Теория

CSS3 3D-трансформации создают объемные реалистичные эффекты на веб-страницах. 3D-трансформации работают аналогично с 2D-трансформациями с разницей в том, что элементы могут перемещаться вдоль оси Z, вглубь экрана и из него. Чтобы активизировать 3D-пространство, нужно установить свойство `perspective` для родительского контейнера.

Хотя некоторые значения свойства `transform` позволяют преобразовывать элемент в трехмерной системе координат, сами элементы не являются трехмерными объектами. Они существуют в двумерной плоскости (плоская поверхность) и не имеют глубины.

Установка 3D-перспективы perspective

В нормальном потоке элементы отображаются плоскими и в той же плоскости, что и блок, содержащий их. Двумерные функции преобразования могут изменять внешний вид элемента, но этот элемент по-прежнему отображается в той же плоскости, что и содержащий его блок.

Свойства `perspective` и `perspective-origin` можно использовать для добавления ощущения глубины в сцену, делая элементы выше по оси Z (ближе к зрителю) и кажущимися большими, а те, которые находятся дальше — меньшими. Масштаб пропорционален `d / (d - Z)`, где `d` — значение перспективы, является расстоянием от плоскости рисования до предполагаемого положения глаза зрителя.

Если 3D-перспектива задается с помощью функции `perspective()`, 3D-пространство активизируется только для одного элемента. Свойство `perspective` активирует 3D-пространство внутри элемента, содержащего дочерние трансформированные элементы и применяется к ним. Свойство не наследуется.

perspective

Значения:

длина	Задает глубину просмотра, т.е. расстояние по оси Z. Значение может быть любым положительным числом. Если единица измерения не указана, по умолчанию она считается в `px`. Чем больше значение, тем менее выражен эффект. `0` означает отсутствие перспективы.
`none`	Значение по умолчанию. Означает отсутствие перспективы.

initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Задание точки трансформации для 3D-элемента perspective-origin

Свойство устанавливает точку начала координат для свойства perspective. Значение по умолчанию perspective-origin: 50% 50%;. Позволяет изменять направление трансформации дочернего 3D-элемента. Свойство должно использоваться вместе со свойством perspective для блока-контейнера и свойством transform для дочернего элемента. Не наследуется.

perspective-origin

Значения:

значение по оси X/значение по оси Y	Первое значение указывает положение элемента по оси X. Может задаваться в единицах длины, в % или одним из трех ключевых слов: left (эквивалентно 0% по оси X), center (эквивалентно 50% по оси X) или right (эквивалентно 100% по оси X). Второе значение указывает положение по оси Y. Задается в единицах длины, в % или одним из трех ключевых слов: top (эквивалентно 0% по оси Y), center (эквивалентно 50% по оси Y) или bottom (эквивалентно 100% по оси Y).
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Стиль 3D-преобразований transform-style

Свойство определяет, как дочерние трансформированные элементы (элементы, для которых задано свойство transform) отрисовываются в трехмерном пространстве. Задается для блока-контейнера. Не наследуется.

transform-style

Значения:

flat	Значение по умолчанию. Все дочерние элементы отображаются плоскими в двумерной плоскости блока-контейнера.
preserve-3d	Располагает элементы в трехмерном пространстве.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Видимость обратной стороны элемента backface-visibility

Свойство определяет, будет ли видна пользователю обратная сторона преобразованного элемента. У непреобразованного элемента к пользователю обращена передняя сторона. Не наследуется.

backface-visibility

Значения:

visible	Значение по умолчанию. Указывает, что обратная сторона видна.
hidden	Скрывает обратную сторону элемента.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Задание: создайте страницу с трансформацией (см. гифку)

1. Создадим новую страницу и файл стилей. Укажем стиль для тега <body>, <h1>,<h2>. Установим фоновое изображение, укажем надписи, которые будут в блоках.

Результат

Пример css

```css
body {
  margin: 0;
  background-image: url('SNuymfs.jpg');
  background-repeat: no-repeat;
  background-position: 50% 50%; /*расположение по центру фонового*/
  background-size: cover; /*Масштабирует изображение с сохранением пропорций так, чтобы его ширина или высота равнялась ширине или высоте блока.*/
  height: 100vh;
  padding: 20px;
  font-size: 16px;
  font-family: 'Forte', sans-serif;
  overflow: hidden; /*Отображается только область внутри элемента, остальное будет скрыто.*/
}

h1 {
  color: white;
  font-size: 5em;
  margin: 0;
  transform: translate(-50%, -50%);/*Сдвигает элемент на заданное значение по горизонтали и вертикали.*/
  position: absolute;
  top: 50%;
  left: 50%;
}
h2 {
  font-size: 3em;

}
```

Пример html

```html
<html>
 <head>
  <meta charset="utf-8">
  <title>9</title>
<link rel="stylesheet" type="text/css" href="laba9.css" />
 </head>
 <body>
  <h1>LabaN9</h1>
  <h2>about me</h2>
 </body>
</html>
```

2. Добавим на страницу панель с иконками, на которые установим ссылки на социальные сети.

Шрифт Awesome — это коллекция масштабируемых векторных иконок. Иконки можно форматировать с помощью css-свойств, устанавливать для них цвет, размер, тень и многое другое. Шрифт версии 4.5.0 включает 605 иконок.

Чтобы подключить:

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css">

Результат

Пример css

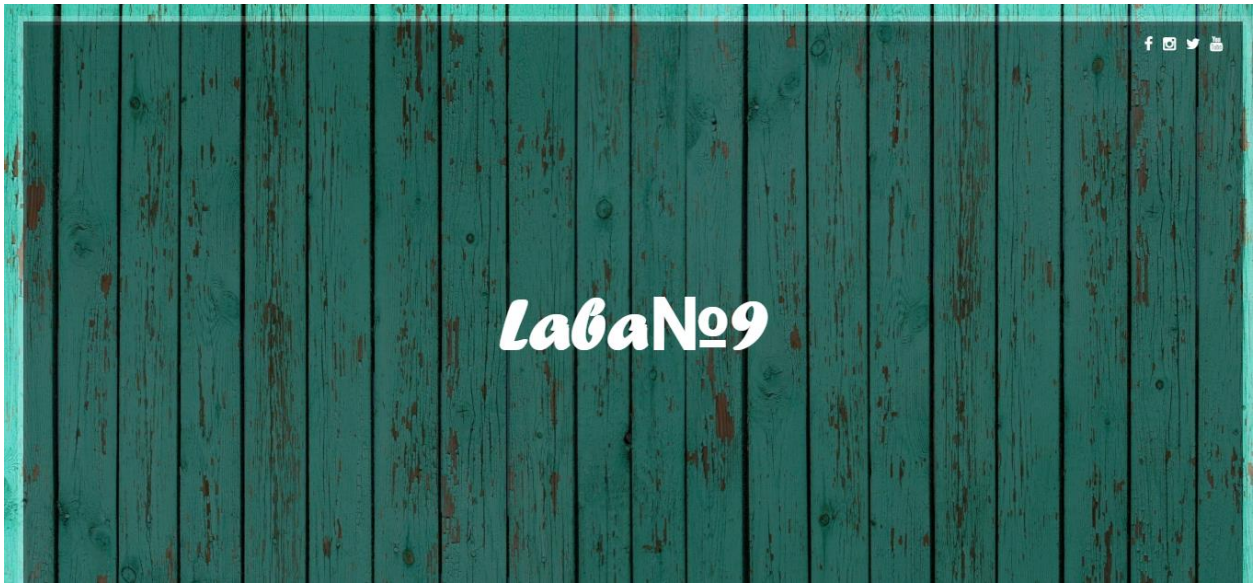
```
.share-block {
  float: right;
  padding-right: 15px;
  line-height: 50px;
}
.share-block a {
  color: #ffffff;
  display: inline-block; /*Это значение генерирует блочный элемент, который обтекается другими элементами веб-страницы подобно встроенному элементу.*/
  margin-right: 7px;
  transition: .4s linear;
}
.share-block a:hover {
  color: #48c5b3;
}
```

Пример html

```
<div class="share-block">
    <a href=""><i class="fa fa-facebook"></i></a>
    <a href=""><i class="fa fa-instagram"></i></a>
    <a href=""><i class="fa fa-twitter"></i></a>
    <a href=""><i class="fa fa-youtube"></i></a>
</div>
```

3. Создадим блок, который будет вращаться. Установим правила для передней и задней панели.

Результат



Пример html

```
<body>
<div class="container">
<div class="flipper" id="target">
<div class="front">
<div class="share-block">
        <a href=""><i class="fa fa-facebook"></i></a>
        <a href=""><i class="fa fa-instagram"></i></a>
        <a href=""><i class="fa fa-twitter"></i></a>
        <a href=""><i class="fa fa-youtube"></i></a>
      </div>
      <h1>Laba№9</h1>
    </div>
<div class="back">
<div class="inner">

<h2>about me</h2>
</div>
 </div>
</div>
</div>
</body>
```

Пример css

```css
.container {
  width: 100%;
  height: 100%;
    -webkit-perspective: 1200;
  perspective: 1200;
  -moz-transform: perspective(1200px);
  -webkit-transform-style: preserve-3d;
  -moz-transform-style: preserve-3d;
  transform-style: preserve-3d;
  position: relative;
}

.flipper {
  border: 10px solid rgba(255,255,255,0.2);
  position: relative;
  width: 100%;
    height: 100%;
  transform-style: preserve-3d;
  transition: .7s linear;
}

.front, .back {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0,0,0,.5);
  backface-visibility: hidden;
}
.front {
  z-index: 2;
  transform: rotateX(0deg);
}
.back {
  transform: rotateX(-180deg);
  color: white;
}
.flipper:target {
  transform: rotateX(180deg);
}
.inner {
  padding: 20px;
  line-height: 2;
  letter-spacing: 1px;
}
```

4. Создадим «кнопки» перехода и зададим правила для них.

Резальтат

Пример css

```
.button {
    text-decoration: none;
    width: 120px;
    height: 50px;
    position: relative;
    display: inline-block;
    top: 0;
    left: 0;
    z-index: 3;
    line-height: 50px;
    padding-left: 30px;
    transition: .4s linear;
}
.button:before {
    content: "";
    position: absolute;
    top: 0;
    left: 0;
    width: 0;
    height: 0;
    border-top: 20px solid #CE1D5A;
    border-right: 20px solid transparent;
}
.button:after {
    content: "";
    position: absolute;
    top: 8px;
    left: 8px;
    width: 0;
    height: 0;
    border-top: 20px solid white;
    border-right: 20px solid transparent;
}
.home:hover, .close:hover {
    color: white;
}
.home, .close {
    color: #999;
    font-size: 20px;
}
```

Пример html

```
<body>
<div class="container">
<div class="flipper" id="target">
<div class="front">
<a href="#target" class="home button">about me</a>
<div class="share-block">
        <a href=""><i class="fa fa-facebook"></i></a>
        <a href=""><i class="fa fa-instagram"></i></a>
        <a href=""><i class="fa fa-twitter"></i></a>
        <a href=""><i class="fa fa-youtube"></i></a>
      </div>
      <h1>Laba№9</h1>
    </div>
<div class="back">
<a href="#close" class="close button"><i class="fa fa-home"></i></a>
<div class="inner">

<h2>about me</h2>
</div>
  </div>
</div>
</div>
</body>
```