Лабораторная работа №5.

Тема: Применение CSS-анимаций и переходов.

Цель работы: освоить CSS-анимации и трансформации для создания современных пользовательских интерфейсов.

Основные термины и понятия:

CSS-переходы позволяют анимировать исходное значение CSS-свойства на новое значение с течением времени, управляя скоростью смены значений свойств. Большинство свойств меняют свои значения за 16 миллисекунд, поэтому рекомендуемое время стандартного перехода – 200ms.

Смена свойств происходит при наступлении определенного события, которое описывается соответствующим псевдоклассом. Чаще всего используется псевдокласс: hover. Данный псевдокласс не работает на мобильных устройствах. Универсальным решением, работающим в настольных и мобильных браузерах, будет обработка событий с помощью JavaScript (например, переключение классов при клике).

Переходы применяются ко всем элементам, а также к псевдоэлементам :before и :after. Для задания всех свойств перехода обычно используют краткую запись свойства transition.

CSS-переходы могут применяться не ко всем свойствам и их значениям.

Свойство transition-property содержит название CSS-свойств, к которым будет применен эффект перехода. Значение свойства может содержать как одно свойство, так и список свойств через запятую. При создании перехода можно использовать как начальное, так и конечное состояние элемента.

Создаваемые эффекты должны быть ненавязчивыми. Не все свойства требуют плавного изменения во времени, что связано с пользовательским опытом. Например, при наведении на ссылку мы хотим видеть мгновенную смену цвета ссылки или цвета и стиля подчёркивания. Поэтому переходы следует использовать для тех свойств, к которым действительно нужно привлечь внимание.

```
div {
  width: 100px;
  transition-property: width;
}
div:hover {
  width: 300px;
}
```

Свойство transition-duration задаёт промежуток времени, в течение которого должен осуществляться переход. Если разные свойства имеют разные значения для перехода, они указываются через запятую. Если продолжительность перехода не указана, то анимация при смене значений свойств происходить не будет.

```
div {
  transition-duration: .2s;
}
```

Свойство transition-timing-function задаёт временную функцию, которая описывает скорость перехода объекта от одного значения к другому. Если вы определяете более одного

перехода для элемента, например, цвет фона элемента и его положение, вы можете использовать разные функции для каждого свойства.

```
div {
  transition-timing-function: linear;
}
```

Необязательное **свойство transition-delay**, позволяет сделать так, чтобы изменение свойства происходило не моментально, а с некоторой задержкой.

Краткая запись перехода

Все свойства, отвечающие за изменение внешнего вида элемента, можно объединить в одно свойство transition:

transition: transition-property transition-duration transition-timing-function transition-delay;

Плавный переход нескольких свойств

Для элемента можно задать несколько последовательных переходов, перечислив их через запятую.

```
div {transition: background 0.3s ease, color 0.2s linear;}
```

CSS-анимация придаёт веб-страницам динамичность. Она оживляет веб-страницы, улучшая взаимодействие с пользователем. В отличие от CSS-переходов, создание анимации базируется на ключевых кадрах, которые позволяют автоматически воспроизводить и повторять эффекты на протяжении заданного времени, а также останавливать анимацию внутри цикла.

CSS-анимация может применяться практически для всех html-элементов, а также для псевдоэлементов :before и :after. При создании анимации не стоит забывать о возможных проблемах с производительностью, так как на изменение некоторых свойств требуется много ресурсов.

Ключевые кадры используются для указания значений свойств анимации в различных точках анимации. Ключевые кадры определяют поведение одного цикла анимации; анимация может повторяться ноль или более раз.

Ключевые кадры указываются с помощью правила @keyframes, определяемого следующим образом:

```
@keyframes имя анимации { список правил }
```

Создание анимации начинается с установки **ключевых кадров** правила @keyframes. Кадры определяют, какие свойства на каком шаге будут анимированы. Каждый кадр может включать один или более блоков объявления из одного или более пар свойств и значений. Правило @keyframes содержит имя анимации элемента, которое связывает правило и блок объявления элемента.

```
@keyframes shadow {
  from {text-shadow: 0 0 3px black;}
  50% {text-shadow: 0 0 30px black;}
  to {text-shadow: 0 0 3px black;}
}
```

Ключевые кадры создаются с помощью ключевых слов from и to (эквивалентны значениям 0% и 100%) или с помощью процентных пунктов, которых можно задавать сколько угодно. Также можно комбинировать ключевые слова и процентные пункты. Если кадры имеют одинаковые свойства и значения, их можно объединить в одно объявление:

```
@keyframes move {
  from,
  to {
    top: 0;
    left: 0;
  }
  25%,
  75% {top: 100%;}
  50% {top: 50%;}
}
```

Если 0% или 100% кадры не указаны, то браузер пользователя создает их, используя вычисляемые (первоначально заданные) значения анимируемого свойства.

Если несколько правил @keyframes определены с одним и тем же именем, сработает последнее в порядке документа, а все предыдущие проигнорируются.

После объявления правила @keyframes, можно ссылаться на него в свойстве animation:

```
h1 {
  font-size: 3.5em;
  color: darkmagenta;
  animation: shadow 2s infinite ease-in-out;
}
```

Не рекомендуется анимировать нечисловые значения (за редким исключением), так как результат в браузере может быть непредсказуемым. Также не следует создавать ключевые кадры для значений свойств, не имеющих средней точки, например, для значений свойства color: pink и color: #ffffff, width: auto и width: 100px или border-radius: 0 и border-radius: 50% (в этом случае правильно будет указать border-radius: 0%).

Правило стиля ключевого кадра также может объявлять временную функцию, которая должна использоваться при перемещении анимации к следующему ключевому кадру.

Продолжительность анимации: **свойство animation-duration**

Свойство animation-duration определяет продолжительность одного цикла анимации. Задаётся в секундах s или миллисекундах ms. Если для элемента задано более одной анимации, то можно установить разное время для каждой, перечислив значения через запятую.

Временная функция: **свойство animation-timing-function**

Свойство animation-timing-function описывает, как будет развиваться анимация между каждой парой ключевых кадров. Во время задержки анимации временные функции не применяются.

Повтор анимации: **свойство animation-iteration-count**

Свойство animation-iteration-count указывает, сколько раз проигрывается цикл анимации. Начальное значение 1 означает, что анимация будет воспроизводиться от начала до конца один раз. Это свойство часто используется в сочетании со значением alternate свойства animation-direction, которое заставляет анимацию воспроизводиться в обратном порядке в альтернативных циклах.

Направление анимации: **свойство animation-direction**

Свойство animation-direction определяет, должна ли анимация воспроизводиться в обратном порядке в некоторых или во всех циклах. Когда анимация воспроизводится в

обратном порядке, временные функции также меняются местами. Например, при воспроизведении в обратном порядке функция ease-in будет вести себя как ease-out.

Проигрывание анимации: **свойство animation-play-state**

Свойство animation-play-state определяет, будет ли анимация запущена или приостановлена. Остановка анимации внутри цикла возможна через использование этого свойства в скрипте JavaScript. Также можно останавливать анимацию при наведении курсора мыши на объект – состояние :hover.

Задержка анимации: **свойство animation-delay**

Свойство animation-delay определяет, когда анимация начнется. Задается в секундах s или миллисекундах ms.

Состояние элемента до и после воспроизведения анимации: свойство **animation-fill-mode**

Свойство animation-fill-mode определяет, какие значения применяются анимацией вне времени ее выполнения. Когда анимация завершается, элемент возвращается к своим исходным стилям. По умолчанию анимация не влияет на значения свойств animation-name и animation-delay, когда анимация применяется к элементу. Кроме того, по умолчанию анимация не влияет на значения свойств animation-duration и animation-iteration-count после ее завершения. Свойство animation-fill-mode может переопределить это поведение.

Краткая запись анимации: **свойство animation**

Все параметры воспроизведения анимации можно объединить в одном свойстве – animation, перечислив их через пробел:

animation: animation-name animation-duration animation-timing-function animation-delay animation-iteration-count animation-direction;

Для воспроизведения анимации достаточно указать только два свойства: animation-name и animation-duration, остальные свойства примут значения по умолчанию. Порядок перечисления свойств не имеет значения, единственное, время выполнения анимации animation-duration обязательно должно стоять перед задержкой animation-delay.

Множественные анимации

Для одного элемента можно задавать несколько анимаций, перечислив их названия через запятую:

```
div {animation: shadow 1s ease-in-out 0.5s alternate, move 5s linear 2s;}
```

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

В HTML трансформируемый элемент – это элемент с display: block; или display: inline-block;, а также элементы, значение свойства display которых вычисляется как table-row, table-row-group, table-header-group, table-footer-group, table-cell или table-caption.

Трансформированным считается элемент с любым установленным значением свойства transform, отличным от none.

В SVG трансформируемый элемент – это элемент, который имеет атрибуты transform, patternTransform или gradientTransform.

Существуют два вида CSS-трансформаций: 2D и 3D. 2D-**трансформации** преобразовывают элементы в двумерном пространстве с помощью 2D-матрицы преобразований. Эта матрица применяется для вычисления новых координат объекта, на основе значений свойств transform и transform-origin. Преобразования влияют только на

визуальный рендеринг. В отношении макета страницы они могут отразиться на переполнении содержимого блока. По умолчанию точка трансформации находится в центре элемента.

Свойство transform задаёт вид преобразования элемента. Свойство описывается с помощью функций трансформации, которые смещают элемент относительно его текущего положения на странице или изменяют его первоначальные размеры и форму.

Допустимые значения:

matrix() – любое число

translate(), translateX(), translateY() – единицы длины (положительные и отрицательные), %

scale(), scaleX(), scaleY() – любое число

rotate() – угол (deg, grad, rad или turn)

skew(), skewX(), skewY() – угол (deg, grad, rad)

Можно объединить несколько трансформаций одного элемента, перечислив их через пробел в порядке проявления.

Свойство transform-origin позволяет сместить центр трансформации, относительно которого происходит изменение положения/размера/формы элемента. Значение по умолчанию – center, или 50% 50%. Задаётся только для трансформированных элементов.

Все преобразования, определяемые свойствами transform и transform-origin, относятся к положению и размерам опорного блока элемента. Опорный блок элемента – это виртуальный прямоугольник вокруг элемента, который формирует систему координат для отрисовки.

В некоторых браузерах опорный блок принимает центр SVG-холста в качестве точки преобразования. Чтобы решить эту проблему, можно задать для элемента **transform-box**.

Опорный блок добавляет дополнительное смещение к исходной точке, заданной свойством transform-origin.

Задание:

Реализуйте следующие эффекты:

– Плавные переходы (transition) при наведении на ссылки и кнопки.

– Анимацию появления блока текста (keyframes), например, с эффектом увеличения и прозрачности.

– Трансформацию изображений (вращение, увеличение при наведении).

Дополнительно: используйте @keyframes для создания сложной анимации, например, анимации логотипа или иконки загрузки.

Ход выполнения лабораторной работы должен быть отражен в *отчете*. Отчет должен содержать титульный лист, цель работы, задание, листинг исходного кода, описание проделанной работы, скриншоты результата, вывод, ответы на контрольные вопросы.

Контрольные вопросы:

1. Для чего нужны CSS-переходы?
2. К каким элементам можно применить CSS-переходы?
3. Опишите свойство transition-duration.
4. Опишите свойство transition-timing-function.

5. Опишите свойство transition-delay.
6. Для чего нужны CSS-анимации?
7. К каким элементам можно применить CSS-анимацию?
8. Опишите правило @keyframes.
9. Опишите свойство animation-duration.
10. Опишите свойство animation-timing-function.
11. Опишите свойство animation-iteration-count.
12. Опишите свойство animation-direction.
13. Опишите свойство animation-play-state.
14. Опишите свойство animation-delay.
15. Опишите свойство animation-delay.
16. Опишите свойство animation-fill-mode.
17. Для чего нужны CSS-трансформации?
18. К каким элементам применяется CSS-трансформации?
19. Какие виды CSS-трансформации существуют?
20. Опишите свойство transform и его значения.
21. Опишите свойство transform-origin.