Лекція 4.
CSS анімація

Параметри зміни елемента

Трансформує елемент, зокрема, дозволяє його масштабувати, обертати, зрушувати, нахиляти, а також комбінувати види трансформацій.

transform: <функція> [<функція>] * | none

функції трансформації
matrix Задає матрицю перетворень .
rotate Поворот елемента на заданий кут відносно точки трансформації, що задається властивістю transform-origin .
transform: rotate (< кут >)
scale Масштаб елемента по горизонталі і вертикалі.
transform: scale (sx [, sy]); Значення більше 1 збільшує масштаб елемента, менше 1 - зменшує масштаб.
scaleX Масштабує елемент по горизонталі.
transform: scaleX (sx);
scaleY Масштабує елемент по вертикалі.
transform: scaleY (sy);
skewX Нахиляє елемент на заданий кут по вертикалі.
transform: skewX (< кут >)
skewY Нахиляє елемент на заданий кут по горизонталі.
transform: skewY (< кут >)
translate Зрушує елемент на задане значення по горизонталі і вертикалі.
transform: translate (tx [, ty])
translateX Зрушує елемент по горизонталі на вказане значення. Позитивне значення зрушує вправо, негативне вліво.
transform: translateX (tx)
translateY Зрушує елемент по вертикалі на вказане значення. Позитивне значення зрушує вниз, негативне вгору.

transform: translateY (ty)

Описание
Универсальное свойство, которое позволяет одновременно задать значения transition-property, transition-duration, transition-timing-function и transition-delay. Устанавливает эффект перехода между двумя состояниями элемента, они могут быть определены с помощью псевдоэлемента :hover или :active, а также динамически через JavaScript.

Синтаксис
transition: <переход> [, <переход>]*

Здесь:

<переход> = [none | <transition-property>] || <transition-duration> || <transition-timing-function> || <transition-delay>

Описание
Свойство transition-delay устанавливает время ожидания перед запуском эффекта перехода. Значение 0s или 0ms запускает анимацию сразу же. Отрицательное значение также включает анимацию без задержек, но может привести к изменению вида начала анимации.

Допустимо указывать несколько значений, перечисляя их через запятую. Каждое значение будет применяться к свойству, заданному в параметрах transition-property.

Синтаксис
transition-delay: <время> [,<время>]*

Описание
Устанавливает имя стилевого свойства, значение которого будет отслеживаться для создания эффекта перехода.

Синтаксис
transition-property: none | all | <свойство> [,<свойство>]*

Значения
none

Никакое свойство не задано.
all
Все свойства будут отслеживаться.
<свойство>
Название стилевого свойства, регистр при его написании не учитывается. При указании нескольких свойств они перечисляются друг за другом через запятую.

Описание
Устанавливает, насколько быстро должно изменяться значение стилевого свойство для которого применяется эффект перехода.

transition-timing-function представляет собой математическую функцию, показывающую, как быстро по времени меняется указанное через transition-property значение свойства. Начальная точка имеет координаты 0.0, 0.0, конечная — 1.0, 1.0, при этом функция по оси ординат может превышать эти значения в большую или меньшую сторону (рис. 1).

Рис. 1. Вид функции

Синтаксис
transition-timing-function: ease|ease-in|ease-out|ease-in-out|linear|step-start|step-end|steps|cubic-bezier

Значения
ease
Анимация начинается медленно, затем ускоряется и к концу движения опять замедляется. Аналогично cubic-bezier(0.25,0.1,0.25,1).
ease-in
Анимация медленно начинается, к концу ускоряется. Аналогично cubic-bezier(0.42,0,1,1).
ease-out
Анимация начинается быстро, к концу замедляется. Аналогично cubic-bezier(0,0,0.58,1).
ease-in-out
Анимация начинается и заканчивается медленно. Аналогично cubic-bezier(0.42,0,0.58,1).

linear
Одинаковая скорость от начала и до конца.
step-start
Как таковой анимации нет. Стилевые свойства сразу же принимают конечное значение.
step-end
Как таковой анимации нет. Стилевые свойства находятся в начальном значении заданное время, затем сразу же принимают конечное значение.
steps
Ступенчатая функция, имеющая заданное число шагов.
transition-timing-function: steps(<число>, start | end)

Здесь: <число> — целое число больше нуля; start — задаёт полунепрерывную снизу функцию; end — задаёт полунепрерывную сверху функцию.
cubic-bezier
Задаёт функцию движения в виде кривой Безье.

Введение в CSS анимацию

Содержание:
1. Правило @keyframes
2. Название анимации animation-name
3. Продолжительность анимации animation-duration
4. Временная функция animation-timing-function
5. Анимация с задержкой animation-delay
6. Повтор анимации animation-iteration-count
7. Направление анимации animation-direction
8. Краткая запись анимации animation
9. Проигрывание анимации animation-play-state
10. Состояние элемента до и после воспроизведения анимации animation-fill-mode
11. Множественные анимации
12. Урок: создание анимации

CSS3 анимация придаёт сайтам динамичность. Она оживляет веб-страницы, улучшая взаимодействие с пользователем. В отличие от CSS3 переходов,

создание анимации базируется на ключевых кадрах, которые позволяют автоматически воспроизводить и повторять эффекты на протяжении заданного времени, а также останавливать анимацию внутри цикла.

CSS3 анимация может применяться практически для всех html-элементов, а также для псевдоэлементов :before и :after. Список анимируемых свойств приведен на этой странице. При создании анимации не стоит забывать о возможных проблемах с производительностью, так как на изменение некоторых свойств требуется много ресурсов.

Также не следует использовать анимацию только потому, что вы подобные эффекты нравятся вам. Неуместная анимация будет раздражать пользователей и мешать им. Анимация должна поддерживать взаимодействие с пользователем. Например, можно добавить небольшой эффект (отблеск или отскок) при нажатии на кнопку, чтобы подтвердить взаимодействие.

Введение в CSS анимацию

Содержание:
1. Правило @keyframes
2. Название анимации animation-name
3. Продолжительность анимации animation-duration
4. Временная функция animation-timing-function
5. Анимация с задержкой animation-delay
6. Повтор анимации animation-iteration-count
7. Направление анимации animation-direction
8. Краткая запись анимации animation
9. Проигрывание анимации animation-play-state
10. Состояние элемента до и после воспроизведения анимации animation-fill-mode
11. Множественные анимации
12. Урок: создание анимации
Поддержка браузерами

IE: 10.0
Firefox: 16.0, 5.0 -moz-
Chrome: 43.0, 4.0 -webkit-

Safari: 4.0 -webkit-
Opera: 12.1, 12.0 -o-
iOS Safari: 9, 7.1 -webkit-
Opera Mini: —
Android Browser: 44, 4.1 -webkit-
Chrome for Android: 44

1. Правило @keyframes

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.

CSS

```
@keyframes shadow {
from {
  text-shadow: 0 0 3px black;
}
50% {
  text-shadow: 0 0 30px black;
}
to {
  text-shadow: 0 0 3px black;
}
}
1
2
3
4
5
6
7
```

```
8
9
10
11
@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:

CSS

```
@keyframes move {
from, to {
top: 0;
left: 0;
}
25%, 75% {
top: 100%;
}
50% {
top: 50%;
}
1
2
3
```

4
5
6
7
8
9
10
11
@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 у двух ключевых кадров будут одинаковые селекторы, то последующий отменит действие предыдущего.

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

CSS

h1 {
font-size: 3.5em;
color: darkmagenta;
animation: shadow 2s infinite ease-in-out;
}
1
2
3

4
5
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%).

2. Название анимации animation-name

Свойство задаёт имя анимации. Имя анимации создаётся в правиле @keyframes. Рекомендуется использовать название, отражающее суть анимации, при этом можно использовать одно или несколько слов, связанных между собой при помощи пробела - или символа нижнего подчеркивания _. Свойство не наследуется.

animation-name
Значения:
имя анимации Имя анимации, которое связывает правило @keyframes с селектором.
none Значение по умолчанию, означает отсутствие анимации. Также используется, чтобы отменить анимацию элемента из группы элементов, для которых задана анимация.
initial Устанавливает значение свойства в значение по умолчанию.
inherit Наследует значение свойства от родительского элемента.
Синтаксис

CSS

div {
animation-name: mymove;

```
}
1
2
3
div {
animation-name: mymove;
}
```

3. Продолжительность анимации animation-duration

Свойство устанавливает продолжительность анимации, задаётся в секундах или миллисекундах, отрицательные значения не допустимы. Не наследуется. Если для элемента задано более одной анимации, то можно установить разное время для каждой, перечислив значения через запятую.

animation-duration
Значения:
время Длительность анимации задается в секундах s или миллисекундах ms.
initial Устанавливает значение свойства в значение по умолчанию 0.
inherit Наследует значение свойства от родительского элемента.
Синтаксис

CSS

```
div {
animation-duration: 2s;
}
1
2
3
div {
animation-duration: 2s;
}
```

4. Временная функция animation-timing-function

Свойство определяет изменение скорости от начала до конца анимации с помощью временных функций. Задаётся при помощи ключевых слов или кривой Безье cubic-bezier(x1, y1, x2, y2). Не наследуется.

animation-timing-function
Значения:
ease Функция по умолчанию, анимация начинается медленно, разгоняется быстро и замедляется в конце. Соответствует cubic-bezier(0.25,0.1,0.25,1).
linear Анимация происходит равномерно на протяжении всего времени, без колебаний в скорости. Соответствует cubic-bezier(0,0,1,1).
ease-in Анимация начинается медленно, а затем плавно ускоряется в конце. Соответствует cubic-bezier(0.42,0,1,1).
ease-out Анимация начинается быстро и плавно замедляется в конце. Соответствует cubic-bezier(0,0,0.58,1).
ease-in-out Анимация медленно начинается и медленно заканчивается. Соответствует cubic-bezier(0.42,0,0.58,1).
cubic-bezier(x1, y1, x2, y2) Позволяет вручную установить значения от 0 до 1. На этом сайте вы сможете построить любую траекторию скорости изменения анимации.
step-start Задаёт пошаговую анимацию, разбивая анимацию на отрезки, изменения происходят в начале каждого шага. Эквивалентно steps(1, start).
step-end Пошаговая анимация, изменения происходят в конце каждого шага. Эквивалентно steps(1, end).
steps(количество шагов,start|end) Ступенчатая временная функция, которая принимает два параметра. Количество шагов задается целым положительным числом. Второй параметр необязательный, указывает момент, в котором начинается анимация. Со значением start анимация начинается в начале каждого шага, со значением end — в конце каждого шага с задержкой. Задержка вычисляется как результат деления времени анимации на количество шагов. Если второй параметр не указан, используется значение по умолчанию end.
initial Устанавливает значение свойства в значение по умолчанию.
inherit Наследует значение свойства от родительского элемента.
Синтаксис

CSS

```
div {
animation-timing-function: linear;
}
1
2
3
div {
animation-timing-function: linear;
}
```

С помощью пошаговой анимации можно создавать интересные эффекты, например, печатающийся текст или индикатор загрузки.

5. Анимация с задержкой animation-delay

Свойство игнорирует анимацию заданное количество времени, что даёт возможность по отдельности запускать каждую анимацию. Отрицательная задержка начинает анимацию с определенного момента внутри её цикла, т.е. со времени, указанного в задержке. Это позволяет применять анимацию к нескольким элементам со сдвигом фазы, изменяя лишь время задержки.

Чтобы анимация началась с середины, нужно задать отрицательную задержку, равную половине времени, установленном в animation-duration. Не наследуется.

animation-delay
Значения:
время Задержка анимации задается в секундах s или миллисекундах ms. Значение по умолчанию 0.
initial Устанавливает значение свойства в значение по умолчанию.
inherit Наследует значение свойства от родительского элемента.
Синтаксис

CSS

```
div {
```

animation-delay: 2s;
}
1
2
3
div {
animation-delay: 2s;
}

6. Повтор анимации animation-iteration-count

Свойство позволяет запустить анимацию несколько раз. Значение 0 или любое отрицательное число удаляют анимацию из проигрывания. Не наследуется.

animation-iteration-count
Значения:
числоС помощью целого числа задается количество повторов анимации. Значение по умолчанию 1. Дробные значения больше 1 будут воспроизводить анимацию, обрезая её на части следующей итерации.
infinite Анимация проигрывается бесконечно.
initial Устанавливает значение свойства в значение по умолчанию.
inherit Наследует значение свойства от родительского элемента.
Синтаксис

CSS

div {
animation-iteration-count: 3;
}
1
2
3
div {
animation-iteration-count: 3;
}
7. Направление анимации animation-direction

Свойство задает направление повтора анимации. Если анимация повторяется только один раз, то это свойство не имеет смысла. Не наследуется.

animation-direction
Значения:
alternate Анимация проигрывается с начала до конца, затем в обратном направлении.
alternate-reverse Анимация проигрывается с конца до начала, затем в обратном направлении.
normal Значение по умолчанию, анимация проигрывается в обычном направлении, с начала и до конца.
reverse Анимация проигрывается с конца.
initial Устанавливает значение свойства в значение по умолчанию.
inherit Наследует значение свойства от родительского элемента.
Синтаксис

CSS

```
div {
animation-direction: alternate;
}
1
2
3
div {
animation-direction: alternate;
}
```

8. Краткая запись анимации

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

CSS

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

1
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.

9. Проигрывание анимации animation-play-state

Свойство управляет проигрыванием и остановкой анимации. Остановка анимации внутри цикла возможна через использование этого свойства в скрипте JavaScript. Также можно останавливать анимацию при наведении курсора мыши на объект — состояние :hover. Не наследуется.

animation-play-state
Значения:
paused Останавливает анимацию.
running Значение по умолчанию, означает проигрывание анимации.
initial Устанавливает значение свойства в значение по умолчанию.
inherit Наследует значение свойства от родительского элемента.
Синтаксис

CSS

```
div:hover {
animation-play-state: paused;
}
```
1
2
3
div:hover {
animation-play-state: paused;
}

10. Состояние элемента до и после воспроизведения анимации animation-fill-mode

Свойство определяет порядок применения определенных в @keyframes стилей к объекту. Не наследуется.

animation-fill-mode
Значения:
none Значение по умолчанию. Состояние элемента не меняется до или после воспроизведения анимации.
forwards Воспроизводит анимацию до последнего кадра по окончанию последнего повтора и не отматывает ее к первоначальному состоянию.
backwards Возвращает состояние элемента после загрузки страницы к первому кадру, даже если установлена задержка animation-delay, и оставляет его там, пока не начнется анимация.
both Позволяет оставлять элемент в первом ключевом кадре до начала анимации (игнорируя положительное значение задержки) и задерживать на последнем кадре до конца последней анимации.
initial Устанавливает значение свойства в значение по умолчанию.
inherit Наследует значение свойства от родительского элемента.
Синтаксис

CSS

```
div {
animation-fill-mode: forwards;
}
1
2
3
div {
animation-fill-mode: forwards;
}
```

11. Множественные анимации

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

CSS

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

https://html5book.ru/css3-animation/

Правило @keyframes — позволяет пользователю описать анимацию CSS свойств в виде перечня ключевых кадров.

Синтаксис

```
@keyframes <имя анимации> {
  <селектор кадра 1> {
        <свойство 1>:<значение 1>;
        <свойство 2>:<значение 1>;
  }
  <селектор кадра 2> {
        <свойство 1>:<значение 2>;
        <свойство 2>:<значение 2>;
  }
  .....................
  <селектор кадра n> {
        <свойство 1>:<значение n>;
        <свойство 2>:<значение n>;
  }
}
```

- <имя анимации> — задает имя анимации в виде строки в одинарных кавычках (это имя потом используется при вызове анимации в качестве значения свойства animation-name)
- <селектор кадра> — задает селекторы которые определяют ключевые кадры. Они могут принимать значения:

  - from — внутри этого селектора описывается значения анимированных свойств в начальный момент анимации (также можно описать начальный кадр анимации, используя эквивалентный селектор 0%)
  - to — внутри этого селектора описывается значения анимированных свойств в конечный момент анимации (также можно описать конечный кадр анимации, используя эквивалентный селектор 100%)
  - <процент> — описывает значения анимированных свойств в данный момент времени (момент времени отсчитывается в процентном отношении от значения свойства animation-duration)
- <свойство> — анимируемое CSS свойство
- <значение> — значения анимированного свойств в момент, описываемый селектором кадра

Пример кода

```
div {
        top:50px;
        left:0px;
        animation-name:'movement';
        animation-duration:10s;
}
@keyframes 'movement' {
  from {
        top:50px;
        left:0px;
  }
  50% {
        top:150px;
        left:100px;
  }
  to {
        top:400px;
        left:300px;
  }
}
```

В примере описана анимация перемещения (изменение координат блока с течением времени). В первый момент анимации, описываемый селектором кадра from, координаты блока top:50px, left:0px. Следующий ключевой кадр

описывает значения свойств через 50% времени анимации (в данном примере - 5 секунд). Координаты блока станут top:150px, left:100px. В последний момент времени описываемый селектором кадра to (через 10с) блок получит абсолютные координаты top:400px, left:300px. Замечание! Анимация происходит не в три рывка, а плавно.

CSS-анімація появи

Як правило, стандартне використання анімації полягає в тому, щоб змінювати якісь елементи сайту плавно з плином часу. Але це ж занадто банально, чи не так? Тому тут ми поділимося з вами неординарним способом використання часу появи анімації.

Справа в тому, що реально зміни на сайті в шматку анімації або в її повному вигляді можуть бути практично миттєвими. Для цього ми задаємо два будь-яких ключових кадра, але при цьому використовуємо дуже маленький інтервал. Наприклад, він може бути дорівнює 0,001%. В такому випадку анімація CSS може статися миттєво. Це відмінно підійде для імітації будь-якої неонової вивіски. Вивіска ця буде блимати, а якщо анімація CSS при цьому буде ще й використовувати прозорість і властивість text-shadow, то вивіска вийде майже як справжня.

Наведемо приклад коду.

```
@keyframes toggleOpacity {
50% { opacity: 1;    /* Вимкнуто */
50.001% { opacity: 0.4;    /* Стан утримується на деякий час */
52.999% { opacity: 0.4;    /* Повернення */
53% { opacity: 1; }
}
```

Покращуємо функціонал кнопок

Якщо розглянути питання, якою незвичайною може бути CSS-анімація кнопки, то можна сказати, що варіантів прикрасити кнопки на сайтах безліч. Розглянемо один із прикладів. Ефект опуклої кнопки. Приклад коду - нижче.

```
<!DOCTYPE html>
<html>
<head>
<style>
.button {
padding: 15px 25px;
font-size: 24px
text-aling: center;
cursor: pointer;
outline: none;
color: #fff;
background-color: #4CAF50;
border: none;
border-radius: 15px;
box-shadow: 0.9px #999
}
.button:hover {background-color: #3e8e41}
.button:active {
background-color: #3e8e41;
box-shadow: 0.5px #666;
transform: translateY(4px);
}
</style>
</head>
<body>
<h2>Anymated Button – “Pressed Effect”</h2>
```

CSS-анімація при наведенні на фрагмент сайту

Чим динамічніше і сучасніше сайт, тим довше на ньому залишається користувач. Крім того, важливу роль відіграє також і інтерактивність. Безумовно, це так, але що ж може допомогти зробити сайт максимально інтерактивним?

Дуже непогано тут виглядає робота з дизайном елементів і фрагментів сайту при наведенні миші. Анімацію кнопок при наведенні ми розібрали вище, але крім цього можна "оживити" і всілякі шматки сайту, зробивши їх максимально стильними. Як і з усім іншим, головний принцип тут - не переборщити.

Отже, є чудова властивість transition, яке може приймати аж до чотирьох пов`язаних з нею властивостей.

```
.example {
transition: [transition-property] [transition-duration] [transition-timing-function]
[transition-delay];
}
```

Протягом певного часу CSS-анімація при наведенні на цей шматок коду буде змінюватися. Час визначається властивістю duration. Тобто при наведенні на який-небудь елемент у нього почне своєрідно змінюватися якась властивість, яку ми поставили в селекторі. Для нашого випадку це - .element (точка попереду). Трохи нижче наводиться приклад коду, коли при переході у контейнер div, який володіє псевдокласом .hover, змінюється фон з червоного на зелений.

```
div {
transition: background-color 0.5s ease;
background-color: red;
}
div:hover {
background-color: green;
}
```

При наведенні миші користувача анімація CSS змінюється не відразу, а з затримкою в деяку частку секунди, що створює цікавий ефект.

Крім того, якщо в попередньому прикладі ви задавали певне правило, за яким фон у елемента div повинен був змінюватися з червоного і переходити на зелений протягом 0,4 секунди, то слід врахувати, що використовуючи значення all, ви можете звернутися до всієї властивості відразу.

Приклад коду представимо в такий спосіб.

```
div {
transition: all 0.5s ease;
background: red;
padding: 10px;
}
div:hover {
background: green;
padding: 20px;
}
```

Як було показано, у властивостей padding і background відбудеться ефект переходу, який визначається властивістю transition. Вам слід врахувати, що ви можете через кому вказати цілий певний набір значень.

```
div {
transition: background 0.2s ease, padding 0.8s linear;
}
```

Анімація тексту

За допомогою коду може бути задана і чудова анімація тексту CSS. Це може бути і якась стаття, і великий заголовок, титул сайту.

Що ж, давайте спробуємо дізнатися, що таке анімація тексту CSS, і створимо її і текстові тіні. Можливо, ви бачили фільми жахів, в яких слова немов би вицвітають на якомусь похмуро-темному фоні. Спробуємо на загальному прикладі відтворити щось подібне.

Анімуємо текст в стилі фільмів жахів

Власне, сама ідея полягає в тому, щоб зробити текст, в якому букви були б трохи розмиті і оберталися б. Між буквами при цьому має бути простір. Ми будемо використовувати інтервали між буквами тіні, а також інтервал. Для того щоб реалізувати задум, нам буде потрібно скрипт Lettering.js авторства Дейва Руперта. Він потрібен, щоб обернути слова плюс букви в кілька тегів span. Спочатку потрібно обернути фрази в тег h2. Наведемо приклад коду.

```
<div class=”os-phrases”>
<h2>Sometimes it’s better</h2>
<h2>to hide</h2>
<h2>in order to</h2>
<h2>survive evil</h2>
<h2>Thanatos</h2>
<h2>This fall</h2>
<h2>Prepare</h2>
<h2>Refresh to replay</h2>
</div>
```

Потім обернемо всі слова в тегах h2 в тег span. Це станеться ось так.

```
$("#os-phrases > h2").lettering('words').children("span").lettering().children("span").lettering();
```

Виглядати це буде дещо громіздко, але нехай вас це не бентежить. Вийде досить божевільна структура.

```
<div class=”os-phrases” id=”os-phrases”>
<h2>
<span class=”word1”>
<span class=”char1”><span class=”char1”>S</span></span>
<span class=”char2”><span class=”char1”>o</span><span class=”char3”>
<span class=”char1”>m</span></span><!--…-->
</h2>
</span><!--/word1-->
<1--…--></h2><h2>
<!--…-->
</h2>
<!--…-->
</h2>
</div>
```

В результаті ми кожну з наявних у нас букв обернули в тег span. Їх вийшло дійсно багато. Але в коді вище наведений приклад досить спрощений. Цілком всю структуру ви зможете написати самі, і вона буде дещо більше. Залежить це також і від того, який текст ви будете використовувати.

Завершимо нашу роботу деякої невеликої стилізацією. Всі наші заголовки в наведеному прикладі будуть розташовуватися на всю ширину екрану. І займати вони будуть майже все вільне місце.

```
.os-phrases h2 {
font-famyly: ‘Dosis’, ‘Lato’, sans-serif;
font-size: 70px;
font-weight: 200;
height: 100%;
width: 100%;
overflow: hidden:
text-transform: uppercase;
padding: 0;
```

```
margin: 0;
position: absolute;
top: 0;
left: 0;
letter-spacing: 14px;
text-aling: center;
}
```

Оскільки ми збираємося розташувати всі наші букви по центру екрана, то для нашого контейнера нам також буде потрібно і flexbox. Наведемо приклад його коду.

```
.os-phrases h2;
.os-phrases h2 > span {
height: 100%;
display: flex;
flex-direction: row;
justify-content: center;
aling-items: center;
}
```

Тепер ми зробили так, що всі букви, які обгорнуті в клас span, який належить батьківському класу .os-phrases, будуть розташовуватися і закріплені саме по центру. Не забудьте додати трохи вільного місця, тобто сам міжбуквений інтервал.

```
.os-phrases h2 > span {
margin: 0 15px;
}
```

При цьому перша обгортка буде як би мати певні додаткові властивісті. Це властивість perspective. Воно дозволить нам зробити так, що ця ділянка як би буде виділятися, виходити на перший план.

Самі ж наші букви будуть прозорими, а інтервал анімації ми запустимо для них десь в 5,2 секунди. Нижче - приклад коду.

```
.os-phrases h2 > span > span > span {
display: inline-block;
color: hsla(0,0%,0%,0);
```

```
transform-style: preserve-3d;
transform: translate3d(0,0,0);
animation: OpeningSequence 5.2 linear forwards:
}
```

Важливо також визначити, як і з якою затримкою будуть з`являтися наші фрази. Яка з частин тексту буде з`являтися швидше, ніж попередня, і наскільки. Код буде виглядати наступним чином.

```
.os-phrases h2:nth-child(2) > span > span > span {
animation-delay: 5s;
}
.os-phrases h2:nth-child(3) > span > span > span {
animation-delay: 10s;
}
.os-phrases h2:nth-child(4) > span > span > span {
animation-delay: 15s;
}
.os-phrases h2:nth-child(4) > span > span > span {
animation-delay: 15s;
}
.os-phrases h2:nth-child(4) > span > span > span {
animation-delay: 15s;
}
.os-phrases h2:nth-child(5) > span > span > span {
font-size: 150px;
animation-delay: 21s;
animation-duration: 8s;
}
.os-phrases h2:nth-child(6) > span > span > span {
animation-delay: 30s;
}
.os-phrases h2:nth-child(7) > span > span > span {
animation-delay: 34s;
}
```

Надамо невеликий, але вельми цікавий ефект. Ставте opacity на 0,2. Ліва сторона символу при цьому буде досить велика. Букви також будуть повернені по осі Y. Також не забудьте додати ефект властивості text-shadow. На половині анімації ми зробимо незвичайну річ. Ми загостримо самі букви, а також

зменшимо відстань між ними, потім збільшимо непрозорість, а після повернемо знаки на 0 по осі Y.

В кінці ми знову додамо зникаючі букви і трохи змінемо їх маштаб. Це дасть невеликий ефект розмиття.

```
@keyframes OpeningSequence {
0% {
text-shadow: 0 0 50px #fff;
letter-spacing: 80px
opacity: 0.2;
transform: rotate(-90deg);
}
50% {
text-shadow: 0 01px #fff;
letter-spacing: 14px;
opacity: 0.8;
transform: rotate(0deg);
}
85% {
}
85% {
Text-shadow: 0 0 1px #fff:
Opacity: 0.8;
Transform: rotate(0deg) translateZ(100px);
}
100% {
Text-shadow: 0 0 10px #fff;
Opacity: 0;
Transform: translateZ(130px);
Pointer-events: none;
}
}
```

І нарешті остання фраза CSS-анімації.

```
.os-phrases h2:nth-child(8) > span > span > span {
font-size: 30px
animation: Fadeln 4s linear 40s forwards;
}
@keyframes Fadeln {
```

```
0% {
opacity: 0;
text-shadow: 0 0 50px #fff;
}
100% {
opacity: 0.8:
text-shadow: 0 0 1px #fff;
}
}
```

І так, наостанок додамо останній штрих. Зробимо акцент на деяких конкретних словах. У них буде жирне накреслення. Це і додасть необхідний акцент.

```
.os-phrases h2:first-child.word3,
.os-phrases h2:nth-child(2).word2,
.os-phrases h2:nth-child(4).word2 {
font-weight: 600;
}
```

Джерела:
http://htmlbook.ru - бібліотека, довідник по тегам та атрибутам HTML CSS
http://javascript.ru - довідник по мові програмування JavaScript