Анимация и трансформация

Данильченко Анна Александровна

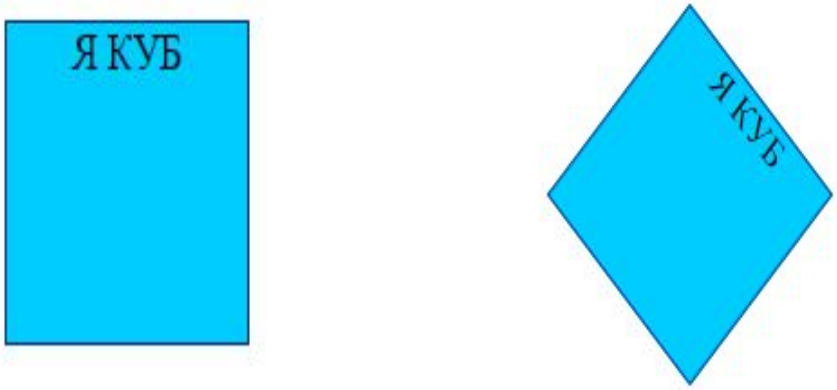
Преподаватель кафедры программного обеспечения систем ЖГТУ

Трансформация

Трансформации позволяют преобразовывать элементы в **двухмерном** и **трехмерном** пространствах.

Поворот

Transform: rotate(45deg);



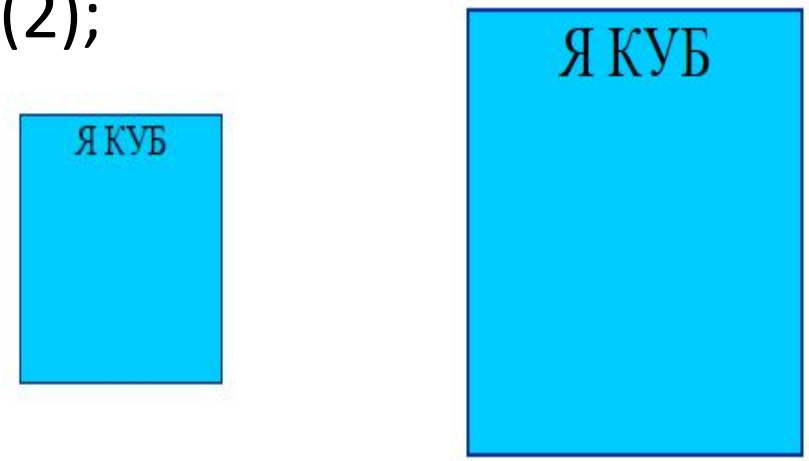
Единицы измерения угла поворота CSS3-трансформации

Единица измерения	CSS3-обозначение	Описание	Пример
Градусы	*deg*	Угол полной окружности 360°	rotate(90deg)
Грады	*grad*	Единица измерения плоских углов, равная 1/400 угла полной окружности или π/200	rotate(100grad)
Радианы	*rad*	2π радиан равно углу полной окружности	rotate(1.57rad)
Обороты	*turn*	1 угол полной окружности равен одному обороту	rotate(.25turn)

Масштабирование scale()

Значение масштаба задается относительно единицы: scale(2) - исходный элемент будет увеличен в два раза, scale(0.5) - элемент уменьшается в два раза

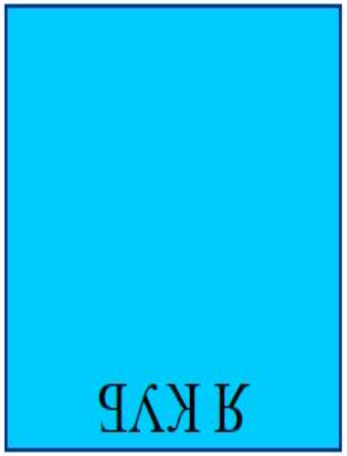
```
transform: scale(2);
```



Можно задать направление: X - по горизонтали, Y - по вертикали, Z - глубина масштабирования.

transform: scaleX(2);

scale() можно использовать для создания эффекта отражения. Для этого нужно передать отрицательное значение в функцию scale().

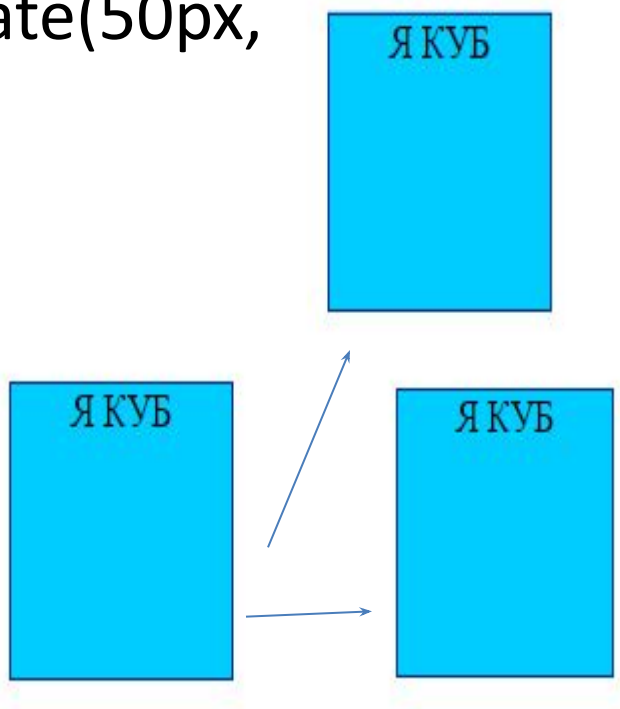


transform: scaleX(-1);

transform: scale(-2);

Перемещение translate(x,y), *translateX(x),translateY(y)*

transform: translate(50px, -4em);



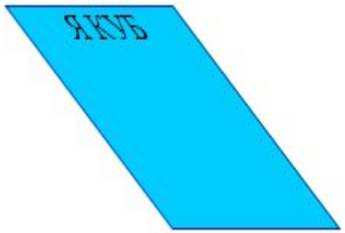
transform: translateX(50px);

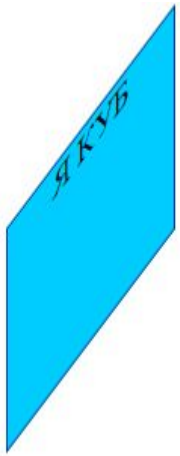
Искажение

skew() искажает форму элемента

transform: skewX(50grad);

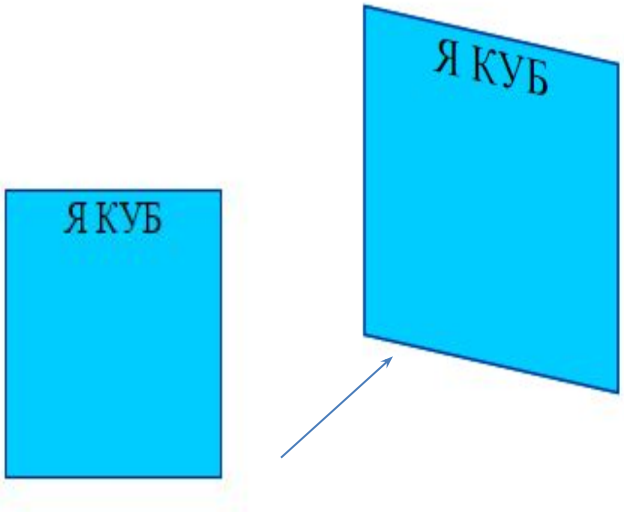
transform: skewY(-50grad);





Использование нескольких трансформаций

transform: scale(1.5) translateX(10px) skew(10deg) rotate(0.175rad);



Сначала элемент увеличивается в полтора раза (используя трансформацию scale), потом перемещается на 10 пикселов влево (посредством трансформации translateX) и, наконец, наклоняется и поворачивается (используя трансформации skew и rotate).

2D

кроссбраузерный CSS

-moz-transform - Firefox

-o-transform - Opera

-webkit-transform - Safari

Transform

Двухмерные трансформации

translate(10px, 10px)

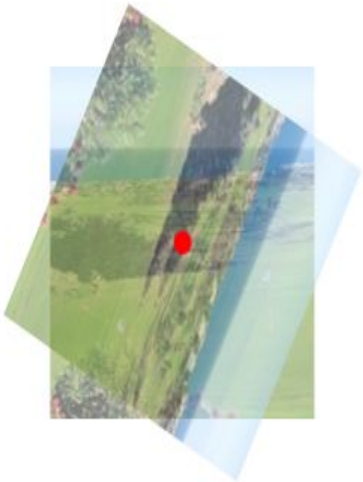
rotate(140deg)

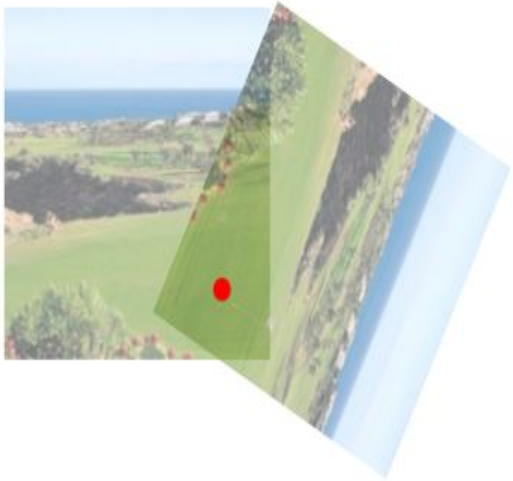
scale(1.2, 1.5)

skew(30deg, 56deg)

Точка, относительно которой...

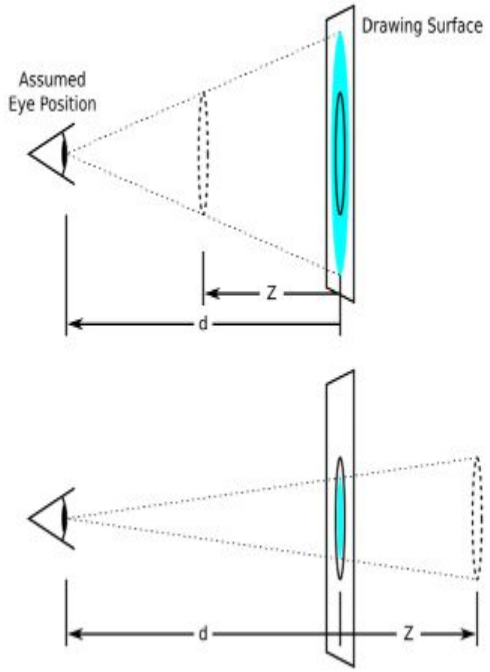
```
transform-origin: 50% 50%; // дефолт

transform-origin: 123px 80%;
```





Перспектива

применяется при трансформациях элемента
расстояние в пикселях от плоскости дочерни
точки из которой пользователь как бы смо

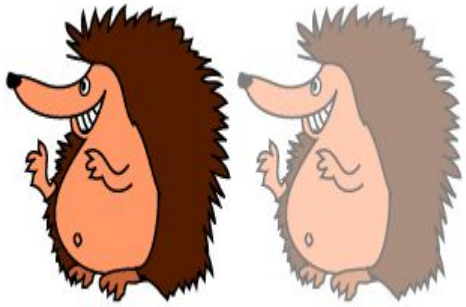


- perspective: 1000;
- perspective: 125px;

opacity

Определяет уровень прозрачности элемента веб-страницы. При частичной или полной прозрачности через элемент проступает фоновый рисунок или другие элементы, расположенные ниже полупрозрачного объекта.

- .semi { opacity: 0.5; /* Полупрозрачность элемента */ }



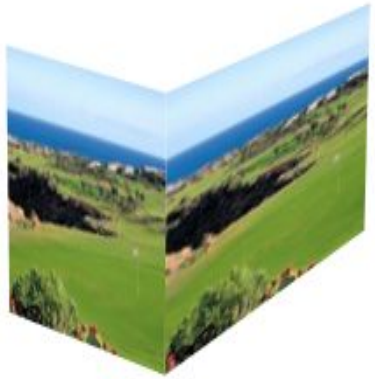
3D

Трехмерные трансформации

translate3d(10px, 5px, 20px)

rotate3d(1, 0, 1, 140deg)

scale3d(1.2, 1.5, 1.1)



Определяет, как дочерние элементы должны отображаться в 3D-пространстве

```
01. transform-style: preserve-3d;
02. transform-style: flat;
```

flat	Дочерние элементы лежат в той же плоскости, что и их родитель.
preserve-3d	Дочерние элементы будут отображаться в 3D-пространстве.

Как использовать

```
.outer-wrapper
 .inner-wrapper
 .block
```

```
.outer-wrapper{ perspective: 1000px; }
.inner-wrapper{ transform-style: flat; }
.block{ transform: rotate3d(0, 1, 0, 40deg); }
```

Обратная сторона элемента

```
backface-visibility: visible;

backface-visibility: hidden;
```

Анимация

Как это работает

```
<div></div>
```

```
01. div{
02.  width: 100px;
03. }
04. div:hover{
05.  width: 200px;
06. }
```

Анимация

Как это работает с переходами

```
<div></div>
```

```
01. div{
02.  width: 100px;
03. transition: 0.5s;
04. }
05. div:hover{
06.  width: 200px;
07. }
```

Transition

Универсальное свойство, которое позволяет одновременно задать значения transition-property, transitio n-duration, transition-timing-function и transition-delay. Устанавливает эффект перехода между двумя состояниями элемента.

Свойства должны указываться в следующем порядке:

- **transition-property** указывает CSS свойства, которые будут задействованы для создания перехода.
- **transition-duration** указывает время, в течении которого будет совершен переход.
- **transition-timing-function** указывает функцию смягчения отвечающую за плавность выполнения перехода.
- **transition-delay** устанавливает величину задержки перед началом выполнения перехода.

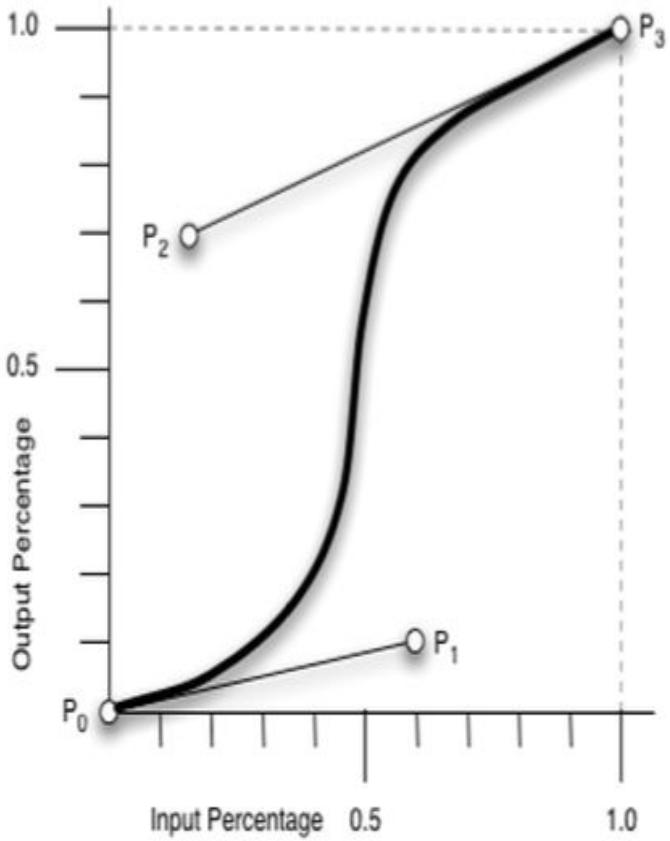
Свойство *transition*

```
transition-property: all;// width, opacity, color...

transition-duration: 2.3s;// 0, 134ms, 0.3s...

transition-timing-function: ease;// linear, ease-in...

transition-delay: 500ms;// 3s, 230ms...


transition: width 2.3s linear 500ms;
```

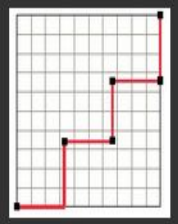
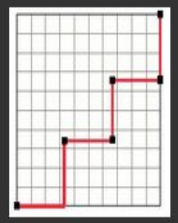
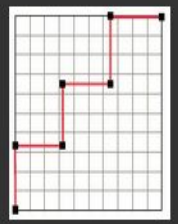
transition-timing-function

linear переход будет иметь одинаковую скорость на протяжении всего времени выполнения;
ease переход будет иметь медленную скорость выполнения в начале потом начнет ускоряться и снова замедляется в конце (*является значением по умолчанию*);
ease-in переход будет иметь медленную скорость выполнения в начале;
ease-out переход будет иметь медленную скорость выполнения в конце;
ease-in-out переход будет иметь медленную скорость выполнения в начале и в конце;
cubic-function(x,x,x,x) позволяет задать плавность выполнения перехода с помощью функции. Функция принимает значения от 0 до 1.

Функция перехода (timing function)



Шаги - **steps()**

Функция	График	Описание
steps(3)		Пауза возникает при запуске анимации. (Эта функция соответствует steps(3, end), т.к. значение end используется по умолчанию.)
steps(3, end)		Пауза возникает при запуске анимации.
steps(3, start)		Анимация запускается сразу, пауза появляется в конце.

Пример

http://shpargalkablog.ru/2011/07/transformaciya-css.html

Анимации. Определение

This CSS module describes a way for authors to animate the values of CSS properties over time, using keyframes.

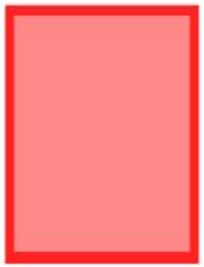
W3C

Манипулируйте значениями свойств, используя **кейфреймы**.

Описывайте поведение анимаций, управляя их продолжительностью, кол-вом повторений и конечным состоянием.

Как это работает

```
<div></div>
```



```
01. div{
02.  width: 100px;
03. }
04. div:hover{
05.  width: 200px;
06. }
```

Как это работает с анимацией

```
<div></div>
```

```
01. div{
02.   ...
03. }
04. div:hover{
05.   animation: boo 1s;
06. }
```

С помощью свойства **@keyframes** Вы можете создать анимацию.

from (*оформление элемента в начале анимации*)

to (*оформление элемента в конце анимации*)

0% - оформление элемента в начале анимации,

100% - оформление в конце анимации

Кейфреймы

```
@keyframes boo{
 from{
  transform: translateX(0px) rotate(0deg);
 }
 50%{
  transform: translateX(100px) rotate(90deg);
 }
 to{
  transform: translateX(0px) rotate(180deg);
 }
}
```

Свойство *animation*

- animation-name: boo; - имя
- animation-duration: 0.5s; - длительность
- animation-delay: 0.3s; - задержка
- animation-timing-function: linear; - плавность изменения - описывает метод расчета промежуточных значений свойств для анимации

Свойство *animation*

- animation-iteration-count: 5; - установить количество повторов анимации

infinite — анимация проигрывается бесконечное число раз

- animation-direction: normal | reverse | alternate; - порядок выполнения анимации (анимация должна выполняться в обратном порядке в четные разы и в нормальном в нечетные)

Свойство *animation*

- animation-play-state: running | paused; - запустить или приостановить анимацию

```
#spinner:hover {
  -webkit-animation-play-state: paused;
}
```

- animation-fill-mode: forwards | backwards | both; - определяет, будет ли видимым эффект от анимации, когда сама анимация уже закончилась (можно указать несколько через запятую).

Допустимые значения

- none — эффект от анимации будет наблюдаться лишь в течение времени анимации
- forwards — эффект от анимации будет виден даже тогда, когда анимация закончилась (после завершения анимации элемент получает не начальные значения, а какие-то промежуточные из анимации)
- backwards — начальный ключевой кадр анимации будет отображаться в течение всего времени задержки анимации (имеет смысл применять только к анимациям с ненулевой задержкой)
- both — начальный ключевой кадр анимации будет отображаться в течение всего времени задержки анимации, а последний будет отображаться даже тогда, когда анимация закончилась (имеет смысл применять только к анимациям с ненулевой задержкой

Доп. Свойства css3

- Градиент
- -webkit-linear-gradient -Создаёт линейный градиент в браузерах Safari и Chrome.
- background-image: -webkit-linear-gradient([<угол> | <позиция> ,] <цвет> [, <цвет>]*);

Позиция	Угол	Описание	Вид
top	270deg	Сверху вниз.	
left	0deg	Слева направо.	
bottom	90deg	Снизу вверх.	
right	180deg	Справа налево.	
top left	-45deg	От левого верхнего угла к правому нижнему.	
top right	225deg	От правого верхнего угла к левому нижнему.	
bottom left	45deg	От левого нижнего угла к правому верхнему.	
bottom right	-225deg	От правого нижнего угла к левому верхнему.	

Пример

- **   **
- **.grad{**
  background-image: -webkit-linear-gradient(0, pink, blueviolet, blue, green, yellow, orange, red);
  background-image: -moz-linear-gradient(0, pink, blueviolet, blue, green, yellow, orange, red);}

CSS фильтры

- Filter - Устанавливает фильтр (визуальный эффект) или их сочетание для элемента. К фильтрам относится изменение прозрачности, добавление тени, трансформация и др.

Фильтры

Alpha	Настраивает прозрачность объекта.
BasicImage	Устанавливает параметры цвета, поворота изображения или прозрачности.
Blur	Размывает содержимое.
Chroma	Показывает определенные цвета прозрачными.
DropShadow	Отображает тень.
Emboss	Показывает содержимое объекта в виде барельефа.
Engrave	Показывает содержимое объекта в виде черно-белой гравюры.
Glow	Добавляет свечение вокруг краев.
Gradient	Создаёт линейный градиент.
ICMFilter	Преобразует цвета содержимого на основе профиля системы управления цветом (Image Color Management, ICM).
Light	Создает эффект лучей света.
MaskFilter	Показывает прозрачные пикселы как цветную маску, а непрозрачные пикселы наоборот, прозрачными.
Matrix	Изменяет размер, поворачивает или отражает объект на основе матричных преобразований.
MotionBlur	Размывает объект так, словно он быстро движется.
Shadow	Добавляет тень.
Wave	Вносит волнообразные искажения.

```css
  .fl:hover{
    -webkit-filter: blur(5px) ;
}
.fl1:hover{
    -webkit-filter:invert(100%);
}
.fl2:hover{
    -webkit-filter:hue-rotate(165deg);
}
.fl3{
    -webkit-filter:grayscale(100%);
    transition: 1.5s;
}
.fl3:hover {
    -webkit-filter: none;
    filter: none; /* Убираем фильтр */
```

```html
<div class="grad">  </div>
<div class="grad2">  </div>
<img src="50.jpg" class="fl">
<img src="50_0.jpg" class="fl1">
<img src="50_1.jpg" class="fl2">
```

Машинка

1. Создаем блок для показа анимации

Html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<div id="anim">



</div>
</body>
</html>
```

css

```
#anim{
    width:1000px;
    height:300px;
    margin:0 auto;
    border:1px solid #069;
    background:url(city2.png) repeat-x 0 100%;
    background-size:100%;
     animation: city 7s linear;
}
```

Анимация для фона.

```
@keyframes city {
    from { background-position: 1000px 100%, 0 0;}
    to { background-position: 0 100%, 0 0; }
}
```

Создать блоки для машинки, колес и дыма

```
<div id="anim">
    <div id="carr">
        <img src="машина2.png" />
        <div id="kol">
            <img src="колесо.png" width="35px;"/>
        </div>
        <div id="kol2">
            <img src="колесо.png" width="35px;"/>
        </div>
        <div id="gaz">
            <img src="gaz_1.png" width="30px"/>
        </div>
        <div id="gaz2">
            <img src="gaz_1.png" width="20px"/>
        </div>
    </div>
</div>
```

У нас должны быть картинка машинки картинка колеса и картинка дыма

Результат

Позиционируем машинку и добавляем анимацию перемещения по оси x

```
#carr{
    transform:translate(40px,180px);
    position:relative;
    z-index:1;
    animation: go 7s linear ;
    animation-fill-mode: both;
}
```

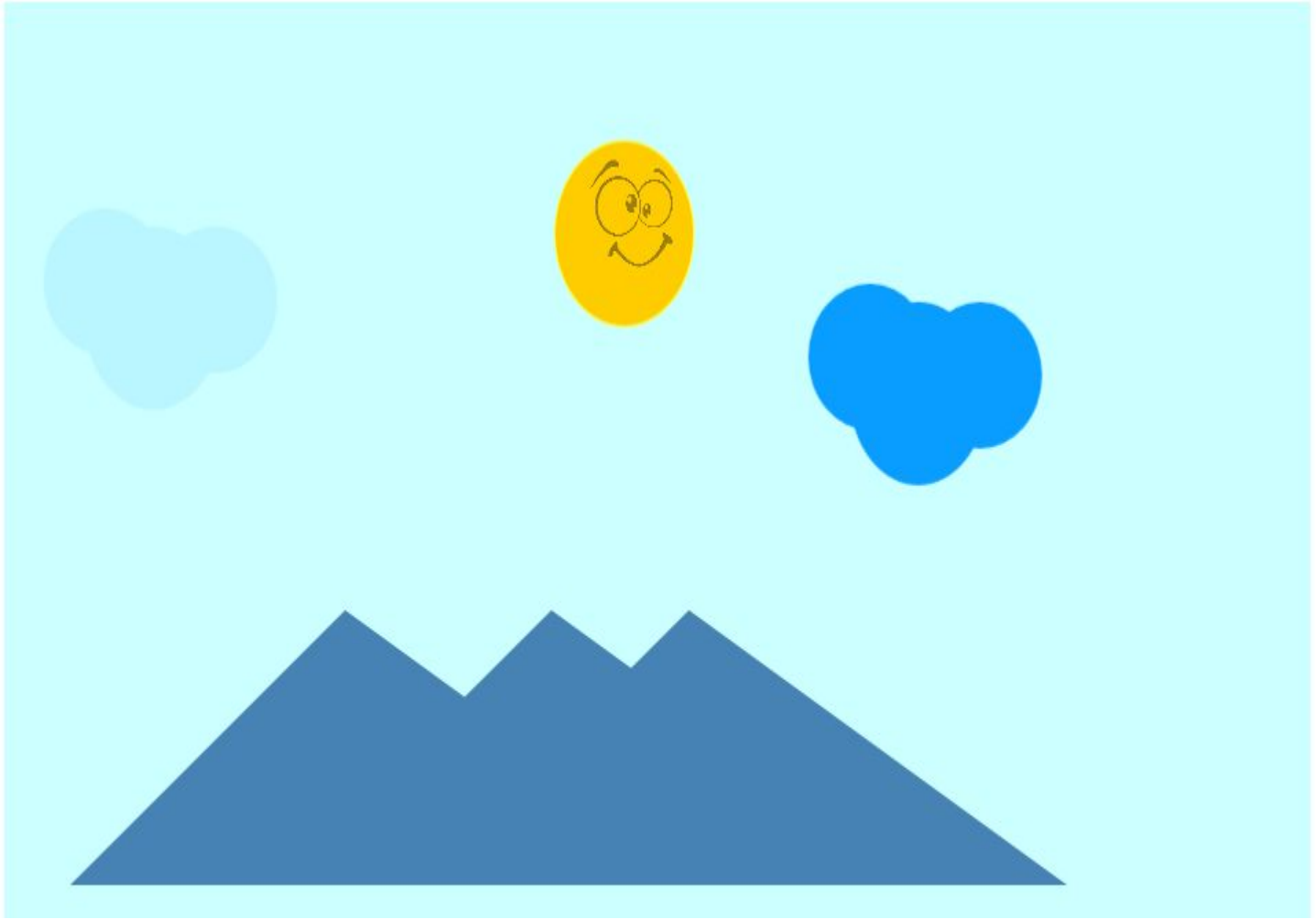
```
@keyframes go{
    0%{
        transform:translate(40px,180px);
    }
    100%{
        transform:translate(800px,180px);
        }
            }
```

Позиционируем колеса и дым

```
#kol,#kol2{
    position:absolute;
    z-index:2;
}
#kol{
    transform:translate(20px,-22px);
    animation:kryg1 7s linear ;
}
#kol2{
    transform:translate(150px,-25px);
    animation:kryg2 7s linear ;
}
```

Дальше сами…

Галерея

Кубик

Задание