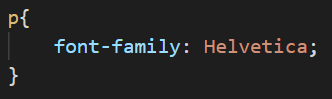
# Занятие 3. Шрифты. Анимация.

## Шрифты. Загрузка и использование шрифтов

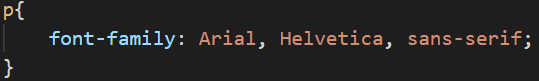
На текущий момент мы умеем строить приятные глазу абзацы, изменять цвет и размеры текста, работать с фоном, но этого, порой, недостаточно для того, чтоб текст на сайте приятно было читать. Не последнюю роль в этом играют шрифты. Одни – прекрасно выглядят, другие - отлично читаются, третьи – идеально распознаются компьютером. В зависимости от целей, поставленных перед заголовком, абзацем и любым другим фрагментом текста, можно использовать разные шрифты.

Для указания, какой конкретно шрифт должен использовать тот или иной элемент на странице применятся свойство font-family, в котором и указывается имя желаемого шрифта. Но тут есть одна засада: свойство font-family будет функционировать только в том случае, если указанный в нём шрифт есть у пользователя на компьютере. Допустим, есть необходимость отобразить параграф шрифтом Helvetica. Выглядеть команда будет так: 

Так вот, сказанное выше означает, что если у одного из пользователей на компьютере не будет шрифта Helvetica, то font-family будет проигнорировано и будет использован шрифт по умолчанию (обычно это Times New Roman).

Чтобы избежать подобной проблемы современные браузеры используют, так называемые, веб-шрифты, которые подключаются дополнительно в css файле правилом @font-face, разберём его далее.

Вообще, есть известный ряд шрифтов стандартный, который есть на большинстве компьютеров и, как правило, дизайнеры останавливают выбор на них, но если красивое оформление текста очень важно для дизайна сайта – веб-шрифты всегда помогут.

Так как описанная выше проблема мучала (и продолжает) людей крайне долго, в сообществе принято в свойстве font-family указывать не один определенный шрифт, а целый стек (или набор) очень похожих. Например  Браузер, увидев такой стиль, будет опрашивать компьютер, есть ли на нём хоть один из представленных шрифтов (в том порядке, в котором они указаны, то есть сначала arial, затем Helvetica и потом любой рубленный шрифт).

В этой связи, обозначим 3 большие группы шрифтов, и какие наборы для них чаще всего указываются разработчиками web-сайтов.

## Антиквенные шрифты

Антиквенные шрифты (с засечками) идеальны для длинных фрагментов текста, поскольку распространено мнение, что засечки — маленькие росчерки на концах основных штрихов — хорошо для глаз связывают одну букву с другой, делая текст более читабельным. Рассмотрим основные наборы:

* "Times New Roman", Times, serif (если название шрифта из нескольких слов – его необходимо записывать в кавычках)
*  Georgia, "Times New Roman", Times, serif
* Baskerville, "Palatino Linotype", Times, serif
* "Hoefler Text", Garamond, Times, serif

## Рубленные шрифты

Рубленые шрифты (без засечек) часто используются для заголовков благодаря простому и четкому внешнему виду. Основные наборы:

* Arial, Helvetica, sans-serif
* Verdana, Arial, Helvetica, sans-serif
* Geneva, Arial, Helvetica, sans-serif
* Tahoma, "Lucida Grande", Arial, sans-serif
* "Trebuchet MS", Arial, Helvetica, sans-serif
* "Century Gothic", "Gill Sans", Arial, sans-serif

## Моноширинные и декоративные шрифты

Моноширинный шрифт часто используется для отображения компьютерного кода. Каждая буква в моноширинном шрифте имеет одинаковую ширину (как буквы в механических печатных машинках). Основные наборы:

* "Courier New", Courier, monospace
* "Lucida Console", Monaco, monospace
* "Copperplate Light", "Copperplate Gothic Light", serif
* "Marker Felt", "Comic Sans MS", fantasy

Сразу добавлю, что выглядит этот набор шрифтов плохо, но советую попробовать, посмотреть, как и что поменяется. Так же, на [сайте](https://www.cssfontstack.com/) можно увидеть, какие шрифты наиболее распространены среди пользователей с определенной операционной системой.

## Использование web-шрифтов.

Для успешного использования такого рода шрифтов потребуется выполнить 3 действия:

1. Найти в интернете и загрузить понравившийся шрифт. Обычно это файл с разрешением eot, ttf, otf, woff. EOT – шрифты для интернет-эксплорера. TTF, OTF – True Type Fonts или Open Type Fonts, ранее – самые распространённые шрифты. WOFF – Web Open Font Face, более сжатый и оптимизированный TTF, работает на всех современных браузерах
2. Использовать правило @font-face для добавления шрифта на страницу
3. Применить шрифт с помощью добавления свойства font-family стилизуемому элементу

В качестве примера используем шрифт Zabatana. Его внешний вид представлен на рисунке 3.1. Найти его можно на [сайте](https://fontlibrary.org/en/font/zabatana) с шрифтами.

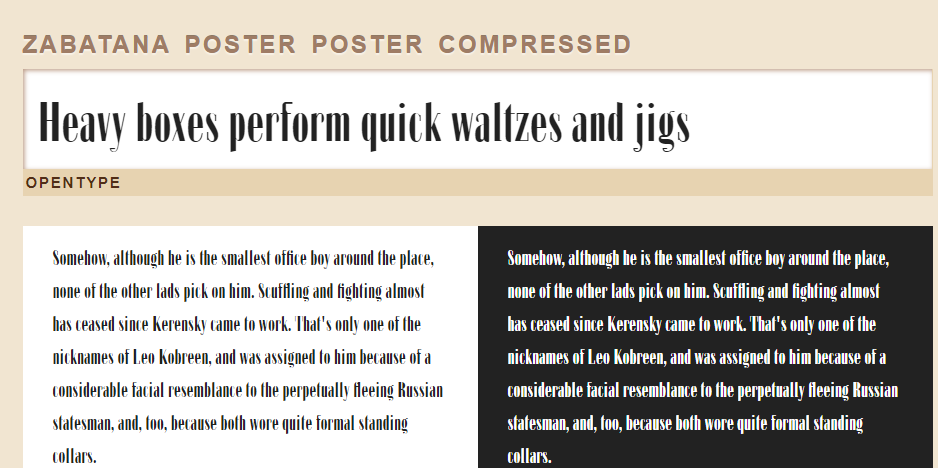


Рисунок 3.1 – Внешний вид шрифта для примера

Идём по пунктам, изложенным выше. Скачиваем шрифт с сайта. Загружается целый архив с шрифтом в разных расширениях, извлекаем файлы, оставить достаточно одну вариацию (для примера оставлен ttf).

В папке с сайтом создается папка fonts, там и будут храниться все файлы шрифтов. Пример на рисунке 3.2.

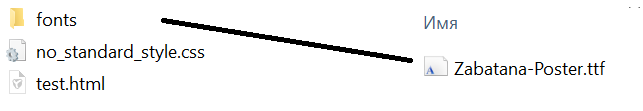


Рисунок 3.2 – Расположение файла с шрифтом

Далее, макет стилизуемой страницы, а также её вид до добавления шрифта, представлен на рисунке 3.3.

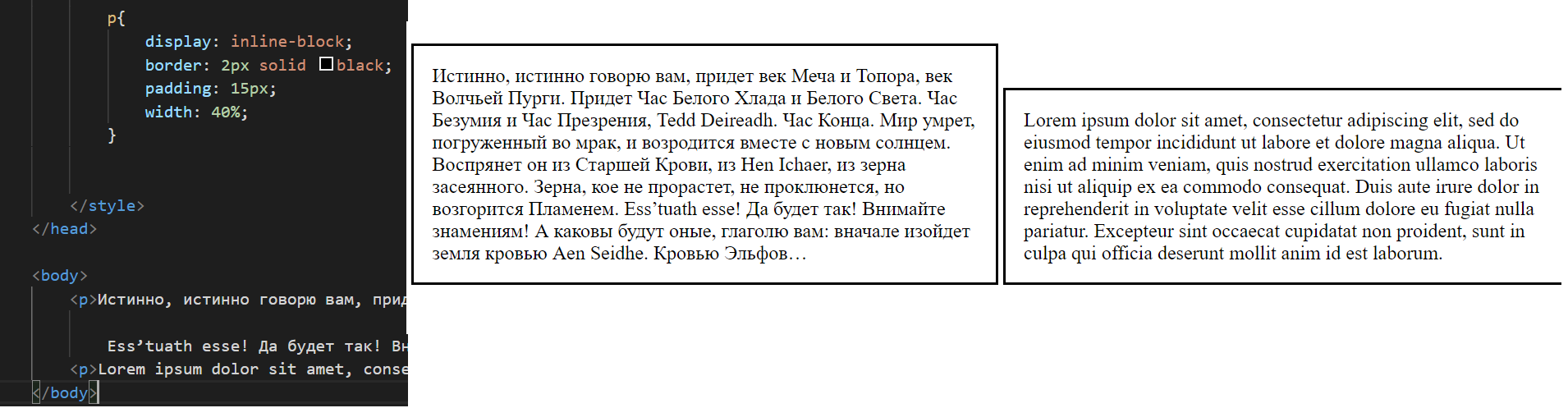


Рисунок 3.3 – Начальный вид страницы

Применяем «правило» font-face. В нём указывается 2 свойства: название шрифта (font-family) и ссылка на шрифт на компьютере или сервере (src).

После того, как «правило» font-face заполнено, останется лишь указать font-family в стилях для тэга p. Полный код примера здесь. Результат и изменения, внесённые в код, можно увидеть на рисунке 3.4. Исходный код [здесь](https://github.com/hellfire0102/web_programming_course/blob/main/test8.html).

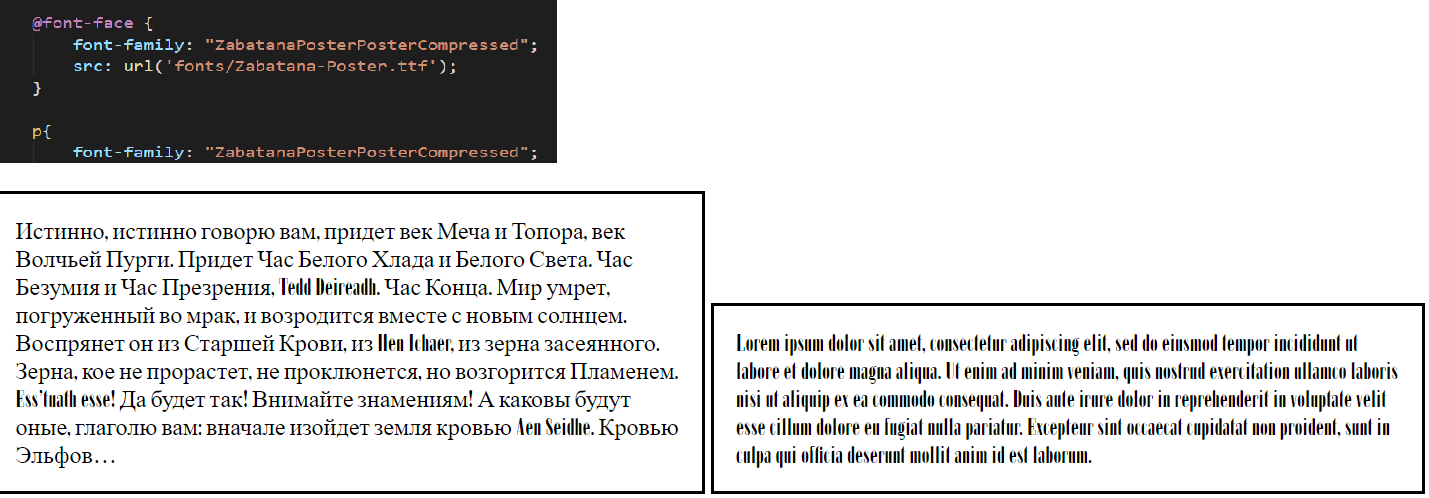


Рисунок 3.4 – результат подключения шрифта

Как видно из итогового результата, шрифт применился только для слов, написанных латиницей, что и было указано на сайте, откуда шрифт был скачан. А так, как в свойстве font-family для p указан лишь 1 шрифт (который не подходит для русского), для него остался стандартный шрифт браузера. Пример ниже продемонстрирует, почему полезно указывать не 1 шрифт, а сразу несколько. Добавим приписку sans-serif (рубленный, красивый, для заголовков). Результат на рисунке 3.5.

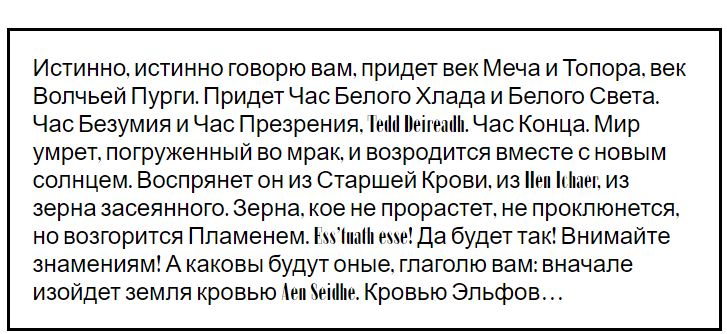


Рисунок 3.5 – Результат редактирования свойства font-family

С web-шрифтами есть несколько проблем. С одной из них столкнулись выше – они могут быть не мультиязычны, вторая проблема – их сложнее декорировать. Например, многие веб-шрифты не делаются курсивными или жирными, а некоторые напротив не становятся тонкими. Иногда для этого приходится скачивать несколько вариаций одного и того же шрифта. При этом в правиле font-face указывать, для каких свойств font-weight и font-style, какой из вариантов использовать.

## Анимация CSS

Каскадные таблицы стилей представляют множество возможностей по преобразованию элемента страницы: вращение, перемещение, масштабирование, искажение вдоль горизонта или вертикальной оси (скашивание). Есть возможность даже комбинировать разные эффекты.

Основным свойством CSS для получения любого из перечисленных выше изменений является transform. Внутрь него передаётся действие, которое необходимо произвести над элементом, допустим transform: rotate(20deg); - вращение элемента на 20 градусов по часовой стрелке.

Если в процессе поиска готовых решений будете видеть свойство вида -webkit-transform, -ms-transform и другие – пугаться не стоит. Эти свойства нужны, дабы анимация работала в браузерах сафари и эксплорер, соответственно. Префиксы есть далеко не только для transform, но и для градиентов и многих других свойств.

Вообще, по-умному, то, что описано выше, называется «вендорными префиксами», о них подробнее расскажет вырезка из книги «Новая большая книга CSS» на рисунке 3.6.

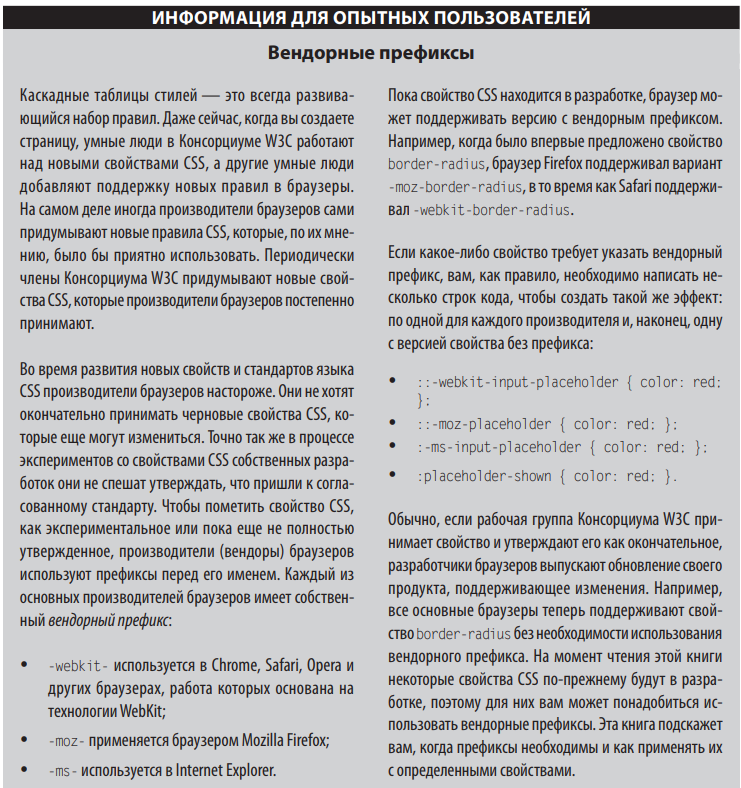


Рисунок 3.6 – Про вендорные префиксы

При использовании transform нужно понимать один важный момент: браузер выделяет пространство под элемент в момент его первой рисовки, то есть, до каких-либо трансформаций, а это значит, что в процессе поворота, увеличения размера и т.д. изменяемый элемент может накладываться на другие части страницы, заслоняя их собой.

Из-за описанных выше проблем с размещением анимированных элементов – такой подход САМОСТОЯТЕЛЬНО используют не слишком часто (а добавляют к анимациям или переходам), а потому ниже приведён лишь один пример, демонстрирующий работу transform. В примере анимируется кнопка. Она при наведении курсора мыши меняет свой размер. Исходный [код](https://github.com/hellfire0102/web_programming_course/blob/main/test9.html) примера на рисунке 3.7.

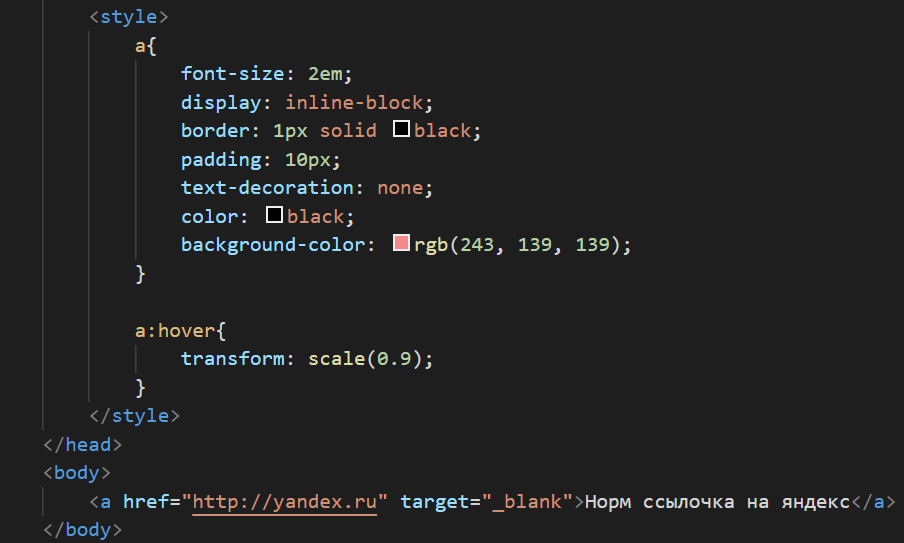


Рисунок 3.7 – Пример использования transform

Стоит лишь добавить, что при использовании таких преобразований элемента как rotate (вращение), skew(скашивание), translate(перемещение), браузер всегда должен отталкиваться от некой точки на элементе, относительно которой будут происходить изменения. По умолчанию, эта точка находится в центре элемента (чтобы в этом убедиться, поэкспериментируйте со свойством rotate, оно наиболее репрезентативно). Но положение этой точки можно менять, для этого используется свойство transform-origin, в котором может указываться как положение на краю элемента (допустим, left top), так и в координатах (допустим, transform-origin: 10px 15px).

## Переходы. Настоящая анимация

Преобразования, конечно, интересны и действительно оживляют страничку, но по-настоящему оживления удастся достичь с помощью переходов! Переходы – смена одного набора стилей элемента другим, за указанное время. В качестве примера – изменим размер кнопки и её цвет за 2 секунды. Что для этого понадобится?

* Два стиля для элемента. Один - будет указывать форматирование элемента до анимации, второй – как будет выглядеть элемент после перехода.
* Свойство transition. Как правило, оно применяется к исходному элементу, до начала анимации (используется далеко не всегда и не везде)
* Инициатор. Это действие пользователя или какое-то событие на сайте, которое запускает выполнение анимации (например псевдокласс :hover). Как только событие перестало выполняться (допустим, пользователь убрал указатель мыши с кнопки), элемент тут же возвращается в начальное состояние, так же при помощи анимации, за то же время.

Код примера до анимации представлен на рисунке 3.8.



Рисунок 3.8 – Начальное состояние, до анимации

Переходы добавляются при помощи свойства transition. Код указывается в том же блоке, в котором описано начальное состояние. Итак, какие же свойства существуют для настройки transition.

* Transition-property. Определяет свойство или набор свойств, изменение которых нужно анимировать. Пример: transition-property: background-color, color. В этом кусочке примера мы хотим анимировать изменение цвета фона и цвета шрифта. В случае когда указывается несколько свойств для анимации рекомендуется для каждого указать свой duration и delay. Также есть возможность заставить анимироваться любые изменяющиеся свойства CSS (которые в принципе возможно анимировать). Для этого необходимо воспользоваться следующим кодом: transition-property: all;
* Transition-delay. Определяет, через какое время после совершения определенного события необходимо запустить анимацию. Простой пример – допустим, анимацию изменения внешнего вида кнопки нужно запускать через 0.1 секунду после наведения на неё указателя мыши. Код будет выглядеть так: transition-delay: 0.1s; Обычно свойства transition помещают в начальное состояние элемента (без псевдоклассов), но вот со свойством duration можно поступать иначе, например, в свлучае с всплывающими меню, чтобы если вдруг пользователь случайно убрал указатель мыши с пункта меню, оно тут же не закрылось
* Transition-duration. Похоже на свойство, рассмотренное выше, только отвечает оно за длительность анимации. (Вспомним момент, когда анимации не было и просто использовался hover, выглядело всё слишком резко, описанное свойство – решение проблемы)
* Transition-timing-function. Отвечает за скорость выполнения анимации. Очень легко спутать её с transition-duration, но они совершенно разные. Duration отвечает за длительность процесса анимации. А timing-function за то, как распределена анимация по установленному duration времени (равномерно, быстро вначале, медленно в конце или наоборот). В качестве значения этого свойства могут использоваться 5 ключевых слов: linear, ease, ease-in, ease-out, ease-in-out. Подробнее про каждый вариант: По умолчанию используется **ease** – анимация медленная вначале, быстрая в середине и вновь замедляется к концу. Вариант **linear** производит изменения равномерно во всё время анимации. В случае с остальными вариантами – проще их попробовать, чем описать, что будет происходить. Поэкспериментируйте

Пример решения задачи, с которой начался этот урок, а именно изменение размера кнопки и её цвета при наведении с использованием анимации на рисунке 3.9. Исходный код примера [здесь](https://github.com/hellfire0102/web_programming_course/blob/main/test10.html).

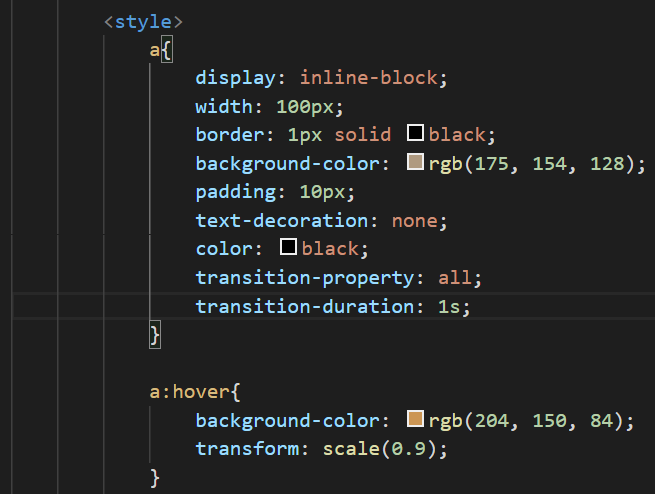


Рисунок 3.9 – Добавление анимации в пример

Ну и в качестве развлекательного примера – вращающийся кружочек с акцией на сайте, который также будет вращаться при наведении указателя мыши. Демонстрация [кода](https://github.com/hellfire0102/web_programming_course/blob/main/test11.html) и получившегося элемента на рисунке 3.10.

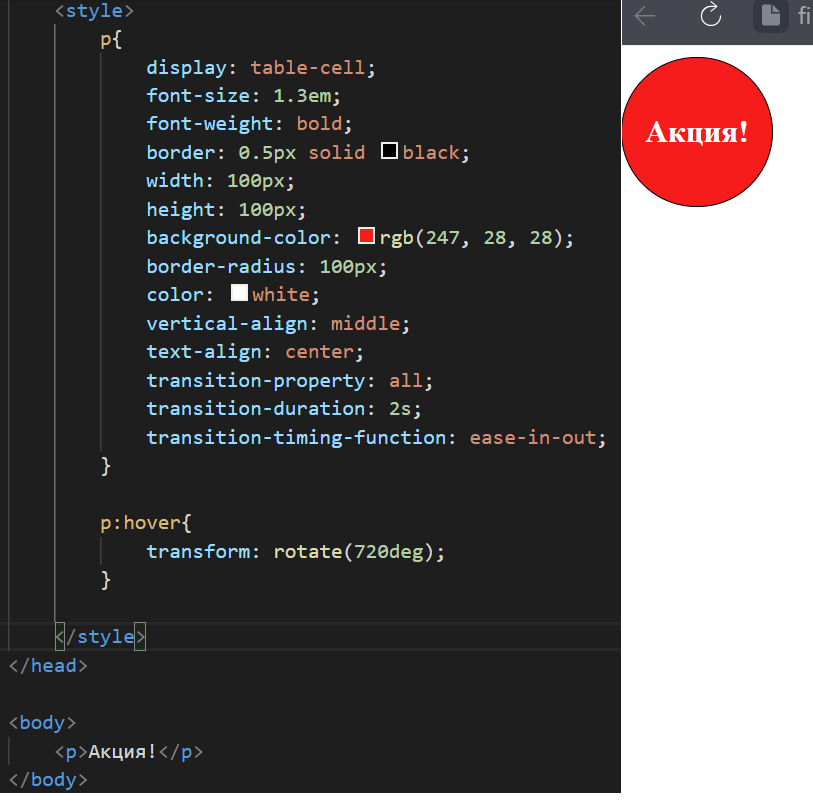


Рисунок 3.10 – Пример красивой анимации вращения

## Совсем настоящая анимация. Команда @keyframes

С помощью показанных выше переходов можно запустить смену одного набора стилей для элемента другим за определенное время. Анимация же позволяет производить несколько переходов! То есть от начального набора свойств к второму, затем к третьему и т.д.

В отличие от обычных переходов – анимации не нужен инициатор (то есть, она может воспроизводиться при загрузке страницы, автоматически привлекая внимание посетителей сайта к определенному его участку).

Для указания списка промежуточных состояний используются так называемые «ключевые кадры» (@keyframes). Этот элемент, как и @font-face указывается в самом верху стилевого файла, над всеми остальными стилевыми элементами. В случае с font-face это оправдано тем, что стилевой файл интерпретируется сверху вниз и web-шрифт успеет загрузиться или подключиться до момента, когда первый раз будет использован. В случае с анимацией это оправдано тем, что некоторые анимации могут запускаться при загрузке страницы, сразу, ну и второй аргумент – читабельность.

Итак, основные 2 этапа создания анимации:

1. Определение анимации. Настройка ключевых кадров со списком анимируемых свойств CSS
2. Применение анимации к элементу

## Определение ключевых кадров

Синтаксис правила @keyframes, предназначенного для определения кадров, представлен на рисунке 3.11.

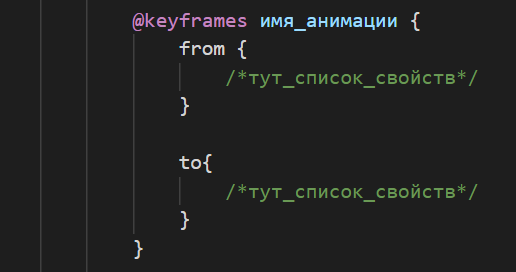


Рисунок 3.11 – Синтаксис правила @keyframes

Как видно из примера, после ключевого слова keyfarames обязательно указывается имя анимации (оно должно быть описательным, чтобы было понятно примерно, что оно делает, например fadeIn или fadeOut). Затем, как и в случае с переходом, указывается из какого, в какое состояние должен перейти элемент (в секциях from и to перечисляются свойства CSS, которые должны сменить друг друга).

## Примечание

@keyframes не является свойством css, а является, так называемым, правилом. Есть и другие похожие @font-face (с которым мы встречались ранее в уроке), @media для определения стилей для различных типов аппаратуры (с помощью этого правила делается адаптивный дизайн), @import для загрузки внешней таблицы стилей.

## Ключевые кадры. Продолжение

В качестве самого простого примера, используем появление параграфа на странице (то есть, изменение свойства opacity). Исходный код [здесь](https://github.com/hellfire0102/web_programming_course/blob/main/test12.html). Пример на рисунке 3.12.



Рисунок 3.12 – Пример простой анимации

На данном этапе особого отличия от перехода нет, тоже 2 состояния, тоже указано время анимации, в состояниях свойства. Но кадров, как было описано выше, может быть много. С помощью процентных значений можно определить множество кадров. Процентное значение определяет момент общей длительности анимации, в который должно произойти изменение. Допустим, есть необходимость сделать, чтоб с середины анимации, помимо изменения цвета фона, начал расти и сам элемент. Исходный код примера представлен на рисунке 3.13 и по [ссылке](https://github.com/hellfire0102/web_programming_course/blob/main/test13.html).

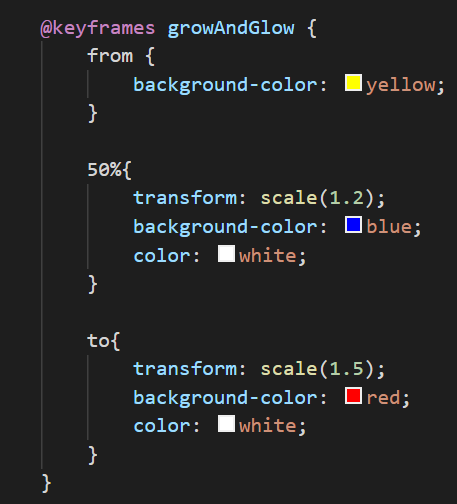


Рисунок 3.13 – Множество ключевых кадров

Таким образом, можно создавать огромное количество кадров, а процентами указывать, когда они должны начинать применяться (соответственно, заканчивает применяться кадр при переходе к следующему, допустим, один начала применяться в 20%, а другой в 40%, то от 20% до 40% применяется первый кадр, а затем переходит ко второму)

Как стало понятно из вышеописанного, сама по себе анимация не привязана к конкретному элементу, потому может быть использована в множестве мест и вызвана при помощи css свойства animation-name.

К одному элементу страницы можно применять несколько различных анимаций, перечисляя их имена через запятую в свойстве animation-name и указывая их длительность в свойстве animation-duration так же через запятую.

Кстати, так же как и в переходах, для анимаций существует свойство animation-timing-function, которое отвечает за скорость и имеет те же значения, что и в переходах, советую так же поэкспериментировать. Вдобавок указывать это свойство можно не только в стилях анимируемого элемента, но и внутри описания кадров, если есть необходимость управлять скоростью при переходах между конкретными кадрами.

Безусловно, что, как и в переходах, к анимации применимо свойство animation-delay, отвечающее за отсрочку начала анимации.

Также существует возможность указывать количество раз, которое анимация будет применяться, для этого используется свойство animation-iteration-count. Оно может принимать значение в целых числах (1, 2, 10) или в словах, например, infinite.

Чтобы добиться эффекта «мигания», то есть, в первый раз анимация шла в прямом порядке, а второй раз в обратном, применяется свойство animation-direction и указывается его значение alternate ( по умолчанию, анимация всегда идёт в одном направлении, будто этому свойству присвоено значение normal)

Неважно, сколько раз и в каком направлении будет происходить анимация – в момент, когда она будет завершена, элемент вернётся в своё начальное состояние, что было до всяких анимаций, если это не то, что вы хотите, можно зафиксировать элемент в конечном состоянии при помощи свойства animation-fill-mode: forwards;

## Самостоятельная работа

Добавить на свой сайт вращающийся и искажающийся любым образом заголовок (пускай вращение будет медленным, дабы глаза у пользователя не выпали). Пускай анимация длится бесконечно.

Попробовать использовать на сайте любой из веб шрифтов, например [этот](https://fontlibrary.org/ru/font/fedorovsk-unicode).