**Неделя 5. Анимация CSS. Transition, свойство Transform, CSS правило Кey Frames и библиотека Animate CSS.**

Сегодня не один сайт не обходится без анимации, даже какой-то минимальной. Анимация добавляет некоторую интерактивность сайту, что в свою очередь дает возможность удержать клиента на Вашем сайте дольше, чем на похожем статическом.

На данный момент существует довольно много библиотек, с помощью которых можно сделать анимацию сайта. Но все-таки есть смысл посмотреть на различные инструменты, которые помогают реализовать интерактивность без использования javaScript или библиотеки jQuery.

**Transition**

CSS переходы (или Transition) используются для оживления верстки и могут применяться ко всем элементам на странице, даже к псевдоклассам. Его можно использовать для построения несложных сценариев анимации. Верстка, в которой используются элементы анимации выглядит более дорого, нежели статика.

Добавим на страницу блок, из которого сделаем что-то вроде кнопки ☺

<div class="animate">Click Me</div>

Добавим стили для кнопки.

.animate

{

    width: 250px;

    border-radius: 10px;

    margin: 30px auto;

    background-color: brown;

    text-align: center;

    line-height: 60px;

    padding: 0px 30px;

    font-size: 20px;

    font-weight: bold;

    box-shadow: 0 7px 10px rgb(184, 94, 94);

    position: relative;

}

Первое свойство, которое мы рассмотрим – transition-duration. Как Вы уже догадались, оно определяет время работы прописанной для объекта анимации, за которое будет меняться значение какого-либо CSS свойства.

Может измеряться в секундах или миллисекундах.

Добавим для дива небольшой эффект при наведении, который меняет фон.

.animate:hover{

    background: rgb(214, 124, 20);

    box-shadow: 0 4px 10px rgb(184, 94, 94);

    top: 3px;

}

В данном случае анимации не будет вообще, потому, что не указаны свойства для анимации и нет никакой длительности.

Добавим пару свойств к классу animate. В итоге, все свойства, которые мы переопределяем в псевдоклассе hover будут исполняться плавно.

    transition-duration: 1s;

Можно немного поменять длительность анимации, и мы увидим немного другую картину. ПОСТАВИТЬ 5 секунд.

Также здесь можно указывать несколько значений длительности, через запятую. Но это можно сделать только, когда указаны несколько свойств, которые мы хотим анимировать. Для этого используется свойство transition-property.

**Transition-property**

Добавим к стилям кнопки ЦВЕТ ТЕКСТА.

Добавим данное свойство пока только для фона.

transition-property: background-color;

transition-duration: 1s;

ВИДНО, что анимируется только фон, а, например, цвет текста изменяется без анимации.

А если я добавлю к нашему transition-property еще и цвет текста, то анимация уже будет происходить для двух свойств.

transition-property: background-color, color;

transition-duration: 1s;

И сейчас мы сможем задать время анимации для каждого из свойств.

    transition-property: background-color, color;

    transition-duration: 1s, 5s;

Надо помнить, что не все свойства поддаются анимации, например, display: none.

Следующее свойство

**Transition-delay**

Оно задает задержку начала анимации. Часто используется для создания простых сценариев анимации.

Например, я хочу чтобы сначала изменился фон кнопки, а уже потом изменился цвет текста.

Для этого зададим такие параметры

    transition-duration: 0.5s, 1s;

    transition-delay: 0s,2s;

Видно, что свойства анимируются с небольшой задержкой. Также, при снятии наведения все происходит в обратном порядке, но длительности все выдерживаются.

Следующее свойство

**Transition-timing-function**

Задает стиль или вид перехода. Т.е. как будет происходить анимация.

Здесь также можно указывать для каждого свойства свой тип перехода, как и для свойства transition-property.

По умолчанию, используется значение ease.

transition-timing-function: ease;

Для демонстрации данных свойств, я добавлю немного кода на страницу.

    <div id="wraper">

        <div class="button btn1">ease</div>

        <div class="button btn2">ease-in</div>

        <div class="button btn3">ease-out</div>

        <div class="button btn4">ease-in-out</div>

        <div class="button btn5">linear</div>

        <div class="button btn6">steps</div>

        <div class="button btn7">cubic-bezier</div>

    </div>

И стили для этого кода

#wraper{

    width: 600px;

    background-color: rgb(216, 210, 210);

    padding: 30px;

}

.button{

    width: 200px;

    height: 50px;

    color: white;

    letter-spacing: 3px;

    text-transform: uppercase;

    margin-bottom: 20px;

    margin-left: 20px;

    text-align: center;

    padding-top: 15px;

    border-radius: 5px;

    transition: all 2s ease 0s;

}

.btn1{

    background-color: green;

    transition-timing-function: ease;

}

.btn2{

    background-color: rgb(77, 135, 223);

    transition-timing-function: ease-in;

}

.btn3{

    background-color: rgb(245, 146, 33);

    transition-timing-function: ease-out;

}

.btn4{

    background-color: green;

    transition-timing-function: ease-in-out;

}

.btn5{

    background-color: rgb(77, 135, 223);

    transition-timing-function: linear;

}

.btn6{

    background-color: rgb(245, 146, 33);

    transition-timing-function: steps(3);

}

.btn7{

    background-color: rgb(77, 135, 223);

    transition-timing-function: cubic-bezier(0.5, -0.3, 0.4, 1);

}

.button:hover{

    width: 500px;

    color: black;

}

Все вышеуказанные свойства мы можем записать с помощью одного свойства

**Transition**

Если я хочу, чтобы сразу все свойства анимировались одинаково, то можно написать так:

transition: all 2s ease 0s;

Если мы хотим для разных свойств задать различные параметры анимации, то это будет выглядеть следующим образом:

transition: color 2s ease-out 0.5s, width 1s ease 0s;

**ВАЖНО. Мы также можем управлять переходами и при наведении мыши на объект, и когда мышь с объекта убираем. Для этого необходимо прописать свойство transition и для объекта без псевдокласса hover и в случае, когда мы наводим мышь, т.е. с псевдоклассом hover.**

Далее перейдем к свойству transform 2D и 3D.

С помощью этих свойств можно двигать, переворачивать в разные стороны, масштабировать элементы на странице, все это проделывать с анимацией и так далее.

Основное свойство из данного набора свойств – это свойство Transform,

**ВАЖНО. Оно применяется только к блочным элементам.**

Если Вы хотите применить его к строчному элементу, то перед этим необходимо заменить значение свойства display: inline на display: inline-block.

Трансформируемые объекты никак не влияют на окружение. При этом к ним можно применять различные виды анимации.

**Свойство Translate (x,y)**

Позволяет сдвигать элемент на новое место относительно его текущего местоположения. Можно двигать вверх, вниз, вправо, влево. Можно использовать отрицательные значения.

Для примера сделаем простую разметку

<div class="translate">Translate</div>

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Illo neque odio nisi beatae doloremque, quod possimus natus nemo repudiandae amet officia necessitatibus error sint, eaque, quae quos dolorum quam inventore.</p>

И добавим немного стилей

.translate{

    width: 100px;

    height: 100px;

    margin: 50px auto;

    background-color: brown;

    color: white;

    text-transform: uppercase;

    padding: 40px 5px;

}

Попробуем сдвинуть объект на 50px вниз и вправо. Для этого добавим строку

transform: translate(50px, 50px);

Четко видно, что остальные элементы, а именно текст снизу никак не меняются и не сдвигаются. Похожую вещь можно сделать и с помощью позиционирования.

position: relative;

    top: 80px;

    left: 100px;

При этом нам придется возвращать объект обратно, чтобы поставить его на середину, например, с помощью отрицательного марджина.

Но также мы можем сделать это с помощью свойства translate при этом используя процентные единицы.

Для этого укажем проценты вместо пикселей

position: relative;

    left: 50%;

    transform: translate(-50%, 0px);

При этом объект сдвинется влево ровно на половину ширины самого объекта, а не на ширину родительского контейнера, которая используется при работе с позиционированием.

Это свойство можно **анимировать**. Для этого используем такой код.

.translate{

    width: 200px;

    height: 50px;

    background-color: brown;

    color: white;

    text-transform: uppercase;

    text-align: center;

    margin: auto;

    padding: 40px 5px;

    transition: all 0.5s ease;

}

.translate:hover{

    transform: translate(0px, 5px);

}

**Scale (x,y)**

Производит масштабирование объекта. По умолчанию значение x и y равны 1. Первое значение – масштаб по ширине, второе – по высоте.

Если убрать одно из значений, то масштабирование будет сделано сразу для двух значений.

Есть более интересные возможности данного свойства. Мы можем, например, зеркально отобразить объект.

transform: scale(-1,1);

Если значения сделать наоборот, то вращение будет по вертикали.

**Rotate (угол в градусах)**

transform: rotate(45deg);

180градусов – развернет объект. Полный оборот 360 градусов. При использовании этих свойств на сайте вместе с анимацией, то можно придумать различные интересные штуки.

**Scew (x-угол в градусах, y-угол в градусах)**

Данное свойство деформирует объект, т.е. искажает его относительно его текущих координат. Если указывать одно значение, то второе будет рассчитываться автоматом.

transform: skew(30deg, 0deg);

**Matrix (a,b,c,d,x,y)**

По факту просто объединяет все вышеописанные свойства. Т.е. это более быстрая запись.

transform: translate(30px,0) scale(1,2) rotate(180deg);

Также можно использовать свойства только для одной оси, например ScaleX

Можно использовать так называемую множественную трансформацию.

transform: translate(30px,0) scale(1,2) rotate(180deg);

И последнее свойство для 2d трансформации.

**Transform-origin**

Это свойство позволяет изменить центр трансформации. По умолчанию он находится в центре объекта. Значения могут быть выставлены как в цифровом виде, так и словами top, left, bottom, right.

Например,

transform-origin: top left; для объекта

transform: rotate(45deg); для Ховера

**3D Трансформация**

Также можно использовать трехмерную трансформацию. К значениям по ширине и высоте просто будет добавлено еще одно измерение. И будет возможность изменять глубину объекта как вглубь, так и в сторону пользователя. Здесь используется значение перспективы (расстояние от объекта до глаз смотрящего).

Более подробно про 3D трансформацию Вы можете почитать и посмотреть примеры на <https://html5book.ru/3d-transform/>. Там есть интересные варианты применения анимации.

**Animation и Key Frames**

Далее рассмотрим свойство animation и ключевые кадры Key Frames.

Как и предыдущие свойства, свойство animation помогает сделать верстку более динамичной и улучшить взаимодействие с пользователем.

Отличие от эффектов перехода состоит в том, что для свойства animation используется набор ключевых кадров, которые позволяют воспроизводить анимацию на протяжении определенного времени, а также приостанавливать. Т.е. мы можем создавать более сложные сценарии анимации в наших проектах.

В кадрах прописываются начальные и конечные свойства, которые будут применяться в ходе анимации.

Примеры использования данных свойств Вы можете найти по следующим ссылкам:

<https://www.w3schools.com/css/css3_transitions.asp>

<https://www.w3schools.com/css/css3_animations.asp>

**Animate.css**

Библиотек «animate.css» позволяет использовать уже готовые варианты эффектов для любых элементов на странице. На сайте <https://animate.style/> подробно описан процесс применения этой библиотеки. Для ее использования Вы можете скачать весь код самой библиотеки себе в проект, либо использовать ссылки на CDN. Также можно добавить этот пакет в качестве зависимости, если Вы работаете в каком-либо фреймворке, напрмер, React.js, Angular.js или Vue.js.

Для того, чтобы анимация применялась к объектам только тогда, когда объекты появятся на экране после скрола, мы можем использовать простой скрипт:

$(document).ready(function(){

  $(window).scroll(function(){

    $('.heading1').each(function(){

      var imagePos = $(this).offset().top;

      var topOfWindow = $(window).scrollTop();

        if(imagePos < topOfWindow + 850)

        {

          $(this).addClass('fadeInLeft');

        }

    });

  });