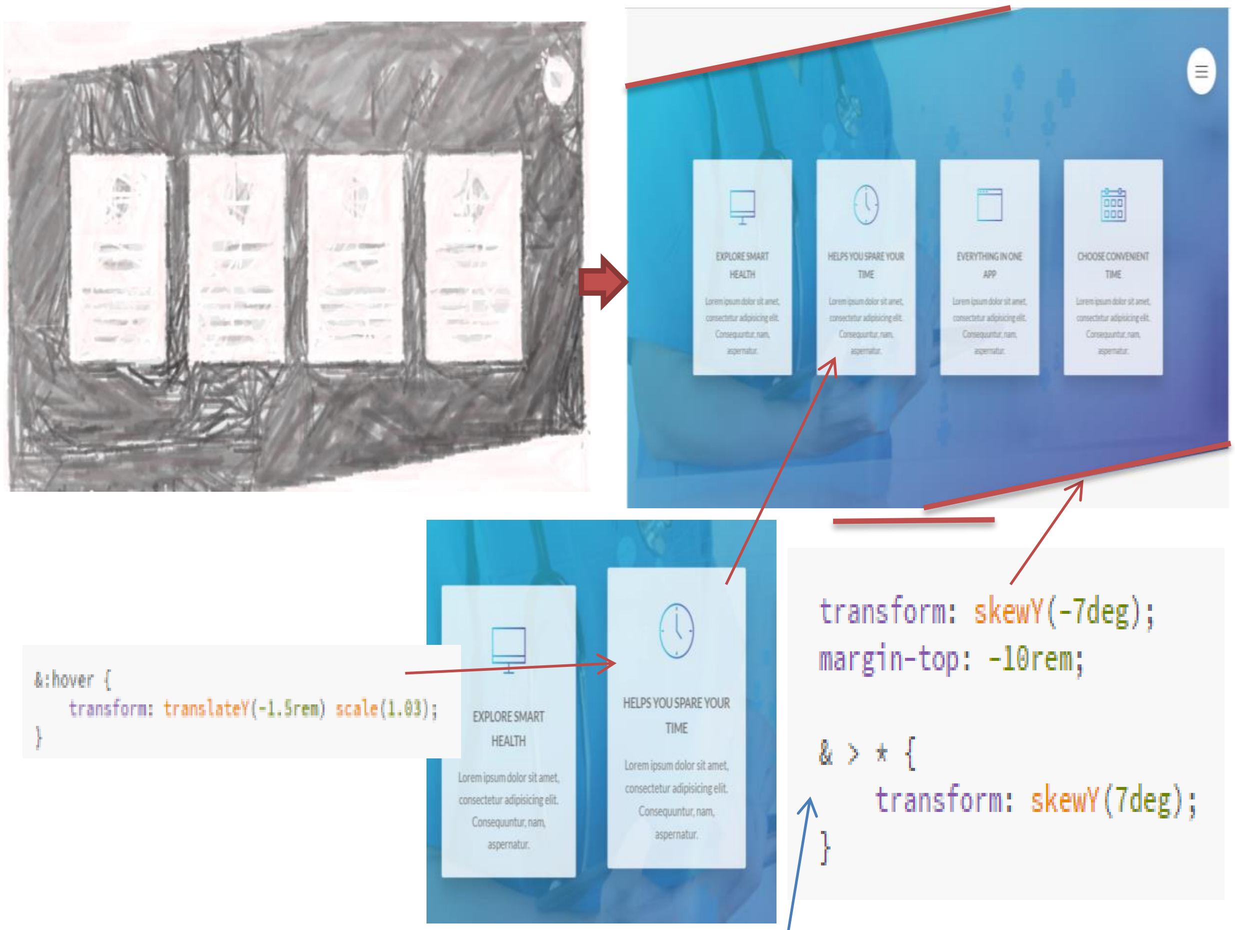
```
display: inline-block;
background-image: linear-gradient(to right, $color-primary-light, $color-primary-dark);
-webkit-background-clip: text;
color: transparent;
```

Picture2. About section

```
&:hover {
    outline: 1.5rem solid $color-primary;
    transform: scale(1.05) translateY(-.5rem);
    box-shadow: 0 2.5rem 4rem rgba($color-black, .5);
    z-index: 20;
}
```

Step3. Next section is features section.

As you can see I implemented linear-gradient background on this section again and other sections too which later you will see. Instead of using clip property I used here another trick called skew property. Further I used icons on each boxes, which were downloaded from source link (linea.io). And this file was included in CSS folder. Whenever you want to change icon you can simply choose from folder another one with different name, instead of searching in internet.



```
&:hover {
    transform: translateY(-1.5rem) scale(1.03);
}
```

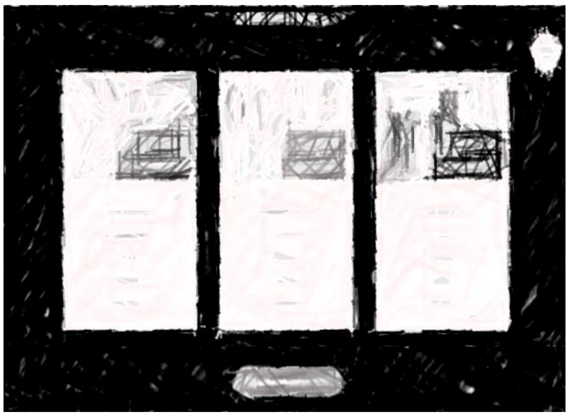
```
transform: skewY(-7deg);
margin-top: -10rem;

& > * {
    transform: skewY(7deg);
}
```

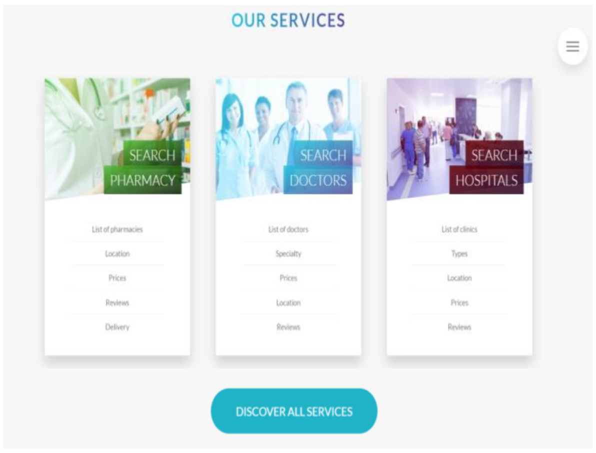
Picture3. Features

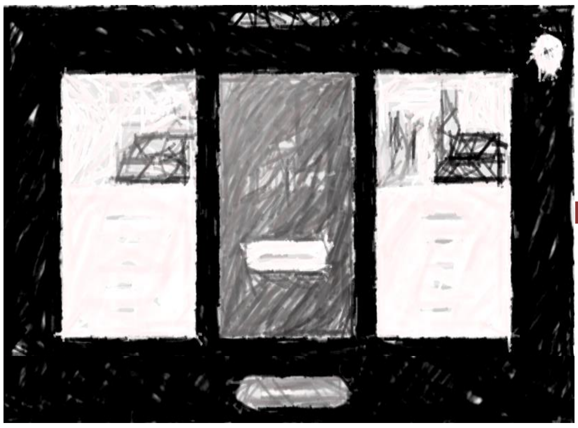
This selects everything that is direct to child

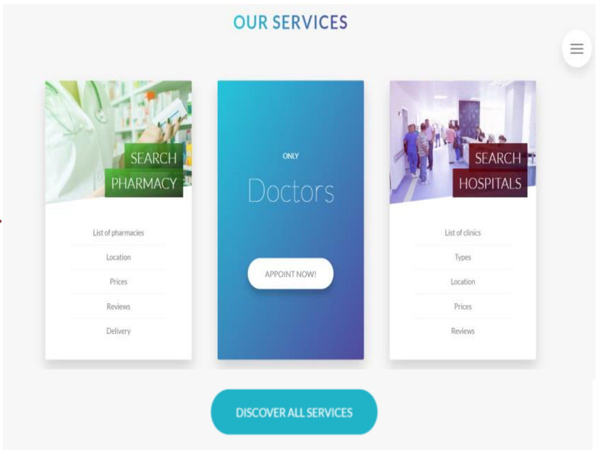
Step4. The Services section (here we have rotating cards)

Well in this section I used new property “perspective” which is not implemented in all of the browsers. The perspective CSS property determines the distance between the z=0 plane and the user in order to give a 3D-positioned element some perspective. Each 3D element with z>0 becomes larger; each 3D-element with z<0 becomes smaller. The strength of the effect is determined by the value of this property. The low the value the more dramatic is perspective.
On cards I used background-blend-mode and box-decoration-break CSS properties. The “background-blend-mode” property determines how an element's background images should blend with each other and with the element's background color. The “box-decoration-break” CSS property specifies how an element's fragments should be rendered when broken across multiple lines, columns, or pages.







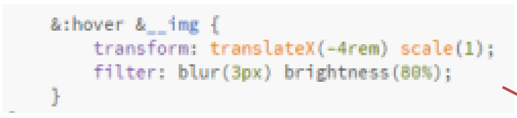


Step5. Story section

In the story section I created background video covering an entire section with the use of <video> element that specifies a standard way to embed a video in a web page. In my case also was used the object-fit CSS property that specifies how the contents of a replaced element, such as an <img> or <video>, should be resized to fit its container.

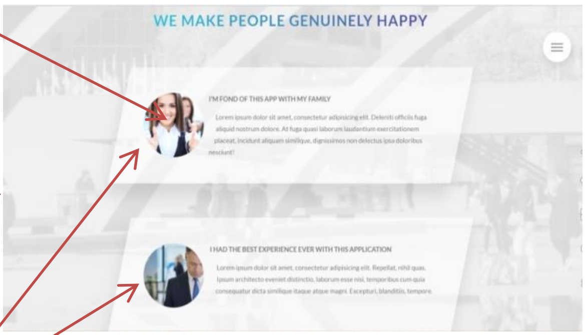
Also here I implemented the same technic (skew) on story boxes as in features section. On images applied the shape-outside CSS property which defines a shape—which may be non-rectangular—around which adjacent inline content should wrap. By default, inline content wraps around its margin box; shape-outside provides a way to customize this wrapping, making it possible to wrap text around complex objects rather than simple boxes.

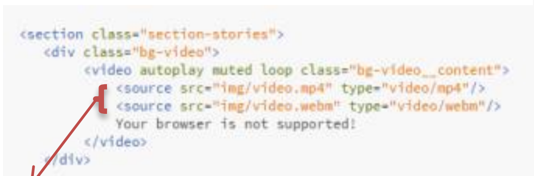
To achieve effects on images filter was used. The “filter” CSS property lets you apply graphical effects like blurring or color shifting to an element. Filters are commonly used to adjust the rendering of images, backgrounds, and borders.



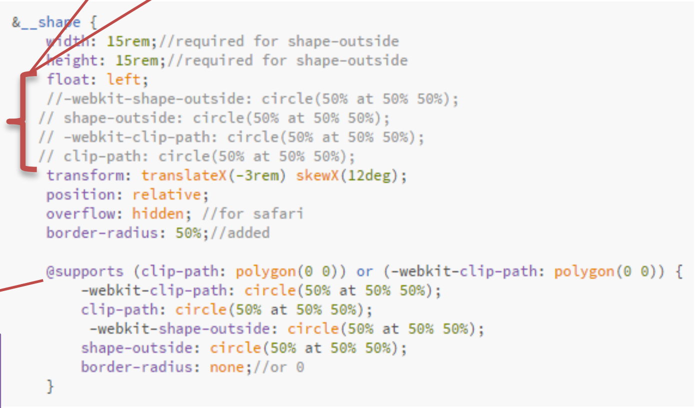
```
&:hover &__img {
    transform: translateX(-4rem) scale(1);
    filter: blur(3px) brightness(80%);
}
```

Picture5. Story section


WE MAKE PEOPLE GENUINELY HAPPY
I'M FOND OF THIS APP WITH MY FAMILY
I HAD THE BEST EXPERIENCE EVER WITH THIS APPLICATION



```
<section class="section-stories">
   <div class="bg-video">
        <video autoplay muted loop class="bg-video__content">
            <source src="img/video.mp4" type="video/mp4"/>
            <source src="img/video.webm" type="video/webm"/>
            Your browser is not supported!
        </video>
   </div>
```



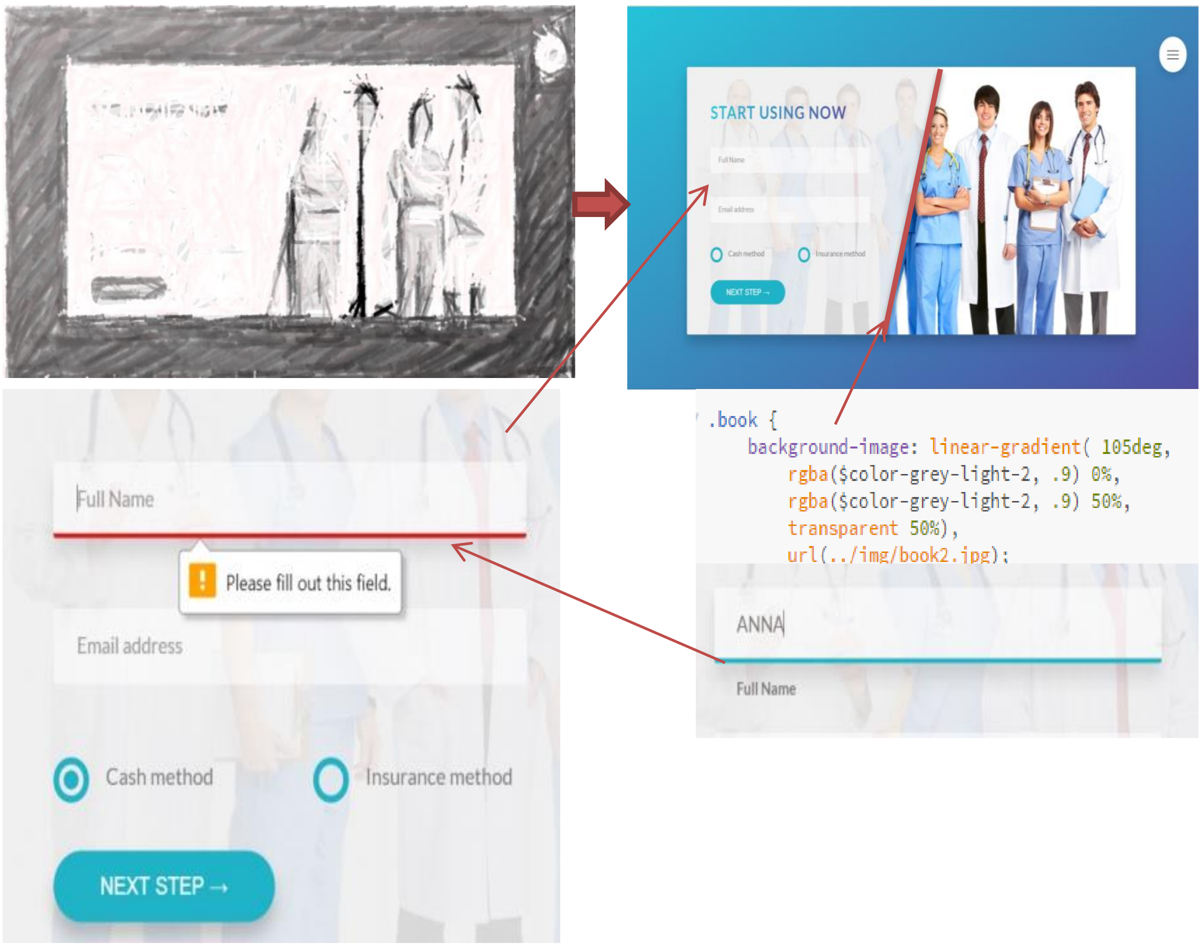
```
&__shape {
    width: 15rem;//required for shape-outside
    height: 15rem;//required for shape-outside
    float: left;
    //-webkit-shape-outside: circle(50% at 50% 50%);
   // shape-outside: circle(50% at 50% 50%);
   // -webkit-clip-path: circle(50% at 50% 50%);
   // clip-path: circle(50% at 50% 50%);
    transform: translateX(-3rem) skewX(12deg);
    position: relative;
    overflow: hidden; //for safari
    border-radius: 50%;//added

    @supports (clip-path: polygon(0 0)) or (-webkit-clip-path: polygon(0 0)) {
        -webkit-clip-path: circle(50% at 50% 50%);
        clip-path: circle(50% at 50% 50%);
         -webkit-shape-outside: circle(50% at 50% 50%);
        shape-outside: circle(50% at 50% 50%);
        border-radius: none;//or 0
    }
```

We need these 2 formats to make sure that they work across all of the

Graceful degradation concept, even if a browser doesn’t support a property that we want to use on site, then we can still apply it to the modern

Step6.Book section

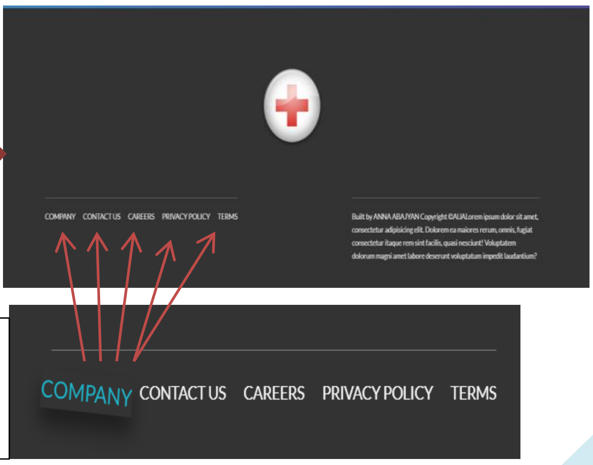
In using section I implemented solid-color gradient function instead of adding new element. Also put a placeholder, saying Full Name and then made it to be “required”. So the browser will not allow us to submit the form unless this field is actually filled out that’s what the required does. And there is also a type in HTML5 called email, where the browser will define if we’re writing in a valid Email address or not (same with the name). Further I used the “focus” which is very important for people who use a webpage without a mouse, but only with a keyboard. So when they move around with a keyboard on the webpage they need to know which form elements are actually focused. And for accessibility reasons we should never just do input focus and then set the outline to none, we should always make the form elements that are focused visible.



```
.book {
    background-image: linear-gradient( 105deg,
        rgba($color-grey-light-2, .9) 0%,
        rgba($color-grey-light-2, .9) 50%,
        transparent 50%),
        url(../img/book2.jpg);
```

Step7. Footer section

Well I made simple footer, with some effects on texts.



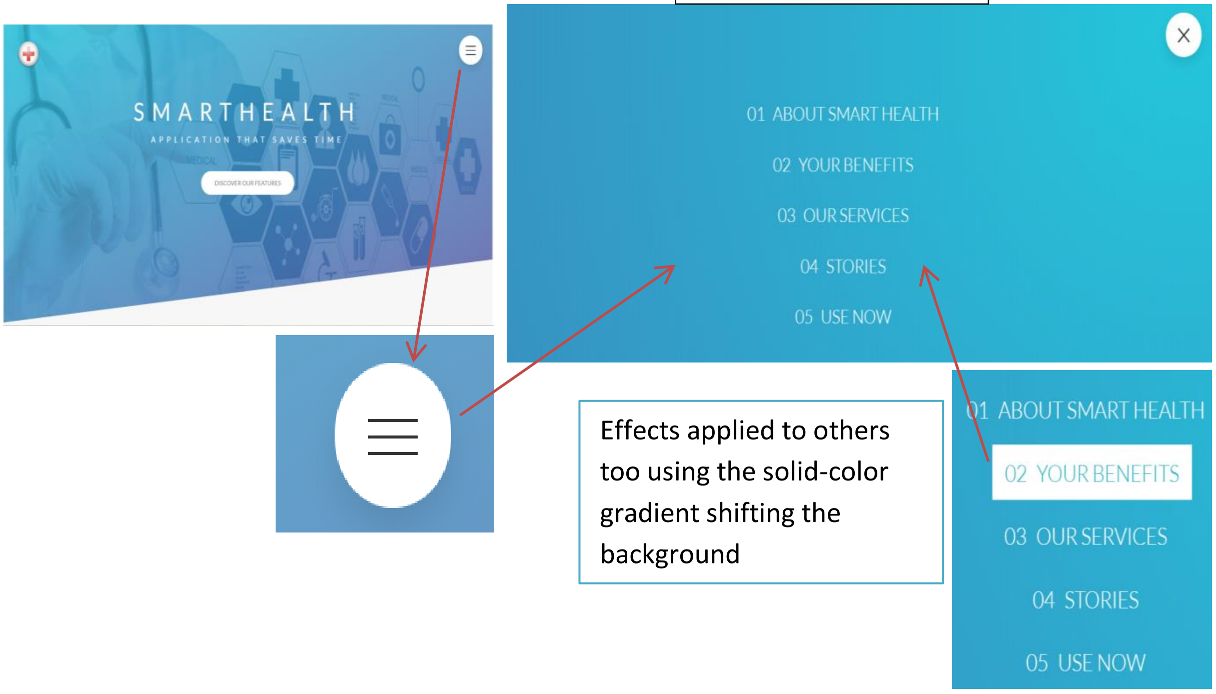
Picture7. Footer

This effect applied

Step8.Navigation

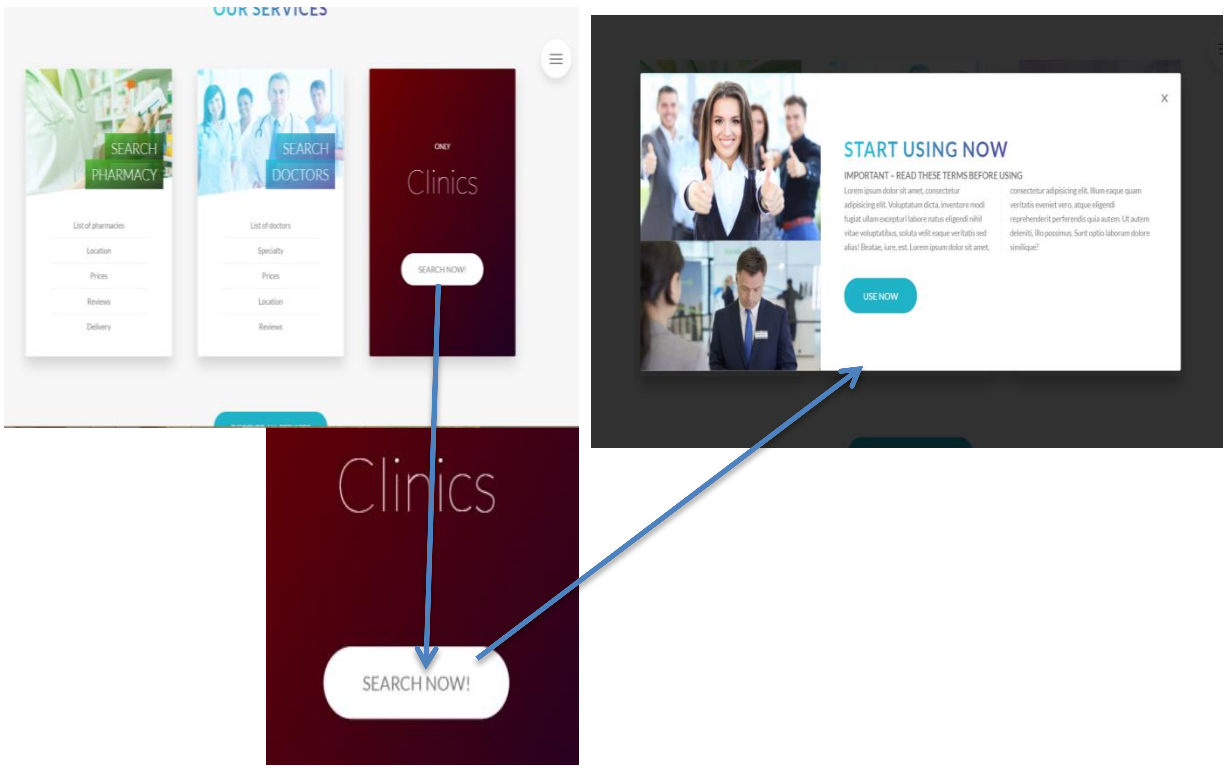
Making navigation I animated “solid-color gradient” and for the background of navigation I used not linear-gradient but new one – radial-gradient. The difference is that linear gradient goes from one side to the other one while the radial gradient starts in the middle of an element and goes from there in all the outside directions basically.

Picture9. Navigation



Step10. Popup

And of course not worth to mention that in order popup work we should use anchor in HTML adding id as we know ID’s are unique and then id=popup will become target. For popup I used the :target CSS pseudo-class that represents a unique element (the target element) with an id matching the URL's fragment. Then in text column layout that is something really new in CSS but we can make it work in most of the modern browsers already. The column-count CSS property break an element’s content into the specified number of columns. The column-gap CSS property sets the size of the gap (spacing) between an element's columns. The column-rule CSS property sets the width, style, and color of the rule (line) drawn between columns in a multi-column layout. Tried also to use the hyphens CSS property that specifies how words should be hyphenated when text wraps across multiple lines. We can prevent hyphenation entirely, use hyphenation in manually-specified points within the text, or let the browser automatically insert hyphens where appropriate. So for me in Google Chrome and other browsers except Firefox it didn't work even with the use of prefix.



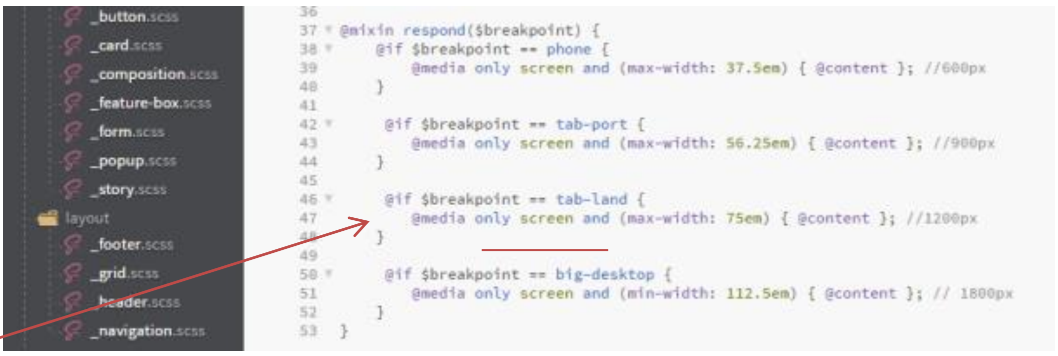
IMPORTANT – READ THESE TERMS BEFORE USING

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptatum dicta, inventore modi fugiat ullam excepturi labore natus eligendi nihil vitae voluptatibus, soluta velit eaque veritatis sed alias! Beatae, iure, est. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Illum eaque quam veritatis eveniet vero, atque eligendi reprehenderit perferendis quia autem. Ut autem deleniti, illo possimus. Sunt optio laborum dolore similique?

Step11. Wrap-ups

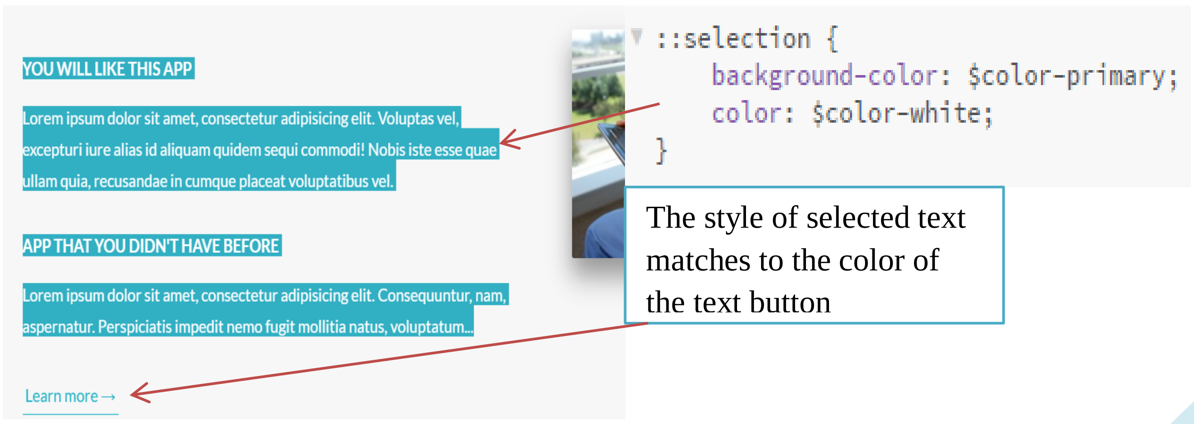
After finishing all sections I began to write media queries making responsive website, considering images as there are crucial part for web performance. Not worth to mention that media queries I wrote with use of mixins. Also at the beginning when I started to use rem, I considered to make the entire layout of the webpage dependent on the font-size. And that's a good thing because now in responsive design I can simply change the font size here and then all of these measurements in rem, all of them will scale up or down according to this font-size. So that's a huge advantage and time saver, otherwise, if I had everything in pixels, I should now go ahead and change all of these pixels to smaller numbers. But with this trick I simply need to increase or decrease the font size here and

then the design will basically shrink. Of course, this will not be enough to make the page responsive, there's a lot of stuff to adjust later on. Next the order of media queries is also important – always the larger ones before the smaller ones, which will bring the behavior we expect. After wrapping and writing all queries we test for browser support and at the end build our project with NPM scripts.
And the most important is that our responsive design won't work without having this meta element in the HTML head <meta name="viewport" content="width=device-width, initial-scale=1.0">. This tells that our website should be rendered with the width of the device, so the width of the content should be the device width. Further on a mobile touch device users do usually not hover and this means that we can't simply identify a touch device by the width of the screen alone. But good news is that in CSS we can actually write a media query which can identify if the user can hover over elements or not. And with that we can easily identify touch devices and non-touch devices, where we have mouse.



```scss
@mixin respond($breakpoint) {
    @if $breakpoint == phone {
        @media only screen and (max-width: 37.5em) { @content }; //600px
    }

     @if $breakpoint == tab-port {
        @media only screen and (max-width: 56.25em) { @content }; //900px
    }

     @if $breakpoint == tab-land {
        @media only screen and (max-width: 75em) { @content }; //1200px
    }

     @if $breakpoint == big-desktop {
        @media only screen and (min-width: 112.5em) { @content }; // 1800px
    }
 }
```

@if is basically makes a test and if the result of this text is true then something happens. So if the breakpoint is phone then we want something to happen. Well as you see I wrote queries in mixins then I included them in files, in parts of code where I had problems decreasing or increasing the screen. And of course "only screen" means that the media query only gets applied to screens "and" then the max width. So if someone wants to copy the text from our webpage they're not allowed.



YOU WILL LIKE THIS APP

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptas vel, excepturi iure alias id aliquam quidem sequi commodi! Nobis iste esse quae ullam quia, recusandae in cumque placeat voluptatibus vel.

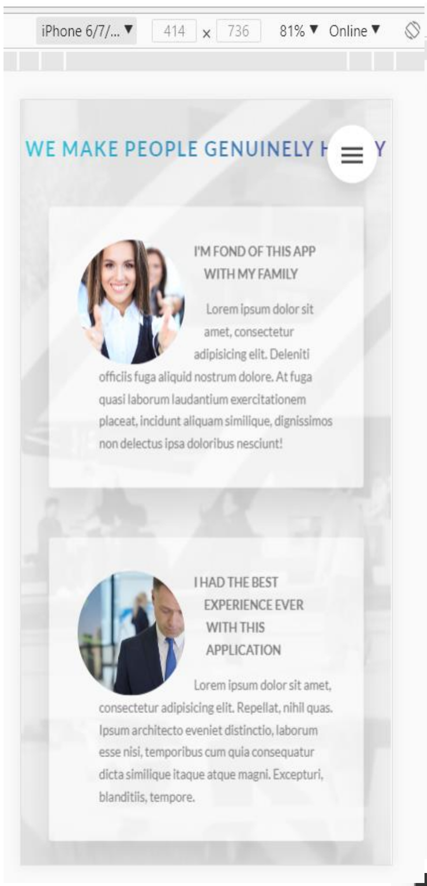
APP THAT YOU DIDN'T HAVE BEFORE

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Consequuntur, nam, aspernatur. Perspiciatis impedit nemo fugit mollitia natus, voluptatum...

Learn more →

```scss
::selection {
    background-color: $color-primary;
    color: $color-white;
}
```

The style of selected text matches to the color of the text button



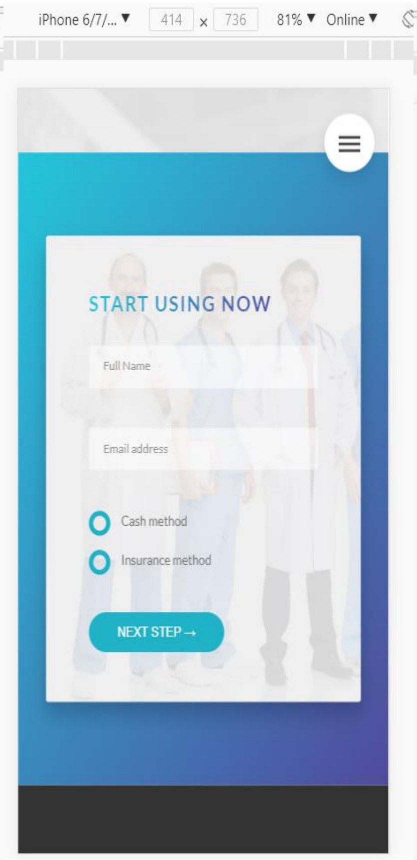
WE MAKE PEOPLE GENUINELY H Y

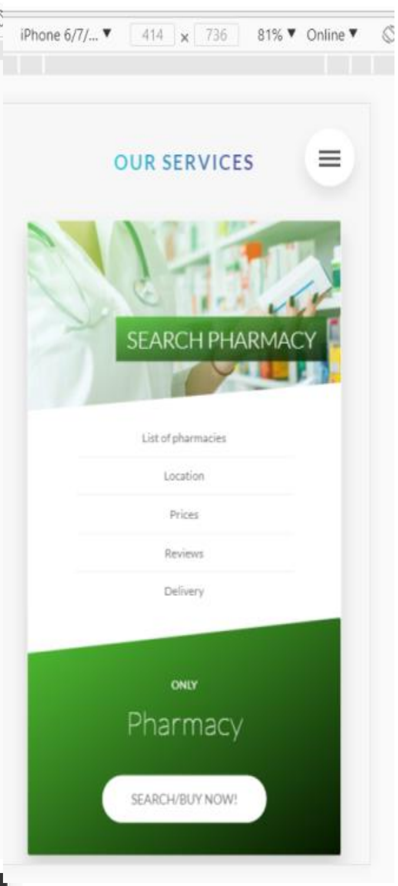
I'M FOND OF THIS APP WITH MY FAMILY

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Deleniti officiis fuga aliquid nostrum dolore. At fuga quasi laborum laudantium exercitationem placeat, incidunt aliquam similique, dignissimos non delectus ipsa doloribus nesciunt!

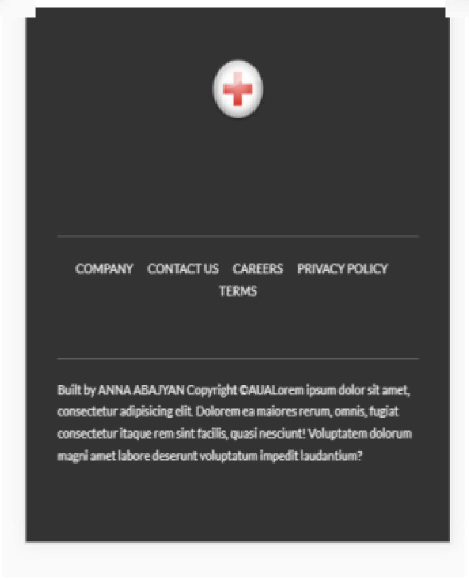
I HAD THE BEST EXPERIENCE EVER WITH THIS APPLICATION

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Repellat, nihil quas. Ipsum architecto eveniet distinctio, laborum esse nisi, temporibus cum quia consequatur dicta similique itaque atque magni. Excepturi, blanditiis, tempore.





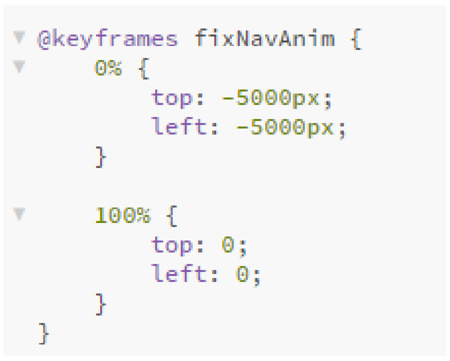
Some snapshots from testing in small



Built by ANNA ABAJYAN Copyright ©AUALorem ipsum dolor sit amet, consectetur adipisicing elit. Dolorem ea maiores rerum, omnis, fugiat consectetur itaque rem sint facilis, quasi nesciunt! Voluptatem dolorum magni amet labore deserunt voluptatum impedit laudantium?

Step12. Issues

Developing this project I had many issues especially creating animations. I will bring few examples related to this. At the end of the project I noticed that when refreshing the page the navigation list was showing for mili seconds which wasn't so good. The same I had with popup it also was appearing on the header page while refreshing. The first problem was fixed with @keyframes fixNavAnim & second took much longer experimenting and changing different properties, but it was fixed happily. Also when something weird happens with animation and translating stuff the property backface-visibility helped, as it fixes such issues. And there were a lot of other stuffs related not only to animations which I was searching in different sources how to fix it and sometimes reading developers community Q&A.



```
@keyframes fixNavAnim {
    0% {
        top: -5000px;
        left: -5000px;
    }

    100% {
        top: 0;
        left: 0;
    }
}
```

Conclusion

In conclusion I can say that overall I reached my goal to show the power of using CSS only and in some sence this project was experimental as most modern technics weren’t supported by all browsers. And in future I hope these issues will be fixed without using any prefixes or waiting when the old browsers will be implemented too. So I used top modern browsers(latest versions) of them. Contrary to popular belief, CSS is not only used to provide basic style for a web page in order to make it look more attractive. There are plenty of other things that one can do with CSS as well. With the ability to create animations and interactions CSS along with HTML allow web developers the opportunity to play around and experiment with different methods.

References:

- https://developer.mozilla.org/en-US/ (most used)
- https://www.w3schools.com/ (most used)
- http://bennettfeely.com/clippy/
- https://uigradients.com
- https://caniuse.com/ (most used)
- https://sass-guidelin.es
- https://www.valuecoders.com/blog/technology-and-apps/top-15-front-end-development-tools-2018/
- https://vanseodesign.com/css/sass-directory-structures/
- https://css-tricks.com/snippets/css/media-queries-for-standard-devices/
- https://www.sitepoint.com/css-architecture-block-element-modifier-bem/
- http://getbem.com/introduction/
- https://coverr.co