ЗАДАНИЕ № 21. СОЗДАНИЕ ОБЪЕМНОЙ ФИГУРЫ

Сразу уточним, что создание сложных 3d моделей – это задача скриптов и специализированного ПО, которое создает такие модели, а мы их в свои страницы внедряем.

В этой работе поближе познакомимся с кубиком, который присутствовал в примерах кода.

Кубик будет состоять из шести блоков. Часть настроек для них будет общей, например, размеры, размер и цвет текста, значение свойства display. А некоторые будут отличаться: цвет и поворот по одной из осей.

Начнем с верстки.

1. Воспроизведем следующую структуру документа. Обратите внимание, что у каждой грани есть как общий класс **face**, так и свой собственный, указывающий на сторону.

```
<div class="container">
    <div id="example-element" class="r3d-1">
        <div class="face front">1</div>
        <div class="face back">2</div>
        <div class="face right">3</div>
        <div class="face left">4</div>
        <div class="face top">5</div>
        <div class="face bottom">6</div>
    </div>
</div>
```

2. Для обертки кубика зададим стандартные настройки позиционирования.

```
.container {
    width: 90%;
    margin: 0 auto;
    margin-bottom: 3%;
```

```
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
}
```

3. Теперь сам куб. Обратите внимание на свойство **transform-style**, которое позволит сделать наши блоки объемными и **rotate-3d** для разворота, чтобы оценить их расположение.

```
#example-element {
    margin: 50px;
    width: 200px;
    height: 200px;
    transform-style: preserve-3d;
    transform: rotate3d(1, 1, 0, 20deg);
}
```

4. Как уже было сказано, зададим общие для всех граней настройки. Обратите внимание, на ширину и высоту каждой грани, они занимают 100% от родителя, то есть в нашем случае 200px.

```
.face {
    display: flex;
    align-items: center;
    justify-content: center;
    width: 100%;
    height: 100%;
    position: absolute;
    backface-visibility: inherit;
    font-size: 5em;
    color: #fff;
}
```

5. Перейдем к настройке граней. Для каждой из них с помощью свойства **transform:translate();** зададим значение **100px**. Это отодвинет наши грани ровно на половину их длины (после завершения работ вы можете поэкспериментировать со значением этого свойства и размерами родительского контейнера).

```
.front {
    transform: translateZ(100px);
}

.back {
    transform: translateZ(100px);
}

.right {
    transform: translateZ(100px);
}

.left {
    transform: translateZ(100px);
}

.top {
    transform: translateZ(100px);
}

.bottom {
    transform: translateZ(100px);
}
```

6. Следующий шаг – разворот граней в пространстве. Стоящую перед нам грань .front разворачивать не нужно. А вот правую, левую и заднюю повернем с помощью свойства transform:rotateY();
   Обратите внимание, что обе настройки записываются внутри одного свойства transform без запятых, просто через пробел.

```
.back {
    transform: rotateY(180deg) translateZ(100px);
}

.right {
    transform: rotateY(90deg) translateZ(100px);
}
```

```
.left {
    transform: rotateY(-90deg) translateZ(100px);
}
```

7. Верхнюю и нижнюю грани повернём с помощью transform:rotateX();

```
.top {
    transform: rotateX(90deg) translateZ(100px);
}

.bottom {
    transform: rotateX(-90deg) translateZ(100px);
}
```

8. Самостоятельно задайте граням цвет, но не забудьте указать прозрачность, чтобы оценить получившуюся фигуру.