Практическая работа 1
Тема работы: «Реализация анимированных эффектов средствами CSS»

1. Цель работы

Формирование умений и навыков создания анимационных эффектов средствами CSS в веб-документе.

2. Задание

Реализовать анимационные эффекты в соответствии с порядком выполнения работы.

3. Оснащение работы

ПК, браузер, редактор исходного кода.

4. Основные теоретические сведения

Анимация – это изменения в движении. Элементы на странице могут менять положение, прозрачность, форму, размеры и так далее. Выделяют функциональную и декоративную анимации.

Функциональная анимация помогает взаимодействовать с интерфейсом, делая его более интуитивным. Такая анимация применяется в интерфейсах продуктов и мобильных приложений.

Декоративная анимация применяется в основном на лендингах или в спецпроектах и служит для привлечения внимания зрителя, делает проект интереснее и помогает вызвать определенные эмоции.

12 принципов анимации – набор основных принципов мультипликации, предложенных аниматорами студии Дисней Олли Джонстоном и Фрэнком Томасом в их совместной работе «Иллюзия жизни: анимация Диснея». Данные принципы основаны на многолетнем опыте работы художников-мультипликаторов студии Уолта Диснея, которые, начиная с 1930 года, разрабатывали методы для получения более выразительной анимации:

- принцип 1. Сжатие и растяжение;
- принцип 2. Подготовка, или упреждение;
- принцип 3. Сценичность (постоянный учёт того, как видит образ зритель);
- принцип 4. Использование компоновок и прямого фазованного движения;
- принцип 5. Сквозное движение (или доводка) и захлёст действия;
- принцип 6. Смягчение начала и завершения движения (спэйсинг);
- принцип 7. Дуги;
- принцип 8. Дополнительное действие (выразительная деталь);
- принцип 9. Расчёт времени (хронометраж);
- принцип 10. Преувеличение, утрирование;
- принцип 11. «Цельный» (профессиональный) рисунок;
- принцип 12. Привлекательность.

Основной идеей принципов является создание иллюзии соблюдения основных законов физики, однако они рассматривают и более абстрактные вопросы, такие как эмоциональность и привлекательность персонажей.

CSS3 позволяет реализовать анимацию большинства HTML элементов без использования JavaScript.

Переходы CSS transition позволяют сделать изменения CSS-свойств плавно и в течение некоторого времени, предоставляя тем самым возможность контролировать процесс перехода элемента от одного состояния к другому.

Управление переходами реализуется посредством следующих свойств:

transition-property – указывает список свойств, которые будут анимироваться; свойства, которые здесь не указаны, будут изменяться обычным образом. Можно анимировать все свойства для конкретного элемента, указав значение all (используется по умолчанию);

transition-duration – задаёт значение продолжительности анимации, время можно указывать в секундах или миллисекундах;

transition-timing-function – временная функция, указывает точки ускорения и замедления за определенный период времени для контроля изменения скорости анимации. Чтобы самостоятельно задать процесс анимации можно использовать кривые Безье;

transition-delay – задаёт задержку времени до начала анимации, можно указывать в секундах или миллисекундах;

transition – это общее свойство, которое позволяет перечислить первые четыре свойства в порядке: property, duration, timing-function, delay.

```
<body>
  <div class="one"></div>
</body>

  .one {
    width: 200px;
    height: 200px;
    background-color: darkgreen;
    border-radius: 50%;
    border: 10px solid lawngreen;
    transition: all 3s linear;
  }

  .one:hover {
    background-color: lawngreen;
    border: 10px solid darkgreen;
    box-shadow:0px 0px 10px 10px rgba(108, 107, 107, 0.5);
  }
```

CSS-анимация позволяет анимировать свойства CSS от одного значения к другому. В то время как переходы позволяют выполнять отдельные движения, анимации позволяют создавать более сложные сценарии анимаций.

CSS-анимации состоят из двух компонентов: стилевое описание анимации и набор ключевых кадров, определяющих начальное, конечное и возможные промежуточные состояния анимируемых стилей.

Для управления CSS animation анимациями предусмотрены следующие свойства:

animation-name – здесь указвается имя анимации, которое связывает правило @keyframes с селектором;

animation-iteration-count – задаёт количество повторов анимации, значение по умолчанию 1. Значение infinite означает, что анимация будет проигрываться бесконечно;

animation-direction – задаёт направление анимации;

animation-play-state – данное свойство управляет остановкой и прогрыванием анимации. Есть два значения, running (анимация проигрывается, по умолчанию) и paused (останавливает анимацию);

animation-fill-mode – устанавливает, какие CSS-свойства будут приминены к объекту до или после анимации. Может принимать такие значения:

none – анимируемые CSS-свойства применятся к объекту только во время воспроизведения анимации, по окончании объект возвращается в исходное состояние;

forwards – анимируемые CSS-свойства применятся к объекту по окончании воспроизведения анимации;

backwards – анимируемые CSS-свойства применятся к объекту до начала воспроизведения анимации;

both – анимируемые CSS-свойства применятся к объекту и до начала, и после окончания воспроизведения анимации;

animation – общее свойство анимации, которое позволяет перечислить свойства в следующем порядке:

```
animation-name: ;
animation-duration: ;
animation-timing-function: ;
animation-delay: ;
animation-direction: ;
animation-iteration-count: ;
animation-fill-mode: ;
animation-play-state: running;
```

Свойства **animation-duration, animation-timing-function, animation-delay** работают так же, как аналогичные свойства в CSS transition.

```
<body>
  <div class="one"></div>
</body>

div {
  position: relative;
  width: 100px;
  height: 100px;
```

```
  background-color: aquamarine;
  border-radius: 50%;
  animation: ball 10s infinite cubic-bezier(.01,.66,1,.22);
 }

 @keyframes ball {
  from {top: 0; left: 0;}
  25% {top: 0; left: calc(100% - 100px); background-color:aquamarine;}
  50% {top: 200px; left: calc(100% - 100px); background-color: blanchedalmond;}
  75% {top: 200px; left: 0; background-color: burlywood;}
  to {top: 0; left: 0;}
  }
```

Множественные анимации могут быть объявлены в селекторе с использованием значений через запятую.

```
 <body>
  <div class="one"></div>
 </body>

 div {
  position: relative;
  width: 100px;
  height: 100px;
  background-color: aquamarine;
  border-radius: 50%;
   animation: move 10s infinite cubic-bezier(.01,.66,1,.22), color 10s infinite cubic-bezier(.01,.66,1,.22) ;
 }

 @keyframes move {
  from {top: 0; left: 0;}
  25% {top: 0; left: calc(100% - 100px);}
  50% {top: 200px; left: calc(100% - 100px);}
  75% {top: 200px; left: 0;}
  to {top: 0; left: 0;}
  }

  @keyframes color {
   from {background-color: aquamarine;}
   25% {background-color:rgb(127, 136, 255);}
   50% {background-color: blanchedalmond;}
   75% {background-color: burlywood;}
   to {background-color: aquamarine}
   }
```

CSS3-трансформации позволяют сдвигать, поворачивать и масштабировать элементы. Трансформации преобразовывают элемент, не затрагивая остальные элементы веб-страницы, т.е. другие элементы не сдвигаются относительно него.

CSS свойства для трансформации:

transform – определяет, какая функция будет применяться (translate, rotate, scale и др.);

transform-origin – позволяет изменять точку начала преобразования (работает как background-position);

transform-style – для настройки 3D.

Свойство transform **не имеет сокращённой записи**. Трансформируемые элементы не влияют на поток: независимо от вращения, масштабирования или перемещения, они не будут влиять на другие элементы.

Существует два вида css-трансформаций: 2D-трансформации на плоскости и 3D-трансформации в пространстве. Осуществляются такие трансформации при помощи анимируемого ненаследуемого css-свойства transform.

Свойство transform может принимать следующие значения-функции, которые можно условно разбить на несколько групп:

1. Функции перемещения:

translate() – сдвигает элемент на плоскости вдоль осей X и Y;

translateX() – сдвигает элемент вдоль оси X;

translateY() – сдвигает элемент вдоль оси Y;

translateZ() – сдвигает элемент вдоль оси Z;

translate3d() – сдвигает элемент в трехмерном пространстве.

Функция может принимать:

– один параметр: перемещает элемент вдоль оси x;

– два параметра: первое значение для оси x, второе для оси y.

Когда нужно сместить элемент вдоль конкретной оси, можно применить соответствующие функции трансформации:

translateX(X), translateY(Y), translateZ(Z)

Сместить элемент по всем трём осям можно при помощи функции:

translate3d(X, Y, Z)

```
.element {
transform: translate(100px);
}
.element {
transform: translate(-150px, 150px);
}
```

2. Функции масштабирования:

scale() – масштабирует элемент на плоскости;

scaleX() – масштабирует элемент вдоль оси X;

scaleY() – масштабирует элемент вдоль оси Y;
scaleZ() – масштабирует элемент вдоль оси Z;
scale3d() – масштабирует элемент в трехмерном пространстве.
Функция может принимать:
– один параметр: изменение размеров элемента одинаково по высоте и ширине;
– два параметра: первое значение изменяет размер элемента по горизонтали, второе по вертикали.

Когда необходимо растягивать или сжимать элемент только по горизонтали, вертикали или третьей оси Z можно воспользоваться функциями:

scaleX(X), scaleY(Y), scaleZ(Z)

Если нужно масштабировать элемент по всем трём осям, можно использовать функцию:

scale3d(X, Y, Z)

Диапазон возможных значений:
1: элемент сохраняет свой первоначальный размер;
2: элемент удваивается в размере;
0.5: элемент уменьшается наполовину;
0: элемент в основном исчезает (так как его высота и ширина равны нулю);
-1: элемент отражается.

```
.element {
transform: scale(1.5, 0.8);
}
.element {
transform: scale(-1.3, 1.3);
}
.element {
transform: scale(-0.9);
}
```

3. Функции поворота:
rotate() – поворачивает элемент на требуемый угол на плоскости относительно точки трансформации, которая задается свойством transform-origin;
rotateX() – поворачивает элемент на требуемый угол относительно оси X;
rotateY() – поворачивает элемент на требуемый угол относительно оси Y;
rotateZ() – поворачивает элемент на требуемый угол относительно оси Z;
rotate3d() – поворачивает элемент в трехмерном пространстве.
По умолчанию, вращение происходит вокруг центра элемента. В функцию передаём единицы измерения углов (deg, rad, turn), например 45deg или 0.5turn. Обычное вращение элемента на странице – это вращение относительно оси Z. Если необходимо

вращать элемент относительно других осей, то нужно не забывать про перспективу. С ней повороты относительно X или Y будут выглядеть максимально естественно.

Функция аналогична rotateZ(Z). Чтобы не запоминать ось для типового вращения элемента, можно использовать просто слово rotate:

rotate(Z)

Если нужно повернуть элемент по всем трём осям, можно использовать функцию:

rotate3d(X, Y, Z)

```
.element {
transform: rotate(-10.5deg);
}
.element {
transform: rotate(0.11rad);
}
.element {
transform: rotate(10grad);
}
.element {
transform: rotate(-0.5turn);
}
```

4. Функции наклона:
skew() – наклоняет элемент на заданный угол на плоскости;
skewX() – наклоняет элемент вдоль оси X;
skewY() – наклоняет элемент вдоль оси Y.

Функция позволяет искажать элемент, сдвигая его стороны вдоль линий (skewX сдвигает верхнюю сторону элемента относительно нижней, skewY – правую сторону относительно левой). Может принимать:

– один параметр: элемент искажается по горизонтали;

– два параметра: первое значение искажает элемент по горизонтали, второе по вертикали.

В функцию разрешается передавать любые доступные в CSS единицы измерения углов: градусы, грады, радианы или же просто обороты (например, skew(2.5turn, -30deg)). При этом вместо нулевых значений допустимо использовать просто ноль.

```
.element {
transform: skew(-0.5rad);
}
```

5. Прочие функции:

matrix() – задает двумерную матрицу преобразований, объединяя несколько функций в одной;

matrix3d() – задает трехмерную матрицу преобразований.

Функция использует матричные преобразования и позволяет описать любую трансформацию в плоскости экрана, позволяет писать сложные динамические анимации:

matrix(0.707107, 0.707107, -0.707107, 0.707107, -0.707107, 34.6482)
аналогична
rotate(45deg) translate(24px, 25px)

Популярные JS-библиотеки для анимации «под капотом» используют как раз матричные преобразования, а не конкретные функции трансформации.

Если необходимо реализовать трансформации в трёхмерном пространстве, а не в плоскости экрана, то нужно использовать функцию perspective(), которая задает перспективу (глубину сцены).

Несмотря на то, что экран плоский, имеется возможность перемещать элемент вдоль оси Z. Она направлена перпендикулярно плоскости экрана в сторону пользователя. Ощущение движения к нам или от нас реализуется посредством изменения размеров элементов: элемент должен становиться крупнее или мельче. Стул, который стоит рядом с нами, будет визуально крупнее, чем стул, стоящий в конце комнаты. Эта разница в размерах – следствие перспективы. Элемент на экране может вести себя подобно объектам в реальном мире и менять размер при перемещении к нам или от нас. Чтобы это заработало, нужно элементу задать свойство perspective. Это свойство необходимо применять при любых трансформациях, выходящих из плоскости экрана.

Функция perspective() принимает один параметр – расстояние до точки схождения перспективы: любые доступные в CSS единицы измерения длины (если не указаны, то по умолчанию будут использованы пиксели), при этом чем больше положительное значение, тем менее выражен эффект перспективы, и наоборот; отрицательные значения и нуль отменяют действие перспективы. Плоскость экрана принимается за начало координат. Например, запись perspective(500px) означает, что точка схождения перспективы находится как бы на расстоянии 500 пикселей вглубь от плоскости экрана.

```
.element {
transform: perspective(500px);
}
```

Значение свойства transform:none – отменяет эффекты трансформации для данного элемента.

Можно применять сразу несколько функций трансформаций:

```
.element {
  transform: translate(20px, 20px) skew(20deg);
}
```

Если среди значений есть функция perspective(), то она должна быть первой среди всех значений:

```
.element {
  transform: perspective(500px) translate3d(10px, 0, 20px) rotateY(3deg);
}
```

В последних версиях спецификации появились отдельные CSS-свойства для трансформаций. Это rotate, translate и scale.

Т.е. вместо комплексных трансформации можно применить несколько функций, описав их отдельным свойством:

```
.transform-function {
  transform: translate(100px, 100px) rotate(180deg) scale(2);
}

.individual-transforms {
  translate: 100px 100px;
  rotate: 180deg;
  scale: 2;
}
```

Оба этих способа записи выполнят одни и те же трансформации, но у них есть ряд отличий:

– использование индивидуальных свойств позволяет создавать классы-модификаторы без опасения перекрыть всё свойство transform;

– при использовании свойства transform результирующая трансформация вычисляется с учётом порядка функций. Например, transform: rotate(15deg) translateX(100px) translateY(30px) и transform: translateX(100px) translateY(30px) rotate(15deg); дадут разный конечный результат; при использовании индивидуальных свойств, результат будет одинаковым вне зависимости от порядка свойств.

Свойство **transform-origin** задаёт положение точки, относительно которой применяются трансформации. Оно задаёт координаты точки, относительно которой будет выполняться трансформация.

Для них разрешается использовать:

для координаты X – любые доступные в CSS единицы измерения длины, проценты (по умолчанию X=50%), а также ключевые слова left (эквивалентно 0% по оси X), center (эквивалентно 50% по оси X) или right (эквивалентно 100% по оси X);

для координаты Y – любые доступные в CSS единицы измерения длины, проценты (по умолчанию Y=50%), а также ключевые слова top (эквивалентно 0% по оси Y), center (эквивалентно 50% по оси Y) или bottom (эквивалентно 100% по оси Y);

для координаты Z – любые доступные в CSS единицы измерения длины, исключая проценты и ключевые слова.

```
.element {
  transform-origin: 0 0;
  transform: rotate(-30deg);
}
```

Способов записи несколько:

```
.element {
  transform-origin: 5px;
}
.element {
  transform-origin: top;
}
.element {
  transform-origin: 5px 10%;
  transform-origin: 3cm 2px;
  transform-origin: left 2px;
  transform-origin: right top;
  transform-origin: top right;
}
.element {
  transform-origin: 2px 30% 10px;
  transform-origin: left 5px -3px;
  transform-origin: right bottom 2cm;
  transform-origin: bottom right 2cm;
}
```

Два значения задаются для осей X и Y соответственно. Три значения – для X, Y и Z. Для осей X и Y можно задавать ключевые слова: top, bottom, left, right, center. Для оси Z можно задавать только числовые значения.

По умолчанию точка сходимости линий (т.е. точка расположения глаз зрителя при взгляде на экран) при использовании функции perspective () находится на оси, проходящей через центр элемента перпендикулярно плоскости экрана. Однако эту точку можно смещать, изменяя ее координаты X и Y. Делается это при помощи ненаследуемого css-свойства perspective-origin, которое должно использоваться совместно с perspective() для родительского блока и свойством transform для дочернего элемента.

В качестве значений **perspective-origin** принимает через пробел координаты X и Y точки сходимости линий. Для них разрешается использовать:

для координаты X – любые доступные в CSS единицы измерения длины, проценты (по умолчанию X=50%), а также ключевые слова left (эквивалентно 0% по оси X), center (эквивалентно 50% по оси X) или right (эквивалентно 100% по оси X);

для координаты Y – любые доступные в CSS единицы измерения длины, проценты (по умолчанию Y=50%), а также ключевые слова top (эквивалентно 0% по оси Y), center (эквивалентно 50% по оси Y) или bottom (эквивалентно 100% по оси Y).

Если передать свойству perspective-origin только одно значение, то оно будет считаться координатой X, а для координаты Y будет использовано значение по умолчанию.

Свойство **transform-style** определяет, как будут вести себя потомки элемента в 3D-пространстве при трансформации. Свойство применяется к родительскому контейнеру и определяет, как дочерние трансформированные элементы (т.е. элементы, для которых будет задано свойство transform) будут отображаться в3D-пространстве:

flat – будут трансформироваться в плоскости родителя (используется по умолчанию);

preserve-3d – будут трансформироваться в своем собственном 3D-пространстве.

При transform-style: preserve-3d каждый дочерний элемент получает независимую от родителя плоскость, к которой можно применять 3D-трансформации. При transform-style:flat существует ровно одна плоскость – плоскость родителя, и никакие трансформации не могут заставить дочерний элемент выйти из этой плоскости, т.е. дочерние элементы всегда будут лежать в плоскости родителя и не смогут пересечь его ни при каких поворотах, масштабировании и прочих условиях.

```
.parent {
  transform-style: flat; /* По умолчанию */
  transform-style: preserve-3d;
}
```

Свойство transform-style не наследуется. Нужно индивидуально задавать его для каждого следующего уровня вложенности.

Когда в результате трансформации элемента (например, поворота вокруг оси X) он поворачивается к нам обратной стороной, его обратная сторона по умолчанию остается видимой пользователю и отображается зеркально. Однако такое поведение можно легко изменить при помощи ненаследуемого css-свойства **backface-visibility**, которое определяет, будет показываться обратная сторона элемента или нет. В качестве значений свойство принимает два ключевых слова:

visible – обратная сторона элемента видна и отображается зеркально сквозь переднюю сторону элемента (используется по умолчанию);

hidden – обратная сторона элемента не видна (скрывается за передней поверхностью элемента).

5. Порядок выполнения работы

1.Используя CSS переходы выполнить анимацию индикатора загрузки в соответствии с вариантом.

Вариант	Задание

1	
2	
3	
4	
5	
6	
7	
8	

2. Средствами CSS разработать логотип по теме в соответствии с вариантом:

Вариант	**Тема**

1	HTML
2	CSS
3	JS
4	МРК
5	ПОИТ
6	ПССИП
7	№ группы
8	Редактор кода

3. Реализовать декоративную анимацию созданного логотипа.

4. Добавить в веб-документ фигуру (фигуры) в соответствии с вариантом, поместив в нее свое Имя, и реализовать:

– 2D-транформацию по своему выбору;

– множественные (не менее 3-х) 2D-трансформации по своему усмотрению двумя различными способами (2-мя);

– реализовать 3D трансформации с применением перспективы;

При реализации трансформаций, при необходимости, выполнить смещение центра трансформации.

Вариант	**Задание**
1	
2	
3	

4	
5	
6	
7	
8	

5. Реализовать перелистывание альбома.
6. Реализовать поддержку созданных эффектов в различных браузерах.

6. Контрольные вопросы и задания

1. Дайте определение анимации.
2.Перечислите основные типы анимации.
3.Какие свойства CSS используются для создания анимации?

4. Перечислите свойства для реализации CSS трансформаций.

5. Какие функции можно использовать в качестве значения свойства transform? Дайте краткую характеристику.

7. Рекомендуемая литература

1. https://www.internet-technologies.ru/articles/polnoe-rukovodstvo-po-veb-animacii.html
2. http://animation-ua.com/ru/shkola/uroki-2d-animacii/479-12-basic-principles-of-animation
3. https://www.youtube.com/watch?v=LcoDiK2a_oM
4. https://tlum.ru/news/obasnaem-12-principov-animacii-disnea-na-gifkah-s-mi-mi-miskami/
5. https://dtf.ru/howto/809857-perevod-21-princip-animacii-ot-dermota-o-konnora