Тест 💎

1. Какими способами можно подключать CSS-стили? Найдите сами еще один способ, не указанный в уроке

* c помощью уже знакомого нам атрибута style - это удобно, если нужно что-то быстро протестировать или для короткого набора стилей. Это слово будет написано <span style="color:'red'">красным</span>
* c помощью тега <style>. В этом случае свойства CSS описываются в самом документе и располагаются в заголовке веб-страницы. Это позволяет хранить стили в одном месте, прямо на той же странице - *и этот способ гораздо более поддерживаемый, ведь так стили гораздо легче находить и при необходимости изменять*/
* c помощью тега <link> можно подключить стили, находящиеся в отдельном файле. *Это самый оптимальный способ, ведь тогда одни и те же стили можно использовать на разных страницах в случае, если их больше одной.*
* Еще один вариант — это подключение css к html при помощи директивы @import. Прописывается он в теге style.

<style>

@import url("/theme/style.css");

</style>

Можно указывать как абсолютный, так и относительный путь к файлу. Вот как выглядит на странице:

1. Зачем нужен Normalize.css?

* сохранять полезные настройки браузера, а не стирать их;
* нормализовать стили для широкого круга HTML-элементов;
* корректировать ошибки и основные несоответствия браузера;
* совершенствовать юзабилити незаметными улучшениями;
* объяснять код, используя комментарии и детальную документацию.

1. Что такое CSS-директивы?

Директивы — это конструкции, которая позволяет создавать в CSS инструкции для изменения отображения либо поведения элементов страницы. Директива начинается со знака @, за которым следует одно из служебных слов. Это общий синтаксис, которому следуют все директивы.

Обычные правила. Такие директивы следуют стандартному синтаксису:

@ [КЛЮЧЕВОЕ СЛОВО] (ПРАВИЛО);

* @charset

Эта директива определяет кодировку, используемую браузером. Это очень удобно, если таблица стилей содержит символы, не входящие в [ASCII](http://www.ascii.cl/htmlcodes.htm) (например [UTF-8](https://en.wikipedia.org/wiki/UTF-8)). Обратите внимание, что кодировка [указанная в HTTP-заголовке](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/meta#Attributes) переопределяет любые @charset, заданные в вашей таблице стилей.

* @import

Эта директива сообщает таблице стилей о необходимости запросить и включить в себя внешний CSS-файл, причём содержимое этого файла будет добавлено непосредственно в то место, где находится @import.

* @document

Эта директива определяет условия для стилей, которые применяются к конкретной странице. Например, можно задать URL страницы и затем настроить стили именно для неё. На других страницах эти стили будут проигнорированы.

* @font-face

Эта директива позволяет загружать пользовательские шрифты для использования на странице. Такие шрифты [по-разному поддерживаются браузерами](https://css-tricks.com/snippets/css/using-font-face/), однако директива поддерживает разные условия, которые позволяют подключать и использовать эти шрифты.

* **@keyframes**

Эта директива является базовым для создания [покадровой](https://css-tricks.com/snippets/css/keyframe-animation-syntax/) [анимации](https://css-tricks.com/almanac/properties/a/animation/), позволяя задавать начальные, конечные и промежуточные шаги CSS-анимации.

* @media

В этой директиве находятся условные выражения, применяющие определённые стили в зависимости от характеристик экрана. Эти выражения могут содержать, например, размеры экрана, что может пригодиться для создания [адаптивных стилей для различных устройств](https://css-tricks.com/snippets/css/media-queries-for-standard-devices/). Эта же директива позволит добавлять стили только при печати документаэ

* @page

Эта директива определяет стили для отдельных страниц при выводе на печать. Например, она содержит специальные псевдоэлементы для обращения к первой (:first), а также к левой (:left) и правой (:right) страницам при печати двух страниц на листе.

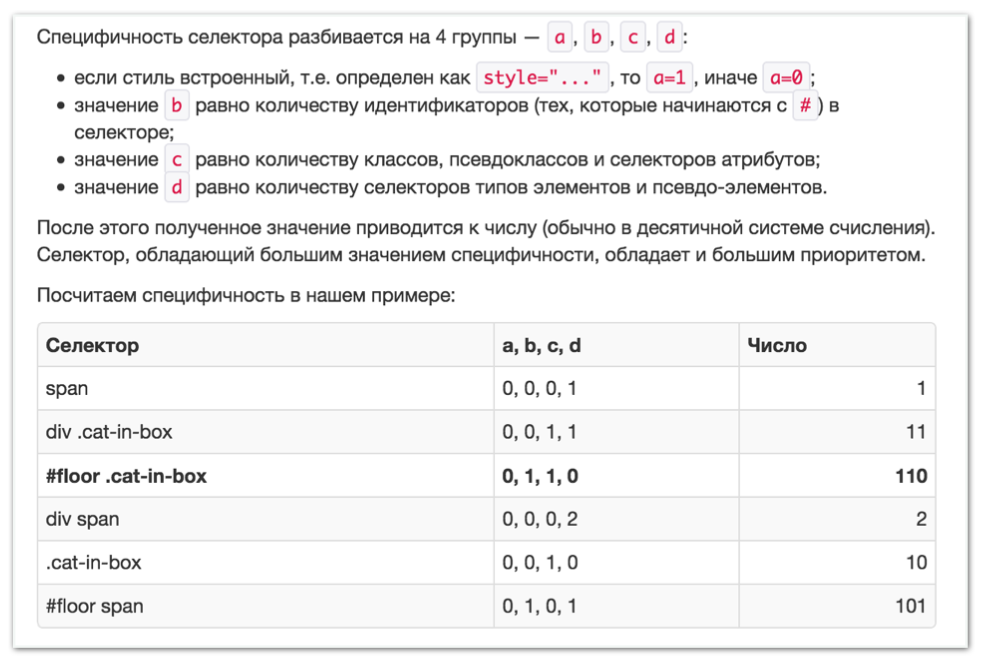
* @supports

Эта директива определяет поддерживает ли браузер те или иные возможности, и, если поддерживает, применяет к элементам заданные стили.

1. В чем разница между **margin** и **padding**?

Margin - внешний отступ (снаружи от границ блока до остальных элементов страницы)  
Padding - внутренний отступ (внутри от границ блока до контента)

1. Как в CSS определяются приоритеты? Какое из свойств будет приоритетнее - #link .main или span #login?



Приоритетнее #link .main (число 110), чем span #login (число 101).

1. В чем разница между CSS1 и CSS3?

* **Селекторы.**Селекторы лежат в основе каскадных таблиц стилей, и представляют собой шаблоны, используемые для выбора элементов HTML, которые необходимо стилизовать. CSS3 предлагает совершенно новый, надежный набор инструментов с расширенными селекторами, которые позволяют выбирать более конкретные элементы для стилизации, устраняя необходимость в тех произвольных идентификаторах и классах, которые вы так часто включаете. С таким широким набором селекторов вы можете нацелиться практически на любой элемент страницы. CSS3 расширяет базовую функциональность селектора атрибутов, позволяя выбирать элементы на основе строк в значениях атрибутов.
* **Цвет.**CSS3 предлагает несколько новых захватывающих способов игры с цветами. Раньше цвета объявлялись в шестнадцатеричном формате – системе нумерации, начинающейся с 00 и заканчивающейся FF. Спецификация цвета CSS3 определяет расширенный список ключевых слов цвета, которые поддерживаются веб-браузерами. Список теперь включает дополнительные 147 ключевых слов цвета и новую модель под названием «RGBA» (добавлен альфа-канал), которая позволяет определять непрозрачность цвета. Наиболее значительным изменением является то, что теперь можно объявлять полупрозрачные цвета.
* **Скругление границ блока.**Свойство CSS “border-radius” определяет закругленные углы любого элемента, что позволяет легко использовать закругленные углы в элементах дизайна. С помощью CSS3 вы даже можете использовать изображения для границ и создавать закругленные углы на границах без изображений или дополнительной разметки. Оно также имеет дополнительные свойства фона, такие как возможность иметь несколько фонов и улучшения для управления размером фонового изображения, ориентацией и обрезкой. Свойство “box shadow” позволяет добавлять тени к элементам.
* **Форматирование текста.**CSS3 предоставляет множество свойств форматирования текста, многие из которых уже присутствуют в CSS2, но с некоторыми дополнительными функциями, такими как новое свойство “text shadow”. Это новое свойство “text shadow” позволяет добавлять тени к отдельным символам в текстовых узлах. Эффект тени для текста делает веб-страницу более яркой и привлекательной. До CSS3 сделать то же самое, можно было используя изображение или дубликат текстового элемента, а затем необходимо было разместить его.
* **Градиенты.**Градиенты – это одна из самых ожидаемых функций, которые станут родными для CSS. Это невероятно мощные дополнения к CSS, позволяющие установить градиентный цвет фона элемента. Причём можно использовать простой двухцветный градиент, или можно сделать все возможное, используя множество цветовых точек и различные уровни непрозрачности по всему градиенту. Градиенты могут быть линейными или радиальными и создавать визуальные эффекты, которые так же визуально богаты, как градиенты в других технологиях, таких как SVG.
* **Переходы и анимация.**В CSS3 есть еще одна интересная функция, называемая “transitions”, которая позволяет управлять скоростью анимации при переходе от одного значения свойства CSS к другому. Например, можно анимировать высоту элемента от нуля до сотни пикселей, чтобы показать содержимое элемента. CSS3-анимации имеют свою собственную спецификацию, и они позволяют создавать ключевые кадры для управления анимацией и свойствами, которые позволяют управлять временем, длительностью и цикличностью анимации.

1. Что такое псевдоклассы? А псевдоэлементы?

Псевдоклассы определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также положение в дереве документа. Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на неё курсора мыши. При использовании псевдоклассов браузер не перегружает текущий документ, поэтому с помощью псевдоклассов можно получить разные динамические эффекты на странице.

Псевдоэлементы создают "виртуальные теги" и позволяют стилизовать их.

Синтаксис использования псевдоэлементов следующий:

селектор:псевдоэлемент { ... }

Каждый псевдоэлемент может применяться только к одному селектору, если требуется установить сразу несколько псевдоэлементов для одного селектора, правила стиля должны добавляться к ним по отдельности, как показано ниже.

.foo:first-letter { color: red; }

.foo:first-line { font-style: italic; }

1. Изучите статью про "плохие" теги <https://msiter.ru/tutorials/html-srednego-urovnya/plokhie-tegi> и пришлите список тегов, которые нежелательно использовать

* **<b>** отображает текст жирным шрифтом.
* **<i>** отображает текст курсивом
* **<big>** увеличивает размер отображаемого текста. выделения следует использовать свойство CSS [font-size](https://msiter.ru/references/css-reference/font-size" \o "Справка по свойству font-size), которое предоставляет больший контроль над страницей.
* **<small>** уменьшает размер отображаемого текста
* **<hr>**создает горизонтальную линию.
* **<u>**в свое время использовался для подчеркивания текста. Так получилось, что подчеркнутый текст прочно ассоциируется со ссылками. Именно поэтому этот тег давно умер, и именно поэтому крайне не желательно иметь в тексте подчеркнутые участки, не являющиеся ссылками.
* <center> использовался для центрирования одного элемента относительно другого. Свойство CSS [text-align](https://msiter.ru/references/css-reference/text-align" \o "Справка по свойству text-align) имеет не только значение ***center***, но и ***left***, ***right*** и ***justify***.
* *<***layer>**аналогичен [тегу <div>](https://msiter.ru/references/html-reference/div), однако работает он только в старых версиях браузера Netscape и потому совершенно бесполезен.
* **<blink>** или ***<marquee>*.** Наше твердое "Нет!". Они должны были делать мигающий текст, однако из-за ограниченной поддержки вскоре превратились в дурную шутку.
* ***<font>*** использовался для определения имени, размера и цвета шрифта элементов. Этот тег снискал репутацию злого гения мира тегов. Старые сайты (и даже некоторые новые) подобно оспинами буквально испещрены тегами *<font>*. Большинство из них обязаны своему появлению различным программам автоматизации создания сайтов, вставляющие тег *<font>* вокруг всякого элемента, размер или цвет которого изменяет автор. Так как этот тег приходится вставлять вокруг каждого появления элемента (скажем, каждый раз, как используется [элемент <p>](https://msiter.ru/references/html-reference/p)), то при помощи свойств CSS это можно сделать при помощи одной маленькой строки кода и применить ее ко всему сайту. Такой способ не только делает код веб-страницы меньше, но и позволяет при модифицировании стиля изменить всего одну строку, а не отыскивать все теги *<font>* в коде всех веб-страниц. Тег *<font>* и неуместное использование таблиц – две наиболее частые причины распухания кода веб-страниц.

1. Как можно подключать шрифты локально?

Для этого вам нужно скачать нужный шрифт с любого онлайн портала, которые предоставляют на выбор массу шрифтов.После этого, скачанные файлы со шрифтами положить в папку (условно ) fonts.

Для подключения шрифтов используется специальное ЦСС‑правило @font-face, которое мы видели в примере выше. Его нужно вставить в ЦСС‑файл на верхнем уровне:

@font-face {

font-family: 'Rubik';

font-style: normal;

font-weight: 400;

font-display: swap;

src: url(<путь к шрифту>) format('woff');}

В параметре src можно указать сразу несколько источников. Браузер по очереди пройдётся по ним и выберет первый подходящий, например:

@font-face {

font-family: 'Rubik';

font-style: normal;

font-weight: 400;

font-display: swap;

src: url(./fonts/Rubik-regular.woff2) format('woff2'),

url(./fonts/Rubik-regular.woff) format('woff'),

url(./fonts/Rubik-regular.ttf) format('truetype'),

url(./fonts/Rubik-regular.otf) format('opentype');}

В url нужно указать относительный путь к файлу со шрифтом или адрес удалённого ресурса. Дополнительно можно указать формат файла, чтобы дать браузеру подсказку, стоит ли его использовать.

Функцию local можно использовать, чтобы браузер поискал шрифт на машине пользователя и в случае успеха использовал его.

1. Почему не стоит использовать сокращенную запись без необходимости? И если все же использовать, как это делать правильно?

Если что-то в сокращенной записи не указано, оно пропишется по умолчанию. Если вы ранее определили одно из свойств шрифта и используете впоследствии свойство font, то оно перепишет заданные ранее значения. Так как сокращенная запись более приоритетная. Н-р:

body { font-size: 16px; line-height: 1.5; }

ul { font: 14px Georgia, serif; }

В этом свойстве font, line-height не было определено, поэтому потеряет значение своего предка 1.5 и вернётся к своему значению по умолчанию (обычно 1.2).

Чтобы избежать подобных проблем, следует, во-первых, группировать свойства по смыслу, это позволит быстрее находить ошибки, а во-вторых, если вам нужно переопределить значения ранее заданных свойств, не используйте сокращённую запись.

1. Разберитесь самостоятельно, как сделать анимацию через CSS

1. Ключевые кадры

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 }

CSS

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.

2. Название анимации: свойство animation-name

Свойство animation-name определяет список применяемых к элементу анимаций. Каждое имя используется для выбора ключевого кадра в правиле, которое предоставляет значения свойств для анимации. Если имя не соответствует ни одному ключевому кадру в правиле, нет свойств для анимации, отсутствует имя анимации, анимация не будет выполняться.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. Задержка анимации: свойство animation-delay

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

9. 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.

10. Краткая запись анимации: свойство animation

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

11. Множественные анимации

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