Лабораторная работа №8

Создание визуальных эффектов и анимации на HTML-странице

Цель урока:

обучающая: научить учащихся созданию и использованию визуальных эффектов в CSS;

развивающая: активизировать познавательную деятельность учащихся, развивать образное и логическое мышление, творческий подход к профессиональной деятельности;

воспитательная: воспитывать познавательный интерес к изучению адаптивных Html-страниц, аккуратности, дисциплинированности, ответственности при выполнении заданий.

Тип урока: урок практического применения знаний.

Формы работы учащихся: фронтальная, индивидуальная.

Необходимое техническое оборудовании: мультимедийный проектор, ноутбук, доступ к Интернету.

Длительность занятия : 2 часа.

Этапы урока:

1. Организационный момент (2 мин.)
2. Актуализация опорных знаний учащихся (8 мин.)
3. Применение знаний на практике (70 мин.)
4. Рефлексия и подведение итогов урока (8 мин)
5. Домашнее задание (2 мин.)

Список используемых источников:

1. Брылёва, А.А. Программные средства создания интернет-приложений : учеб. пособие / А.А. Брылёва. – Минск : РИПО, 2019. – 377 с.

Побединский, Е.В. Проектирование веб-сайтов с использованием технологий PHP, HTML, CSS и WordPress: учеб. пособие [Электронный ресурс] / Е.В. Побединский, В.В. Побединский. – Режим доступа: https://elar.usfeu.ru/bitstream/123456789/7602/1/WEB.pdf.

Теоретический материал

CSS-анимации позволяют анимировать переходы от одной конфигурации CSS стилей к другой. CSS-анимации состоят из двух компонентов: стилевое описание анимации и набор ключевых кадров, определяющих начальное, конечное и, возможно, промежуточное состояние анимируемых стилей.

Есть три преимущества CSS-анимации перед традиционными способами:

1. Простота использования для простых анимаций; вы можете создать анимацию, не зная JavaScript.
2. Анимации будут хорошо работать даже при умеренных нагрузках системы. Простые анимации на JavaScript, если они плохо написаны, часто выполняются плохо. Движок может использовать frame-skipping и другие техники, чтобы сохранить производительность на таком высоком уровне .
3. Позволяет браузеру контролировать последовательность анимации, тем самым оптимизируя производительность и эффективность браузера. Например, уменьшая частоту обновления кадров анимации в непросматриваемых в данный момент вкладках.

Чтобы создать CSS-анимацию вы должны добавить в стиль элемента, который хотите анимировать, свойство animation или его подсвойства. Это позволит вам настроить ускорение и продолжительность анимации, а также другие детали того, как анимация должна протекать. Это не поможет вам настроить внешний вид анимации, который настраивается с помощью @keyframes (en-US), рассматриваемой далее в Определение последовательности анимации с помощью ключевых кадров.

Свойство animation имеет следующие подсвойства:

animation-name
Определяет имя @keyframes (en-US), настраивающего кадры анимации.

animation-duration
Определяет время, в течение которого должен пройти один цикл анимации.

animation-timing-function
Настраивает ускорение анимации.

animation-delay
Настраивает задержку между временем загрузки элемента и временем начала анимации.

animation-iteration-count
Определяет количество повторений анимации; вы можете использовать значение infinite для бесконечного повторения анимации.

animation-direction
Даёт возможность при каждом повторе анимации идти по альтернативному пути, либо сбросить все значения и повторить анимацию.

animation-fill-mode

Настраивает значения, используемые анимацией, до и после исполнения.

animation-play-state
Позволяет приостановить и возобновить анимацию.

Пауза анимации при наведении курсора

Когда пользователь наводит курсор на элемент **.pin**, анимация псевдоэлементов должна становиться на паузу. CSS позволяет применять стили к дочерним элементам, когда на их родителя наведен курсор. То же самое работает и с псевдоэлементами.

```
.pin:hover:before,
.pin:hover:after {
   animation-play-state: paused;
}
```

Данный код говорит: когда на элемент **.pin** наведен курсор (состояние **:hover**), нужно поставить на паузу анимацию псевдоэлементов **:before** и **:after**.

Практический материал

Все задания выполняем на своем сайте!

(выбираем блок, который вам понравится и выполняем)

Задание 1. Скольжение текста

Скольжение текста в элементе <p> от правого края окна браузера.

Обратите внимание на то, что анимация может сделать страницу шире, чем окно браузера. Этого можно избежать, поместив элемент, который будет анимироваться, в контейнер и установив ему свойство overflow: hidden.

Задание 2. Добавьте другие ключевые кадры в предыдущее задание. Например, чтобы размер шрифта заголовка временно увеличивался по мере продвижения влево, а потом возвращался к первоначальному значению .

Задание 2. Смещение элемента

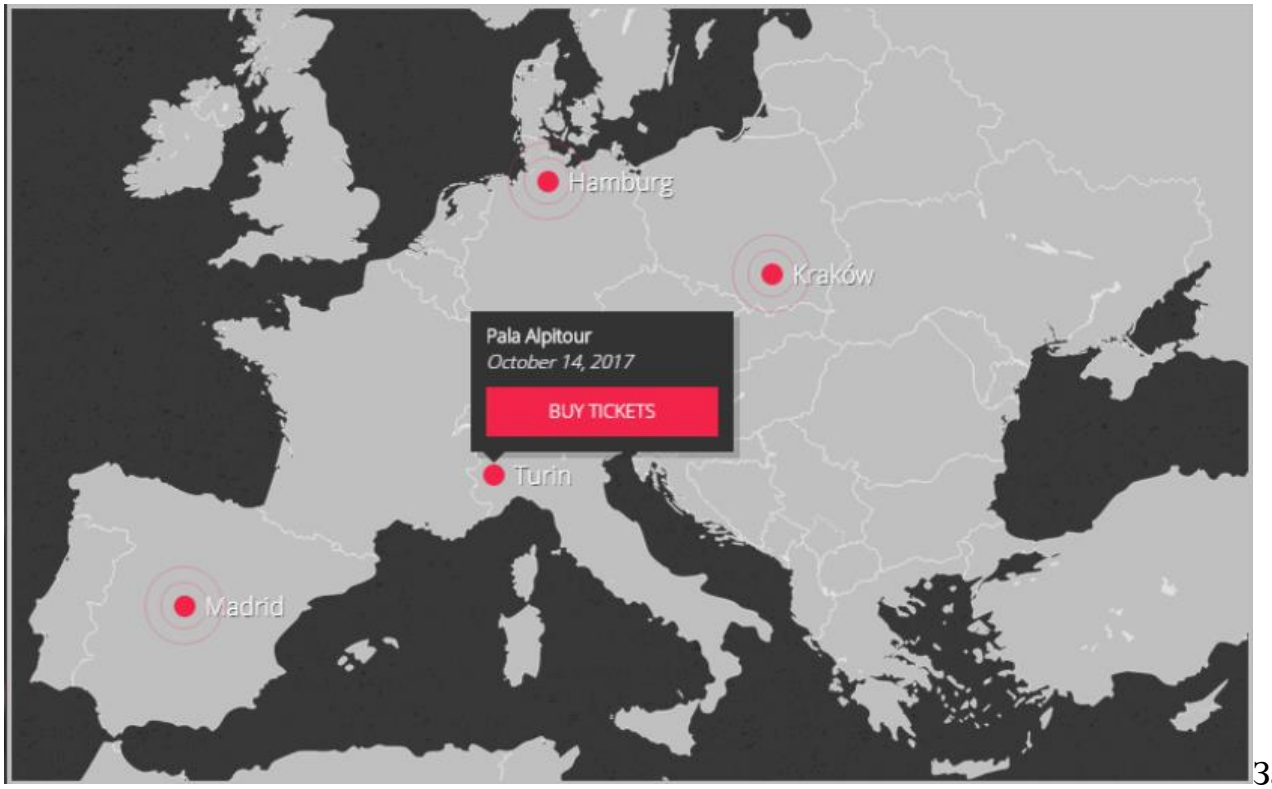
Установите смещение блока текста от левого края окна браузера и примените повторение.

Задание 3. Установка паузы

Когда пользователь наводит курсор на элемент, анимация псевдоэлементов должна становиться на паузу.

Задание 4. Обобщенное задание

В папке находится веб-страничка путешественников. Создайте интерактивную карту с выносками, в каких городах путешественники хотят побывать. При наведении курсора мыши на точку города пусть будет появляться отметка с примерной датой и кнопкой «Купить билет»:



Загрузите файлы из папки

Вам понадобятся определенные файлы для выполнения задания. Скачайте архив на компьютер и ознакомьтесь с его содержимым. Чтобы не тратить время на написание кода, который не относится напрямую к теме этого урока, мы заранее подготовили для вас HTML-разметку и базовые стили CSS. Вам понадобится добавить только те стили, которые будут касаться анимированных объектов.

План работы

1. Для четырех точек на карте (города, в которых будут проходить концерты) создать пульсирующую анимацию для привлечения внимания.
2. При наведении курсора на точку города анимация должна становиться на паузу. Когда курсор убран, анимация возобновляется.
3. Также при наведении курсора на точку города должна плавно появляться небольшая выноска с местом и датой концерта, кнопкой для покупки билетов. Когда курсор убран, информационная выноска исчезает.
4. Цвет кнопки в выноске должен меняться при наведении курсора.

1. Анимация для отметок на карте

Откройте в браузере веб-страницу из скачанного архива. Перед вами блок с картой, на которой нанесено четыре отметки с названиями городов. Наша задача — создать эффект расходящихся волн вокруг каждой точки. Сделайте две волны, в роли которых будут выступать псевдоэлементы **:before** и **:after**:

Псевдоэлементы удобно использовать, когда требуется добавить какое-то украшение к основному элементу. Это избавляет от необходимости добавлять лишний HTML-код, который не несет особого смысла.

Откройте в редакторе кода документ **style.css** из папки **css**. В течение урока мы будем соблюдать порядок в таблице стилей, добавляя новые правила не просто в конец документа, а размещая их там, где это будет логично. Поэтому найдите селектор **.pin .popover** и добавьте *над* ним следующие стили для псевдоэлементов элемента **.pin**:

.pin:before,
.pin:after {
Добавьте " ;
Установите позиционирование (Подсказка:элемент не существует в потоке документа и его положение задаётся относительно краёв браузера);
Установите смещение элемента влево и сверху на 50%;
Установите свойство display со значением block; (подумайте зачем)
Установите скругление блока до 50%;
Установите рамку равную 1px, сплошную, выберите цвет;
Установите высоту и ширину равную 0;
Установите внешний отступ слева и внешний отступ сверзу от элемента равный -2px;
}
Этими действиями вы не добавите никакой контент к этим псевдоэлементам, а только создаете пустой блок с рамкой и скругленными углами, чтобы в итоге получить кольцо.

В начальном виде ширина и высота псевдоэлементов равна нулю. И с помощью анимации вы увеличите их размеры, чтобы создать эффект расходящихся волн. Для этого понадобится создать ключевые кадры для каждой волны.

Найдите в начале таблицы стилей комментарий **/* Keyframes */** и запишите под ним два следующих правила **@keyframes**:

@keyframes pinBeforeWave {
from {
Установите высоту и ширину равную 0;
Установите внешний отступ слева и внешний отступ сверзу от элемента равный -2px;

}
to {
Установите высоту и ширину равную 40px;
Установите внешний отступ слева и внешний отступ сверзу от элемента равный -21px;
Установите прозрачность равную 0;
}
}

@keyframes pinAfterWave {
from {
Установите высоту и ширину равную 0;
Установите внешний отступ слева и внешний отступ сверзу от элемента равный -2px;
}
to {
Установите высоту и ширину равную 66px;
Установите внешний отступ слева и внешний отступ сверзу от элемента равный -34px;
Установите прозрачность равную 0;
}

}
Обе анимации имеют одинаковое начало, но отличаются окончанием. Анимацию **pinBeforeWave** мы применим к псевдоэлементу **:before**, и в процессе ее выполнения он увеличится до размеров 40×40 пикселей (маленькая волна). Анимация **pinAfterWave** предназначена для псевдоэлемента **:after**, к концу которой он примет размеры 66×66 пикселей (большая волна).
Кроме этого, в обеих анимациях предусмотрено смещение элементов на определенное количество пикселей влево и вверх — это нужно для того, чтобы при увеличении кольца отметка города визуально оставалась в его центре. И, наконец, свойство **opacity** в последнем ключевом кадре означает то, что обе волны будут плавно исчезать в процессе своего расширения.
Давайте применим созданные анимации к псевдоэлементам. Для этого добавьте следующий код, разместив его следом под стилем для селектора **.pin:before, .pin:after**:
.pin:before {
Применяем для анимации pinBeforeWave длительностью 1секунду, распределение скорости Начинается медленно и постепенно ускоряется, длится бесконечно;
}
.pin:after {
Применяем для анимации pinAfterWave длительностью 1секунду, распределение скорости Начинается медленно и постепенно ускоряется, длится бесконечно;
}
Обе анимации будут длиться одну секунду. Распределение скорости анимации в течение этой секунды будет происходить по функции *ease-in*. Анимация будет повторяться бесконечно.
Сохраните изменения в таблице стилей и обновите страницу в браузере. Анимация вокруг отметок уже работает, и нам пора переходить к следующему пункту плана.

2. Установка паузы анимации при наведении курсора мыши

Когда пользователь наводит курсор на элемент **.pin**, анимация псевдоэлементов должна становиться на паузу. CSS позволяет применять стили к дочерним элементам, когда на их родителя наведен курсор. То же самое работает и с псевдоэлементами. Добавьте этот стиль под предыдущим:
.pin: пользователь наводит на элемент мышью: отображения желаемого контента до содержимого элемента,
.pin: пользователь наводит на элемент мышью: отображения желаемого контента после содержимого элемента {
Свойство, которое определяет состояние анимации, паузы или проигрыша: пауза;
}
Данный код говорит: когда на элемент **.pin** наведен курсор (состояние **:hover**), нужно поставить на паузу анимацию псевдоэлементов **:before** и **:after**.

3. Появление выноски

При наведении курсора на отметку должно происходить еще одно действие, а именно появление выноски с информацией о путешествии. В изначальном состоянии все выноски скрыты от глаз посетителя с помощью свойств **visibility: hidden** и **opacity: 0**.
Найдите в файле CSS селектор **.pin .popover:before** и добавьте следом за ним следующий стиль:
.pin:hover .popover {

```
    visibility: visible;
    opacity: 1;
}
```

У вас мог возникнуть вопрос, зачем вы скрыли выноску сразу двумя способами. Дело в том, что использования лишь одного из этих свойств будет недостаточно. Если убрать свойство **visibility** и оставить только **opacity: 0**, то элемент не будет по-настоящему скрыт: да, он не будет виден для глаз, но его увидит считывающее устройство, что иногда может мешать.

Если же убрать свойство прозрачности и оставить только **visibility: hidden**, то мы не сможем добиться плавности при появлении элемента, поскольку свойство *transition* не действует на свойство **visibility**. Именно поэтому мы использовали сразу два свойства для скрытия выноски.

Кстати о плавности. Если вы сейчас обновите страницу и наведете курсор на точку, то увидите, что выноска появляется, но делает это резко. Настало время обратиться к свойству, которое позволяет определить переходное состояние между двумя состояниями элемента. Найдите селектор **.pin .popover** и добавьте к нему строку:

свойству, которое позволяет определить переходное состояние между двумя состояниями элемента: 0.2s, сначала медленно, потом быстро, в конце опять медленно;

Теперь выноска появляется плавно: свойство **opacity** переходит от значения **0** к значению **1** за 200 миллисекунд.

Идем дальше. Когда мы рассматривали свойство **transition-delay**, то говорили о том, что его удобно применять для небольшой задержки выпадающего меню перед его исчезновением, чтобы пользователь успевал навести курсор на ссылку меню. Не лишним будет применить этот эффект и для выноски, чтобы она исчезала с легкой задержкой. Для этого добавьте еще одно значение к свойству **transition** для селектора **.pin .popover** — **0.5s** перед точкой с запятой:

свойству, которое позволяет определить переходное состояние между двумя состояниями элемента: изменение всех свойств, 0.2s сначала медленно, потом быстро, в конце опять медленно 0.5s;

Но это еще не всё. Сейчас задержка появления выноски срабатывает в обоих направлениях — когда курсор наводится и когда отводится. Чтобы оставить задержку только для второго случая, допишите следующую строку к селектору **.pin:hover .popover**:

устанавливает время ожидания перед запуском эффекта перехода: 0s;

Сохраните изменения и обновите страницу в браузере. Теперь всё работает правильно: выноска появляется плавно и без задержки, а исчезает так же плавно, но с задержкой в полсекунды.

Нам осталось поработать еще над одной деталью. На этот раз мы вспомним свойство **transform** и его функцию вращения **rotate**, чтобы добавить еще один визуальный эффект при появлении выноски. Она будет словно разворачиваться лицом к зрителю в процессе появления, а затем так же исчезать.

В состоянии невидимости выноска будет развернута на 90 градусов по оси Y. Допишите следующий стиль к селектору **.pin .popover**:

Установите свойство, которое позволяет поворачивать элемент вокруг оси Y на 90 градусов;

При наведении курсора на точку города выноска должна возвращаться в свое нормальное положение. Для этого добавьте строку ниже к стилю для селектора **.pin:hover .popover**:

Установите свойство, которое позволяет поворачивать элемент вокруг оси Y на 0 градусов;

Сохранитесь и обновите веб-страницу. Понаблюдайте теперь за поведением выносок: они появляются плавно и с эффектом разворота, напоминая карточки.

4. Изменение цвета кнопки в выноске

Последний пункт плана — плавное изменение цвета кнопки в выноске при наведении курсора на нее. Для начала создадим стиль для кнопки в состоянии **:hover**. Найдите селектор **.pin .popover .button** и запишите под ним следующий код:

```
.pin .popover .button:hover {
    цвет фона элемента: выберите свой;
}
```

И, наконец, чтобы переход от начального цвета к конечному происходил плавно, добавьте эту строку к стилю селектора **.pin .popover .button**:

```
transition: all 0.2s ease-in-out;
```

Снова сохраните изменения и полюбуйтесь результатом.

Завершение

Поздравляю! Вы полностью создали интерактивную карту.