Используя шаблон HTML-страницы css3.html изучите различные стилевые свойства. К различным элементам страницы применяйте различные значения **одного и того же** свойства, чтоб увидеть отличия.

**Задание №1. Свойства текста**

Примените к элементам HTML-страницы свойства текста.

**text-align**, выравнивает текстовое содержание относительно элемента родителя или же окна браузера, значения:

* left – выровнять текст по левому краю элемента (по умолчанию).
* right – выровнять текст по правому краю.
* center – выровнять текст по центру.
* justify – выровнять текст по обоим краям.

p { text-align: center; }

**text-decoration**, позволяет декорировать текст, присвоив ему одно или несколько значений из ниже представленных вариантов оформления текста, значения:

* blink – текст будет мигать. Такой текст периодически, примерно раз в секунду исчезает, потом вновь появляется на прежнем месте. Это значение в настоящее время не поддерживается браузерами и осуждается в CSS3, взамен рекомендуется использовать анимацию.
* line-through – делает текст перечеркнутым.
* overline – надчеркивание текста.
* underline – подчеркивание текста.
* none – текст без оформления.

a { text-decoration: none; }

**text-indent**, задаёт отступ первой строки в текстовом блоке с левой стороны, проще говоря делает "красную строку". Расстояние от левого края окна браузера или же элемента родителя (блока в который помещен блок с текстом) может быть задано в процентах от ширины окна браузера, а также любых других единицах измерения CSS.

p { text-indent: 20px; }

**text-transform**, трансформирует символы в указанном текстовом блоке, делая их заглавными или прописными по одному из правил в зависимости от присужденного значения данному свойству, значения:

* none – текст отображается без каких-либо изменений.(по умолчанию).
* capitalize – каждое слово в тексте отображается с заглавного символа.
* lowercase – все символы преобразуются в нижний регистр.
* uppercase – все символы преобразуются в верхний регистр.

p { text-transform: lowercase; }

**vertical-align**, вертикальное выравнивание текста в строке, значения:

* baseline – выравнивает базовую линию элемента по базовой линии родителя.
* bottom – выравнивает элемент по нижней части строки.
* middle – выравнивает средину элемента по базовой линии родителя и прибавляет половину высоты родительского элемента.
* sub – нижний индекс (размер шрифта не меняется).
* super – верхний индекс (размер шрифта не меняется).
* text-bottom – нижняя граница элемента выравнивается по нижнему краю строки.
* text-top – верхняя граница элемента выравнивается по верхнему краю строки.
* top – выравнивает элемент по верхней части строки.

Так же вертикальное выравнивание элемента относительно строки может выражаться в процентах, а также любых других единицах измерения CSS, причем эти единицы могут принимать как положительные, так и отрицательные значения.

.align1 { vertical-align: +5px; }

.align2 { vertical-align: bottom; }

.align3 { vertical-align: top; }

.align4 { vertical-align: 50%; }

**white-space**, имитирует работу тега <pre>, определяя показывать или нет пробелы между символов, если таковых больше чем один, а так же разрешает или запрещает перенос строки, значения:

* normal – текст выводится как обычно (лишние пробелы убираются), переносы строк определяются автоматически. (по умолчанию)
* nowrap – запрещает автоматический перенос строки.
* pre – показывает текст в том виде, в котором он был набран. Пробелы и переносы строки не удаляются.

p { white-space: pre; }

**word-spacing**, задаёт расстояние между словами (группами символов не разделенными пробелом) в строке, значения:

* normal – нормальное расстояние (по умолчанию).
* px – расстояние задаётся в пикселях, а также любых других единицах измерения CSS.

p { word-spacing: 10px; }

**letter-spacing**, определяет расстояние между символами в тексте, значения:

* normal – нормальное расстояние (по умолчанию).
* px – расстояние задаётся в пикселях, а также любых других единицах измерения CSS.

p { letter-spacing: 5px; }

**line-height**, задает расстояние между строками текста, значения:

* normal – норма (по умолчанию).
* % – проценты. За сто процентов берется высота шрифта
* 0.5 – множитель. Может быть использовано любое число больше ноля. Так, например множитель 0.5 будет равняться половинному межстрочному расстоянию, а 2 – двойному.
* px – расстояние задаётся в пикселях, а также любых других единицах измерения CSS.

p { line-height: 150%; }

**text-shadow**, задает тень текста, значения:

* Первое значение – определяет сдвиг тени относительно текста по горизонтали, указывается в пикселях, может иметь как положительное, так и отрицательное значение, является обязательным параметром.
* Второе значение – определяет смещение тени относительно текста уже по вертикали, тоже может быть как положительным, так и отрицательным и так же является обязательным параметром.
* Третье значение – делает тень тенью, а не просто сдвинутой копией текста – оно определяет радиус сглаживания тени, делая её более чёткой или же наоборот размытой. Чем большее значение имеет данный параметр, тем больше расфокусируется отбрасываемая от текста тень. Данный параметр не является обязательным и его можно не указывать в этом случае размытость тени по умолчанию примет 0.
* Четвёртое значение – задаёт цвет тени, тоже необязательный параметр и в случае если он не указан тень будет того же цвета что и сам текст.

p { text-shadow: 10px 5px 2px; }

**word-wrap**, позволяет переносить длинные слова и прочие группы символов, неразделённые пробелами на следующую строку, если такое слово не помещается в блок с фиксировано заданной шириной, значения:

* normal – перенос строки не происходит там, где нет пробелов или переносов строк.
* break-word – перенос строки происходит в тех местах, где это становится необходимо.
* inherit – наследует значение от блока родителя.

p { word-wrap: break-word; }

**Задание №2. Свойства шрифта**

Примените к элементам HTML-страницы свойства шрифта.

**font-style**, в зависимости от выбранного значения, определяет стиль шрифта, значения:

* normal – обычный (по умолчанию)
* italic – курсив (своего рода шрифт, взятый из библиотеки шрифтов)
* oblique – наклонный (результат работы алгоритма, где каждый символ слегка наклоняется в правую сторону)

p { font-style: italic; }

**font-variant**, позволяет делать строчные буквы заглавными и уменьшенными, значения:

* normal – обычный (по умолчанию)
* small-caps – все буквы заглавные и уменьшенные

p { font-variant: small-caps; }

**font-size**, определяет размер шрифта, может быть задан в процентах, а также любых других единицах измерения CSS.

Значения абсолютного размера шрифта:

* xx-small – очень очень маленький
* x-small – очень маленький
* small – маленький
* medium – средний
* large – большой
* x-large – очень большой
* xx-large – очень очень большой

p { font-size: medium; }

Значения относительного размера шрифта:

* larger – больше чем размер шрифта родительского элемента
* smaller – меньше чем размер шрифта родительского элемента

p { font-size: larger; }

**font-weight**, определяет жирность шрифта, значения:

* normal – обычный шрифт
* bold – полужирный шрифт
* bolder – жирный шрифт
* lighter – тонкий шрифт

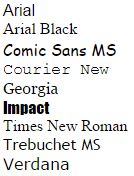
p { font-weight: lighter; }

А также выражается в условном числовом значении от 100 до 900 с шагом 100 (100, 200, 300... 900) где значение 100 тонкий шрифт, а 900 – сверх жирный.

p { font-weight: 600; }

**font-family**, указывает одно, два или три имени шрифта из библиотеки шрифтов. Возможность указывать до трёх имен шрифтов через запятую используется разработчиками во избежание возможных проблем связанных с отсутствием, по тем или иным причинам, указанных имен в библиотеке шрифтов на компьютере пользователя. Если же браузер не найдёт в библиотеке шрифтов пользователя ни одного шрифта из указанных, то он будет использовать тот шрифт который указан в его настройках "по умолчанию", список самых используемых шрифтом следующий:

* Arial
* Arial Black
* Comic Sans MS
* Courier New
* Georgia
* Impact
* Times New Roman
* Trebuchet MS
* Verdana



Однако также можно указать браузеру не только какой-то конкретный шрифт, но и обозначить предпочтительное семейство шрифтов:

* serif – шрифты с засечками
* sans-serif – рубленые шрифты
* cursive – курсивные шрифты
* fantasy – декоративные шрифты
* monospace – моношириные шрифты

p { font-family: Times New Roman, sans-serif; }

**font**, являясь базовым атрибутом, может определять одновременно сразу несколько параметров шрифта принимая те или иные значения от следующих атрибутов:

* font-style
* font-variant
* font-weight
* font-size
* font-family

p {

font-style: italic;

font-weight: bolder;

font-size: 20px;

font-family: Arial;

}

p { font: italic bold 20px Arial; }

Обе записи идентичны.

**Задание №3. Свойство прозрачности**

Примените к элементам HTML-страницы свойства прозрачности.

**opacity**, позволяет делать прозрачным элемент сайта. Степень прозрачности элемента определяется значением от 0 до 1, где 0 – полностью прозрачный, а 1 – полностью непрозрачный. Так, например значение 0.5 свойства opacity применённого к картинке будет значить, что данное изображение должно быть полупрозрачным.

.opacity1 { opacity: 0; }

**Задание №4. Свойства цвета и фона**

Примените к элементам HTML-страницы свойства цвета и фона.

**color**, позволяет перекрасить текст, какого либо элемента в нужный нам цвет, значения:

* #ff0000 – шестнадцатеричное значение цвета RGB
* red – именное значение цвета
* rgb(255, 0, 0) – значение цвета RGB
* hsl(120, 10%, 50%) – значение цвета HSL
* rgba(255, 0, 0, 1) – значение цвета RGB с альфа каналом
* hsla(120, 10%, 50%, 0.5) – значение цвета HSL с альфа каналом

h1 { color: red; }

**background-color**, определяет цвет фона элемента, значения:

* #ff0000 – шестнадцатеричное значение цвета RGB
* red – именное значение цвета
* rgb(255, 0, 0) – значение цвета RGB
* hsl(10, 20%, 50%) – значение цвета HSL
* rgba(255, 255, 0, 1) – значение цвета RGB с альфа каналом
* hsla(120, 10%, 60%, 0.5) – значение цвета HSL с альфа каналом
* transparent – прозрачный фон (по умолчанию)

h1 { background-color: white; }

**background-image**, присваивает фоновое изображение элементу, значения:

* url – путь к файлу с изображением
* none – изображение отсутствует (по умолчанию)

body { background-image: url(fon.jpg); }

Если рисунок не заполняет собой весь фон элемента, то он выкладывается "плиткой".

В СSS3 появилась возможность одновременно использовать сразу несколько фоновых изображений для одного и того же элемента.

body { background-image: url(fon1.jpg), url(fon2.jpg), url(fon3.jpg); }

Изображение что идёт в списке первым, будет находиться поверх других, остальные же будут подкладываться под него в том порядке, в котором они перечислены.

При использовании в качестве фона нескольких изображений к ним так же можно применять дополнительные свойства для работы с фоном такие как: background-attachment, background-repeat, background-position и т.д. Значения для этих свойств, в случае если картинок несколько, тоже же указывается через запятую, где первое значение предназначено для первого фона второе для второго и т.д.

body {

background-image: url(fon1.jpg), url(fon2.jpg), url(fon3.jpg);

background-repeat: no-repeat, repeat-x, repeat;

background-position: center 100px, 0px 250px;

}

Если на странице или в каком либо блоке присутствует полоса прокрутки, то фоновое изображение будет прокручиваться вместе с остальным содержанием данного блока.

**background-attachment**, позволяет зафиксировать фоновое изображение, значения:

* fixed – фиксированный фон
* scroll – подвижный фон (по умолчанию)

body {

background-image: url(fon.jpg);

background-attachment: fixed;

}

Как уже говорилось выше, если рисунок не заполняет собой весь фон элемента, то он повторяется и выкладывается "плиткой".

**background-repeat**, регулирует повторение фонового изображения, значения:

* no-repeat – запретить повторение, показать только одно изображение
* repeat – повторять изображение (по умолчанию)
* repeat-x – повторять только по горизонтали
* repeat-y – повторять только по вертикали
* space – между фоновыми изображениями выкладываемых «плиткой» равномерно добавляются пустые пространства, так чтобы в блоке поместилось максимальное количество целых рисунков
* round – дублируемые фоновые изображения масштабируются до тех пор, пока крайние рисунки целиком не впишутся в заданные размеры

body {

background-image: url(fon.jpg);

background-repeat: repeat-x;

}

**background-position**, позволяет управлять позицией фонового изображения в элементах где оно задано с помощью background-image.

Позицию фонового изображения, а точнее его верхнего левого угла, можно задать в процентах, пикселях, а также любых других единицах измерения CSS, с помощью двух значений, где первое значение будет являться отступом от левой границы элемента, а второе значение отступом от верхней границы элемента.

Например, запись: background-position: 200px 100px; будет обозначать, что фоновое изображение будет смещено по горизонтали на 200 пикселей от левой границы элемента и по вертикали на 100 пикселей от верхней границы элемента.

Так же можно использовать следующие значения:

по горизонтали:

* left – расположить слева
* center – расположить по центру
* right – расположить справа

по вертикали:

* top – расположить сверху
* center – расположить по центру
* bottom – расположить снизу

body {

background-image: url(fon.jpg);

background-position: right bottom;

}

**background-size**, указывает размер изображения выступающего в качестве фона, значения:

* px – в пикселях, а также любых других единицах измерения CSS
* % – в процентах от размеров элемента, которому назначается фон
* cover – растягивает фоновое изображение на весь блок
* contain – растягивает фоновое изображение с сохранением пропорций до тех пор, пока его ширина или высота не упрётся в края блока
* auto – исходный размер изображения (по умолчанию). В случае если значение auto присвоено только одной из сторон, то её размер вычисляется относительно размера второй стороны с указанным размером

.fon1 { background-size: 200px 50px; }

.fon2 { background-size: 20% 80%; }

.fon3 { background-size: contain; }

.fon4 { background-size: cover; }

.fon5 { background-size: auto 50px; }

.fon6 { background-size: auto auto; }

**background-origin**, можно определять область позиционирования фона в том или ином блоке, значения:

* padding-box – фон позиционируется от краёв блока
* border-box – фон позиционируется от границ блока
* content-box – фон позиционируется от полей блока

Свойство background-origin не применяется, если значение background-attachment задано как fixed.

.fon1 { background-origin: border-box; }

**background**, позволяет установить одновременно до пяти характеристик фона. Значения могут идти в любом порядке, браузер сам определит, какое из них соответствует нужному свойству. В CSS3 допустимо указывать параметры сразу нескольких фонов, перечисляя их через запятую.

div { background: #ff0 url(fon.jpg) fixed repeat-x center; }

Работая одновременно с несколькими фонами CSS, код можно формировать с помощью базового свойства background следующим образом:

body {

background: url(fon1.jpg) no-repeat center 100px,

url(fon2.jpg) repeat-x 0px 250px,

url(fon3.jpg) repeat;

}

А можно:

body {

background-image: url(fon1.jpg), url(fon1.jpg), url(fon1.jpg);

background-repeat: no-repeat, repeat-x, repeat;

background-position: center 100px, 0px 250px;

}

В первом случае CSS код выглядит более понятным.

**Градиентом** называют плавный переход от одного цвета к другому, причём самих цветов и переходов между ними может быть несколько. С помощью градиентов создаются самые причудливые эффекты веб-дизайна, например, псевдотрёхмерность, блики, фон и др. Также с градиентом элементы смотрятся более симпатично, чем однотонные. Отдельного свойства для добавления градиента нет, поскольку он считается фоновым изображением, поэтому добавляется через свойство background-image или универсальное свойство background.

div { background: linear-gradient(to top, #ff0, #0ff); }

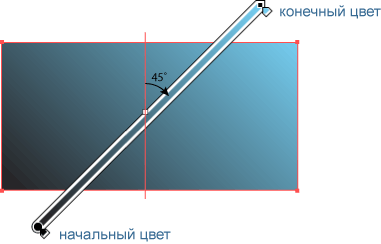
В самом простом случае с двумя цветами вначале пишется позиция, от которой будет начинаться градиент, затем начальный и конечный цвет.



Для записи позиции вначале пишется **to**, а затем добавляются ключевые слова **top**, **bottom** и **left**, **right**, а также их сочетания. Порядок слов не важен, можно написать to left top или to top left. Вместо ключевого слова допускается задавать угол наклона градиентной линии, который показывает направление градиента. Вначале пишется положительное или отрицательное значение угла, затем к нему слитно добавляется **deg**.



Нулю градусов (или 360) соответствует градиент снизу вверх, далее отсчёт ведётся по часовой стрелке. Отсчёт угла наклона градиентной линии показан ниже.



Для создания сложных градиентов двух цветов уже будет недостаточно, синтаксис позволяет добавлять их неограниченное количество, перечисляя цвета через запятую. При этом можно использовать прозрачный цвет (ключевое слово transparent), а также полупрозрачный с помощью формата RGBA или HSLA.

div { background: linear-gradient(to top, rgba(30,87,153,0), rgba(30,87,153,0.8), rgba(30,87,153,1), rgba(30,87,153,1), rgba(41,137,216,1), rgba(30,87,153,1), rgba(30,87,153,1), rgba(30,87,153,0.8), rgba(30,87,153,0)); }

Чтобы точно позиционировать цвета в градиенте, после значения цвета указывается его положение в процентах, пикселах или других единицах. Например, запись red 0%, orange 50%, yellow 100% означает, что градиент начинается с красного цвета, затем на 50% переходит в оранжевый, а затем до конца в жёлтый. Для простоты крайние единицы вроде 0% и 100% можно не писать, они подразумеваются по умолчанию.

div { background: linear-gradient(to top, #b5bdc8 0%, #828c95 36%, #28343b 100%); }

**Задание №5. Свойства границ элементов**

Примените к элементам HTML-страницы свойства границ элементов.

**border-style**, присваивает элементу один из стилей границы, значения:

* none – граница отсутствует (по умолчанию)
* dotted – граница из ряда точек
* dashed – пунктирная граница
* solid – сплошная граница
* double – двойная граница
* groove – граница "бороздка"
* ridge – граница "гребень"
* inset – вдавленная граница
* outset – выдавленная граница

div { border-style: dotted; }

Стиль бордюра может быть задан как для всех сторон элемента одновременно, так и для каждой его стороны отдельно в зависимости от того, сколько значений присвоено свойству border-style. Таковых значений может быть от одного до четырёх по числу сторон элемента. В каждом из четырёх случаев действуют свои "правила" по присуждению стиля рамки той или иной стороне элемента:

Одно значение – устанавливает единый стиль бордюра для всех сторон элемента.

Два значения:

* Первое значение – устанавливает стиль верхней и нижней границы элемента
* Второе значение – устанавливает стиль левой и правой границы элемента

div { border-style: solid double; }

Три значения:

* Первое значение – устанавливает стиль верхней границы элемента
* Второе значение – устанавливает стиль левой и правой границы элемента
* Третье значение – устанавливает стиль нижней границы элемента

div { border-style: solid double dashed; }

Четыре значения:

* Первое значение – устанавливает стиль верхней границы элемента
* Второе значение – устанавливает стиль правой границы элемента
* Третье значение – устанавливает стиль нижней границы элемента
* Четвёртое значение – устанавливает стиль левой границы элемента

div { border-style: solid double dashed ridge; }

**border-width**, устанавливает ширину границы элемента, значения:

* thin – тонкая граница
* medium – средняя толщина границы
* thick – толстая граница
* px – в пикселях, а также любых других единицах измерения CSS

div {

border-style: solid;

border-width: 1px;

}

По аналогии со стилем, толщина бордюра тоже может иметь от одного до четырёх значений и в каждом случае устанавливается для тех или иных сторон бордюра, правила совпадают со стилем бордюра.

.border1 {

border-style: solid;

border-width: 1px 3px;

}

.border2 {

border-style: solid;

border-width: 1px 2px 1px;

}

.border3 {

border-style: solid;

border-width: 1px 2px 3px 4px;

}

**border-color**, устанавливает цвет границы элемента, значения:

* #ff0000 – шестнадцатеричное значение цвета RGB.
* red – именное значение цвета
* rgb(255, 0, 0) – значение цвета RGB
* hsl(10, 20%, 50%) – значение цвета HSL
* rgba(255, 255, 0, 1) – значение цвета RGB с альфа каналом
* hsla(120, 10%, 60%, 0.5) – значение цвета HSL с альфа каналом
* transparent – прозрачный фон (по умолчанию)

div {

border-style: solid;

border-color: #008000;

}

По аналогии со стилем, цвет бордюра тоже может иметь от одного до четырёх значений и в каждом случае устанавливается для тех или иных сторон бордюра, правила совпадают со стилем бордюра.

.border1 {

border-style: solid;

border-color: #008000 #0000ff;

}

.border2 {

border-style: solid;

border-color: #008000 #0000ff #ff0000;

}

.border3 {

border-style: solid;

border-color: #008000 #0000ff #ff0000 #ffff00;

}

Для того, что бы определить стиль, цвет и ширину бордюра для одной из сторон элемента, используются свойства **border-bottom**, **border-left**, **border-right**, **border-top** и их дочерние свойства:

border-bottom – определяет стиль, цвет и ширину нижней границы элемента

* **border-bottom-color** – устанавливает цвет нижней границы элемента
* **border-bottom-style** – определяет стиль нижней границы элемента
* **border-bottom-width** – определяет ширину нижней границы элемента

border-left – определяет стиль, цвет и ширину левой границы элемента

* **border-left-color** – устанавливает цвет левой границы элемента
* **border-left-style** – определяет стиль левой границы элемента
* **border-left-width** – определяет ширину левой границы элемента

border-right – определяет стиль, цвет и ширину правой границы элемента

* **border-right-color** – устанавливает цвет правой границы элемента
* **border-right-style** – определяет стиль правой границы элемента
* **border-right-width** – определяет ширину правой границы элемента

border-top – определяет стиль, цвет и ширину верхней границы элемента

* **border-top-color** – устанавливает цвет верхней границы элемента
* **border-top-style** – определяет стиль верхней границы элемента
* **border-top-width** – определяет ширину верхней границы элемента

div { border-left: 10px solid #008000; }

**border**, базовый атрибут одновременно определяет стиль, цвет и толщину границы элемента, значения родственных свойств указываются в любом порядке через пробел.

div { border: RGB(25, 125, 25) 6px ridge; }

Если необходимо присвоить разные свойства различным сторонам границы элемента или только одной из них, используют свойства border-bottom, border-left, border-right, border-top.

**border-collapse**, определяет стиль отображения границ таблицы. По умолчанию каждая ячейка таблицы имеет собственную рамку (если конечно использовано свойство border), в местах соприкосновения ячеек образуется двойная линия, border-collapse заставляет браузер анализировать таковые места и поступать с ними согласно присвоенному значению данному свойству, значения:

* separate – ячейки таблицы отделены друг от друга (по умолчанию)
* collapse – ячейки таблицы не имеют промежутков между собой
* inherit – свойства наследуются у родителя элемента (работает далеко не во всех браузерах)

Обратите внимание, свойство border-collapse применяется только к тегу <table> и элементам, у которых свойство display имеет значение table или inline-table (свойство display будет рассмотрено позже).

table { border-collapse: collapse; }

**border-radius**, позволяет закруглять углы любого блока. Блок может быть с бордюром, либо фоновым рисунком – в любому случае border-radius сделает круглые углы без дополнительных рисунков. В качестве значений указываются любые единицы измерения CSS. В случае применения процентов, отсчет ведется относительно ширины блока. Разрешается использовать одно, два, три или четыре значения, перечисляя их через пробел:

Одно значение – радиус указывается для всех четырех уголков.

div { border-radius: 40px; }

Два значения:

* Первое значение – радиус верхнего левого и нижнего правого уголка
* Второе значение – радиус верхнего правого и нижнего левого уголка

div { border-radius: 40px 10px; }

Три значения:

* Первое значение – радиус для верхнего левого уголка
* Второе значение – радиус верхнего правого и нижнего левого уголка
* Третье значение – радиус нижнего правого уголка

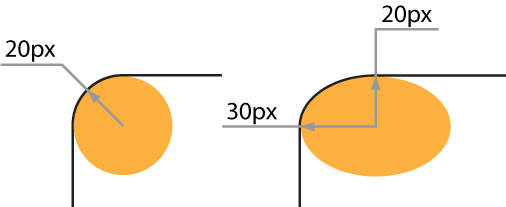
div { border-radius: 40px 10px 30px; }

Четыре значения:

* Первое значение – радиус для верхнего левого уголка
* Второе значение – радиус для верхнего правого уголка
* Третье значение – радиус для нижнего правого уголка
* Четвёртое значение – радиус для нижнего левого уголка

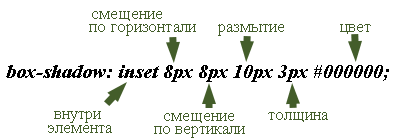
div { border-radius: 40px 10px 30px 5px; }

Также допустимо писать два значения через слэш(/), первый задает радиус по горизонтали, а второй по вертикали (эллиптические уголки).



div { border-radius: 10em/1em; }

**box-shadow**, добавляет тень к элементу. Допускается использовать несколько теней, указывая их параметры через запятую, при наложении теней первая тень в списке будет выше, последняя ниже. Если для элемента задается радиус скругления через свойство border-radius, то тень также получится с закругленными уголками. Добавление тени увеличивает ширину элемента, поэтому возможно появление горизонтальной полосы прокрутки в браузере, значения:



* none – отменяет добавление тени
* inset – тень выводится внутри элемента. Необязательный параметр
* сдвиг по x – смещение тени по горизонтали относительно элемента. Положительное значение этого параметра задает сдвиг тени вправо, отрицательное – влево. Обязательный параметр
* сдвиг по y – смещение тени по вертикали относительно элемента. Положительное значение задает сдвиг тени вниз, отрицательное — вверх. Обязательный параметр
* размытие – задает радиус размытия тени. Чем больше это значение, тем сильнее тень сглаживается, становится шире и светлее. Если этот параметр не задан, по умолчанию устанавливается равным 0, тень при этом будет четкой, а не размытой
* толщина – положительное значение растягивает тень, отрицательное, наоборот, ее сжимает. Если этот параметр не задан, по умолчанию устанавливается 0, при этом тень будет того же размера, что и элемент
* цвет – цвет тени в любом доступном CSS формате, по умолчанию тень черная. Необязательный параметр

div { box-shadow: 0 0 10px rgba(0,0,0,0.5); }

**Задание №6. Свойства размеров элементов**

Примените к элементам HTML-страницы свойства размеров элементов.

**width** и **height**, устанавливают ширину и высоту блочных элементов, значения:

* auto – размеры элемента определяется его содержанием (по умолчанию)
* % – размеры элемента задаются в процентах от высоты/ширины элемента родителя
* px – размеры элемента задаются в пикселях, а также любых других единицах измерения CSS

div {

height: 100px;

width: 250px;

}

Если элементу присвоены точные значения высоты и ширины (height, width) а его содержание, например длинный текст, не вмещается в указанных пределах, то по умолчанию такой элемент растягивается до нужных размеров, что не всегда соответствует задумке. В данном случае выручит свойство **overflow**, которое указывает браузеру, что делать с элементом в таких случаях, значения:

* visible – элемент растягивается до необходимых размеров (по умолчанию)
* hidden – содержание элемента "обрезается", видна лишь та его часть что помещается в элементе
* scroll – добавляются полосы прокрутки (всегда! даже если содержание помещается в пределах элемента)
* auto – полосы прокрутки добавляются при необходимости.

div { overflow: auto; }

Так как размеры элемента по умолчанию регулируются исключительно вложенным в него содержанием, текстом, графикой и т.д. то элементы становятся, вернее, остаются "резиновыми". Иногда требуется ограничить эту "резиновость" конкретными размерами. Для этого есть следующие свойства:

* **max-height** – определяет максимальную высоту элемента
* **max-width** – определяет максимальную ширину элемента
* **min-height** – определяет минимальную высоту элемента
* **min-width** – определяет минимальную ширину элемента

div {

min-height: 50px;

min-width: 100px;

max-height: 250px;

max-width: 300px;

}

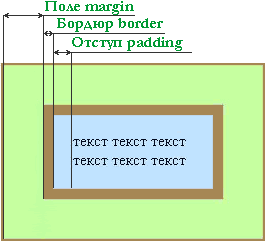
**Задание №7. Свойства полей и отступов элементов**

Примените к элементам HTML-страницы свойства полей и отступов элементов.

**margin**, поле, устанавливает расстояние от внешней границы элемента до границы окна браузера или элемента родителя.

**padding**, отступ, устанавливает расстояние от внутренней границы элемента до его содержания (текста, картинок и т.д.).

На рисунке наглядно показаны эти расстояния, а так же место занимаемое бордюром (border) про него тоже не следует забывать.



Значения margin и padding:

* px – в пикселях, а также любых других единицах измерения CSS
* % – в процентах
* auto – размер полей и отступов автоматически рассчитывается браузером

div {

margin: 40px;

padding: 20px;

}

По аналогии со свойствами бордюра, свойства margin и padding тоже могут иметь от одного до четырёх значений и в каждом случае устанавливаются для тех или иных сторон, правила совпадают с бордюром.

div { margin: 5px; }

div { margin: 5px 40px; }

div { margin: 5px 40px 20px; }

div { margin: 5px 40px 20px 1px; }

Свойства margin и padding являются базовыми и имеют ряд дочерних свойств:

* **margin-top** – поле от верхней границы элемента родителя
* **margin-left** – поле от левой границы элемента родителя
* **margin-right** – поле от правой границы элемента родителя
* **margin-bottom** – поле от нижней границы элемента родителя
* **padding-top** – отступ от верхней границы элемента до его содержания
* **padding-left** – отступ от левой границы элемента до его содержания
* **padding-right** – отступ от правой границы элемента до его содержания
* **padding-bottom** – отступ от нижней границы элемента до его содержания

div {

margin-left: 40px;

padding-right: 50px;

}

**Задание №8. Свойства курсора**

Примените к элементам HTML-страницы свойства курсора.

**cursor**, позволяет установить нестандартный вид курсора для элемента – блока, текста, рисунка и т.д. То есть когда пользователь наведет курсор на такой элемент, он поменяет свой вид, значения:

* auto – курсор назначается автоматически в зависимости от специфики элемента (по умолчанию)
* crosshair – перекрестие
* default – встроенный (основной)
* e-resize – стрелка на "восток"
* hand – указатель аналогия pointer
* help – помощь
* move – перемещение
* n-resize – переместить на "север"
* ne-resize – переместить на "северо-восток"
* nw-resize – переместить на "северо-запад"
* pointer – указатель
* progress – продолжение операции
* s-resize – переместить на "юг"
* se-resize – переместить на "юго-восток"
* sw-resize – переместить на "юго-запад"
* text – текст
* w-resize – переместить на "запад"
* wait – ожидание

Кроме того некоторые браузеры поддерживают дополнительные формы курсоров:

* all-scroll – переместить во все стороны
* col-resize – переместить по горизонтали
* no-drop – указатель "нет доступа"
* not-allowed – нет доступа
* row-resize – переместить по вертикали
* vertical-text – вертикальный текст

Данные значения являются не каким-либо конкретным изображением курсора, а определяют тип курсора. Внешний вид курсора зависит от настроек операционной системы пользователя.

div { cursor: wait; }

**Задание №9. Свойства форматирования**

Примените к элементам HTML-страницы свойства форматирования.

**display**, указывает браузеру, как тот или иной элемент должен быть показан на странице, другими словами определяет параметры вывода браузером элемента на экран, основные значения:

* block – показывает элемент как блочный. Происходит перенос строк вначале и в конце элемента
* inline – показывает элемент как встроенный. Элемент не переносятся на следующую строку
* none – запрещает показ элемента. Элемент "удаляется" из общего потока, его занимаемое пространство на экране не резервируется

p { display: none; }

**visibility**, предназначено для отображения или скрытия элемента, включая рамку вокруг него и фон, однако место, которое он занимает, остается за ним, значения:

* inherit – элемент отображается, так как указанно в элементе родителе
* visible – отображает элемент (по умолчанию)
* hidden – скрывает элемент. В отличии от display: none оставляет за элементом площадь которую он занимает. Проще говоря, если display: none "удаляет" элемент, то hidden делает его "прозрачным"
* collapse – скрывает колонки и строки таблицы идентично display: none. Если свойство применяется к другим элементам, не имеющим отношения к таблицам, то результат будет таким же, как hidden

p { visibility: hidden; }

**clip**, определяет видимую область элемента, в которой будет показано его содержимое. Все, что не помещается в эту область, "обрезается" и становится невидимым. На данный момент единственно доступная форма области – прямоугольник. clip работает только для абсолютно позиционированных элементов и показывает лишь ту его часть, что входит в прямоугольную область заданную четырьмя значениями от левого и верхнего краёв данного элемента. Значения:

* rect(сверху справа снизу слева) – обрезает элемент сверху, справа, снизу и слева в соответствии с заданными, в скобках, четырьмя значениями, которые могут выражаться в процентах от ширины/длины элемента, пикселях, а также любых других единицах измерения CSS
* auto – оставляет элемент или одну из сторон элемента без изменений (по умолчанию)

p {

position: absolute;

clip: rect(15px auto auto 15px);

}

Свойства display, visibility и clip в сочетании со скриптами позволяют достичь весьма интересных результатов.

**Задание №10. Свойства выравнивания**

Примените к элементам HTML-страницы свойства выравнивания.

**float**, определяет, по какой стороне будет выравниваться – всплывать элемент, при этом остальные элементы будут обтекать его с других сторон. Выравнивание элемента происходит по краям родительского элемента или же по краям окна браузера, значения:

* left – выравнивает элемент по левому краю, остальные элементы обтекают его справа
* right – выравнивает элемент по правому краю, остальные элементы обтекают его слева
* none – выравнивание элемента не задается (по умолчанию)

img { float: right; }

**clear**, запрещает обтекание элемента с левой и/или правой стороны. Если для элемента установлено обтекание с помощью свойства float то clear отменяет обтекание данного элемента с указанных сторон, значения:

* both – запрещает обтекание элемента с левой и правой стороны
* left – запрещает обтекание элемента с левой стороны
* right – запрещает обтекание элемента с правой стороны
* none – запрет на обтекание элемента не задается (по умолчанию)

img { clear: both; }

**Задание №11. Свойства позиционирования**

Примените к элементам HTML-страницы свойства позиционирования.

Позиционированием называют определение конкретного месторасположения на странице элемента. Позиционирование бывает абсолютным, относительным, фиксированным и статическим.

**position**, позволяет позиционированть элемент, значения:

* absolute – абсолютное позиционирование элемента
* relative – относительное позиционирование элемента
* fixed – фиксированное позиционирование элемента
* static – статическое позиционирование элемента (Элемент отображаются как обычно)
* inherit – наследует значение элемента родителя

Абсолютно позиционированный элемент (**position: absolute**) выводится из общего потока и несмотря на другие элементы и их взаимное расположение, занимает указанное место на странице от края/краёв окна браузера. При таком способе позиционирования один элемент может накладываться поверх другого.

Для того чтобы позиционировать элемент от края/краёв окна браузера, используются следующие свойства:

* **bottom** – расстояние от нижнего края окна браузера
* **left** – расстояние от левого края окна браузера
* **right** – расстояние от правого края окна браузера
* **top** – расстояние от верхнего края окна браузера

Значения bottom, left, right, top:

* px – в пикселях, а также любых других единицах измерения CSS
* % – в процентах
* auto – размер полей и отступов автоматически рассчитывается браузером (по умолчанию)

div {

position: absolute;

left: 200px;

top: 100px;

}

Относительное позиционирование (**position: relative**) определяет место элемента относительно краёв элемента родителя и не выводится из общего потока. Так же как и в случае с абсолютным позиционированием расстояния от края/краёв родительского элемента задаётся с помощью свойств: bottom, left, right, top.

div {

position: relative;

left: 150px;

}

Фиксированное позиционирование (**position: fixed**) похоже на абсолютное, элемент выводится из общего потока, его координаты рассчитываются от краёв окна браузера, но при прокрутке страницы элемент не меняет своего положения. Так же как и в случае с абсолютным позиционированием расстояния от края/краёв родительского элемента задаётся с помощью свойств: bottom, left, right, top.

div {

position: fixed;

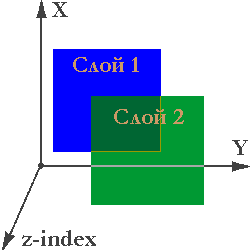
left: 250px;

top: 300px;

}

Позиционированные элементы могут накладываться один поверх другого, имитируя тем самым некую трёхмерность страницы, где каждый последующий наложенный друг на друга элемент является слоем. **z-index** позволяет управлять позициями этих слоёв в глубину экрана (по оси Z.), другими словами позволяет указывать браузеру какие элементы следует показывать на переднем плане, а какие на заднем, значения:

* auto – элементы накладываются друг на друга в том порядке, каком они указаны в html-коде (по умолчанию)
* целое число – чем выше данное значение, тем более высокую позицию занимает элемент по отношению к тем элементам, значение которых ниже



.layer1 { z-index: 1; }

.layer2 { z-index: 2; }

**Задание №12. Свойства списка**

Примените к элементам списка различные свойства.

**list-style-type**, определяет вид маркера, значения:

* disc – диск (по умолчанию для <ul>)
* circle – полый круг
* square – квадрат
* decimal – арабские цифры (по умолчанию для <ol>)
* lower-roman – строчные римские цифры
* lower-alpha – строчные буквы
* upper-roman – заглавные римские цифры
* upper-alpha – заглавные буквы
* none – маркер отсутствует

ul { list-style-type: square; }

**list-style-image**, определяет в качестве маркера списка некое графическое изображение с указанием пути к нему, значения:

* none – отменяет графическое изображение маркера
* url – путь к файлу с рисунком маркера

ul { list-style-image: url(marker.gif); }

**list-style-position**, указывает браузеру на то, как следует отображать текст в списке относительно его маркеров. По умолчанию маркеры находятся в стороне от текста списка, но можно сделать так, что они будут обтекаться текстом, значения:

* outside – маркер находится в стороне от списка (по умолчанию)
* inside – маркер обтекается текстом

ul { list-style-position: outside; }

**list-style**, используется, когда стилю списка необходимо одновременно присвоить несколько значений. Может иметь от одного до трёх значений из свойств, применяемых к стилю списка, в любой последовательности через пробел.

ul { list-style: url(graphics/marker.gif) inside; }