**6** **Блоковые элементы в CSS**

**6.1 Структура блокового элемента**

Для веб-браузера элементы страницы представляют небольшие контейнеры или блоки, которые могут иметь различное содержимое - текст, изображения, списки, таблицы и другие элементы. Схема блочной модели представлена на рисунке 6.1.

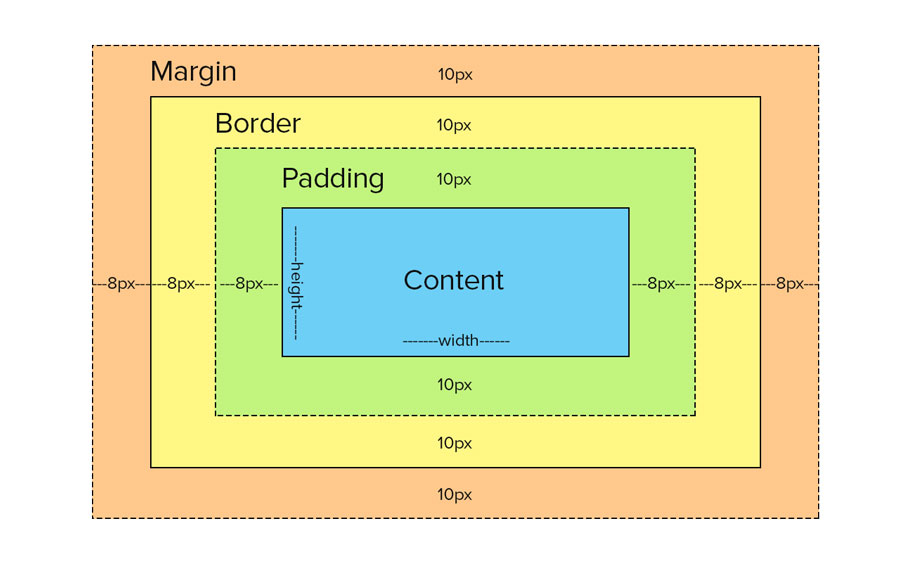


Рисунок 6.1 - Блочная модель в CSS3

Как видно на рисунке 6.1, блоковый элемент имеет следующие свойства:

- margin - внешний отступ, определяет расстояние от границы текущего элемента до других соседних элементов или до границ внешнего контейнера,

- padding - внутренний отступ, определяет расстояние от границы элемента до внутреннего содержимого,

- border - рамка,

- content - внутреннее содержимое, которое также реализует ту же блочную модель и также может состоять из других элементов.

**6.2 Внешние отступы**

Существуют специальные свойства CSS для задания отступов для каждой стороны:

* margin-top: отступ сверху,
* margin-bottom: отступ снизу,
* margin-left: отступ слева,
* margin-right: отступ справа.

Далее представлен пример работы с блоковыми элементами, в которых заданы внешние отступы margin для разных сторон.

**<head><style>**

**div {margin-top: 30px; margin-left: 25px; border: 2px; border-color: LightSkyBlue;**

**margin-right: 20px; margin-bottom: 15px; </style> </head>**

**<body><div> <p>** **Зима - самое холодное время года. В это время природа замирает до пробуждения весной. Повсюду лежат сугробы снега, реки озера покрываются льдом.</p></div>**

**<div> <p>** **Весна - самое прекрасное время года. Природа просыпается от зимней спячки, символизируя начало новой жизни. </p> </div>**

**</body>**

Реализация данного кода представлена на рисунке 6.2.

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.2 – Использование margin в CSS3

В данном примере вместо четырех свойств задать одно:

**Div {margin: 30px 20px 15px 25px; border: 3px solid LightSkyBlue;}**

Свойство задается в формате:

margin: отступ\_сверху отступ\_справа отступ\_снизу отступ\_слева;

Если значения для всех четырех отступов совпадает, то можно указать только одно значение:

**div {margin: 25px;}**

Для установки отступов можно использовать точные значения в пикселях (px) или em, либо процентные отношения, либо значение auto.

Например:

**margin-left: 2em;**

Значение 2 em определяет расстояние, которое в два раза больше размера шрифта элемента. При использовании процентов веб-браузеры вычисляют размер отступов на основе ширины элемента-контейнера, в который заключен стилизуемый элемент.

**6.3 Внутренние отступы**

Свойство padding задает внутренние отступы от границы элемента до его внутреннего содержимого. Как и для свойства margin, в CSS имеются четыре свойства, которые устанавливают отступы для каждой из сторон:

* padding-top: отступ сверху;
* padding-bottom: отступ снизу;
* padding-left: отступ слева;
* padding-right: отступ справа.

Далее представлен пример использования свойства padding для задания внутреннего отступа для блокового элемента.

**<head> <style>**

**div.pad {margin: 25px; padding-top:30px;**

**padding-right: 25px; padding-bottom: 35px;**

**padding-left: 25px; border: 2px solid Turquoise;}**

**div.in {height: 50px;}**

**</style> </head><body>**

**<div class="pad">**

**<div class="in"> Лето-самое жаркое из четырех времен года. В это время года дни длиннее ночей. в этот период времени дни становятся длиннее, а ночи короче. Растения набирают силу и радуют взгляд разнообразным буйством красок. <p> </div>**

**<div class="in"><p> Осень - самое грустное время года. Осень символизирует жизнь и её цикличность, напоминая нам о быстротечности времени и о важности наслаждаться каждым моментом. .</p> </div> </div> </div></body>**

Реализация данного кода представлена на рисунке 6.3.

Изображение выглядит как текст, электроника, снимок экрана, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.3 - Использование свойства padding в CSS3

Для установки значения отступов, как и в margin, могут применяться либо конкретные значения в пикселях, так и процентные значения.

Для записи отступов также можно использовать сокращенную запись:

padding: отступ\_сверху отступ\_справа отступ\_снизу отступ\_слева;

Например:

**div.pad {margin: 25px; padding: 30px 25px 35px 28px; border: 2px solid #0000FF;}**

Если все четыре значения совпадают, то можно писать указать только одно значение для всех отступов:

**div.in { margin: 25px; padding: 30px; border: 2px solid #0000FF;}**

**6.4 Границы**

Граница отделяется элемент от внешнего по отношению к нему содержимого. При этом граница является частью элемента. Для настройки границы могут использоваться сразу несколько свойств: ширина, цвет и начертание.

* border-width: устанавливает ширину границы,
* border-style: задает стиль линии границы,
* border-color: устанавливает цвет границы.

Свойство border-width может принимать значения в единицах измерения, таких как em, px или cm, например

* **border-width: 2px;**

Также может принимать одно из константных значений: thin (тонкая граница - 1px), medium (средняя по ширине - 3px), thick (толстая - 5px), например

* **border-width: medium;**

Свойство border-color в качестве значения принимает цвет CSS:

* **border-color: red;**

Свойство border-style оформляет тип линии границы и может принимать одно из следующих значений:

* none: граница отсутствует;
* solid: граница в виде обычной линии;
* dashed: штриховая линия;
* dotted: линия в виде последовательности точек;
* double: граница в виде двух параллельных линий;
* groove: граница имеет трехмерный эффект;
* inset: граница как бы вдавливается во внутрь;
* outset: аналогично inset, только граница как бы выступает наружу;
* ridge: граница также реализует трехмерный эффект.

Далее представлен пример задания свойств границ для блокового элемента.

**<head> <style> .kity {width: 100px; height: 100px; border-color: coral;**

**border-width: 10px; margin-top: 10px; background-image: url(cat.png);**

**background-size: cover;} </style>**

**</head> <body style="display: flex; justify-content: space-around">**

**<div> <div class="kity" style="border-style: none">none</div>**

**<div class="kity" style="border-style: solid">solid</div>**

**<div class="kity" style="border-style: dashed">dashed</div>**

**</div><div>**

**<div class="kity" style="border-style: dotted">dotted</div>**

**<div class="kity" style="border-style: double">double</div>**

**<div class="kity" style="border-style: groove">groove</div>**

**</div> <div>**

**<div class="kity" style="border-style: inset">inset</div>**

**<div class="kity" style="border-style: outset">outset</div>**

**<div class="kity" style="border-style: ridge">ridge</div>**

**</div> </body>**

Реализация данного кода представлена на рисунке 6.4.

Изображение выглядит как текст, снимок экрана, программное обеспечение, собака

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.4 - Border-style в CSS 3

При необходимости можно определить цвет, стиль и ширину границы для каждой из сторон используя следующие свойства:

**/\* для верхней границы \*/**

**border-top-width**

**border-top-style**

**border-top-color**

**/\* для нижней границы \*/**

**border-bottom-width**

**border-bottom-style**

**border-bottom-color**

**/\* для левой границы \*/**

**border-left-width**

**border-left-style**

**border-left-color**

**/\* для правой границы \*/**

**border-right-width**

**border-right-style**

**border-right-color**

Вместо установки по отдельности цвета, стиля и ширины границы можно использовать одно свойство border.

**border: ширина стиль цвет;**

Например:

**border: 2px solid red;**

Для установки границы для отдельных сторон можно использовать одно из свойств:

* border-top;
* border-bottom;
* border-left;
* border-right.

Их использование аналогично:

**border-top: 2px solid red;**

При работе с границами в HTML5 есть также возможность задать ее радиус с помощью свойства border-radius. Другими словами, свойство border-radius позволяет округлить границу. Это свойство принимает значение радиуса в пикселях или единицах em. Далее представлен пример использования свойства border-radius.

**<head> <style>**

**div {width: 100px; height:100px; border: 3px solid #8A2BE2; border-radius: 30px;}**

**</style> </head>**

**<body> <div>Изменение радиуса границы в блоковых элементах</div></body>**

Реализация данного кода представлена на рисунке 6.5.

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.5 – Использование свойства border-radius для всех углов

В данном примере каждый угол будет скругляться по радиусу в 30 пикселей. Так как у элемента может быть максимально четыре угла, то можно указать четыре значения для гнраниц **border-radius: 15px 30px 5px 40px;** (рисунок 6.6).

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.6 - Использование свойства border-radius для двух углов

Вместо общей установки радиусов для всех углов, можно их устанавливать по отдельности. Так, предыдущее значение border-radius можно переписать следующим образом:

**border-top-left-radius: 15px; /\* радиус для верхнего левого угла \*/**

**border-top-right-radius: 30px; /\* радиус для верхнего правого угла \*/**

**border-bottom-right-radius: 5px; /\* радиус для нижнего левого угла \*/**

**border-bottom-left-radius: 40px; /\* радиус для нижнего правого угла \*/**

Также border-radius поддерживает возможность создания эллиптических углов. То есть угол не просто скругляется, а использует два радиуса, образуя в итоге дугу эллипса (рисунок 6.7).

**border-radius: 30px/20px;**

В данном случае полагается, что радиус по оси X будет иметь значение 30 пикселей, а по оси Y - 20 пикселей.

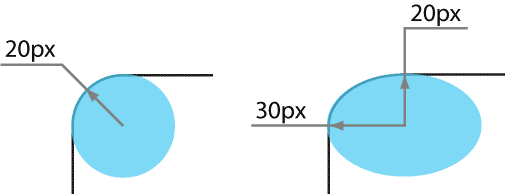


Рисунок 6.7 - Эллиптическое скругление углов в CSS 3

**6.5 Фон блокового элемента**

**6.5.1 Установка цвета и рисунка в качестве фона элемента**

Фон элемента описывается в CSS свойством background. Фактически это свойство представляет сокращение набора следующих свойств CSS:

* background-color: устанавливает цвет фона;
* background-image: в качестве фона устанавливается изображение;
* background-repeat: устанавливает режим повторения фонового изображения по всей поверхности элемента;
* background-size: устанавливает размер фонового изображения;
* background-position: указывает позицию фонового изображения;
* background-attachment: устанавливает стиль прикрепления фонового изображения к элементу;
* background-clip: определяет область, которая вырезается из изображения и используется в качестве фона;
* background-origin: устанавливает начальную позицию фонового изображения.

Далее представлен пример использования свойства background при задании фона блоковых элементов.

**<head> <style> div {width: 250px; height: 200px; margin: 10px; }**

**.colored{background-color: #00BFFF; }**

**.imaged{ background-image: url(see.jpg);**

**background-size: 130% } </style> </head>**

**<body> <div class="colored">Первый блок </div>**

**<div class="imaged">Второй блок</div> </body>**

Реализация данного кода представлена на рисунке 6.8.

Изображение выглядит как снимок экрана, текст, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.8 – Использование свойства background в CSS3

Как видно на рисунке, первый блок окрашен в оттенок синего цвета, а второй блок устанавливает в качестве фона изображение. Все содержимое блока накладывается поверх фона.

На вышеприведенном примере видно, что CSS должным образом масштабирует изображение, чтобы наиболее оптимально вписать его в пространство элемента. Однако в связи с масштабированием изображение может не полностью покрывать поверхность элемента, и поэтому для полного покрытия автоматически CSS начинает повторять изображение.

В CSS3 c помощью свойства background-repeat можно изменить механизм повторения. Оно может принимать следующие значения:

* repeat-x: повторение по горизонтали,
* repeat-y: повторение по вертикали,
* repeat: повторение по обеим сторонам (действие по умолчанию),
* space: изображение повторяется для заполнения всей поверхности элемента, но без создания фрагментов,
* round: изображение масштабируется для полного заполнения пространства,
* no-repeat: изображение не повторяется.

Далее представлен пример работы с фоном блокового элемента.

**<head> <style> div {width: 200px; height: 150px;**

**margin: 10px; border: black solid 1px;**

**background-size: 140px 110px; background-image: url(dubi.png); }**

**.imaged1{background-repeat: repeat-x;}**

**.imaged2{background-repeat: repeat-y;}**

**.imaged3{background-repeat: repeat;}**

**.imaged4{background-repeat: space;}**

**.imaged5{background-repeat: round;} </style> </head>**

**<body style="display: flex;"><div class="imaged1"></div>**

**<div class="imaged2"></div> <div class="imaged3"></div>**

**<div class="imaged4"></div><div class="imaged5"></div> </body>**

Реализация данного кода представлена на рисунке 6.9.

Изображение выглядит как снимок экрана, вода, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.9 - Повторения изображения в блоковых элементах

**6.5.2 Настройка размера фонового изображения**

Помимо этого, можно изменить размер фонового изображения. Для этого следует использовать свойство background-size. Для установки размера можно использовать либо единицы измерения (пиксели, проценты), либо одно из предустановленных значений:

* contain: масштабирует изображение по наибольшей стороне, сохраняя аспектное отношение;
* cover: масштабирует изображение по наименьшей стороне, сохраняя аспектное отношение;
* auto: значение по умолчанию, изображение отображается в полный размер.

Если нужно масштабировать изображение, чтобы оно оптимальнее было вписано в фон, то для обеих настроек можно установить значение 100%.

**background-size: 100% 100%;**

Если задаются точные размеры, то сначала указывается ширина, а потом высота изображения.

**background-size: 200px 150px;**

Можно задать точное значение для одного измерения - ширины или высоты, а для другого задать автоматические размеры, чтобы браузер сам выводил точные значения:

**background-size: 200px auto;**

Далее представлен пример настройки размеров изображений блоков.

**<head> <style> div {width: 200px; height: 150px; margin: 10px; border: black solid 1px;**

**background-image: url(cat3.png); }**

**.imaged1{ background-size: cover; }**

**.imaged2{ background-size: 140px 110px; } </style> </head>**

**<body> <div class="imaged1"></div>**

**<div class="imaged2">**

**</div> </body>**

Во втором случае изображение будет масштабироваться до размеров 140х110. Поскольку еще остается место на элементе, то по умолчанию изображение будет

повторяться для заполнения всей поверхности. Реализация данного кода представлена на рисунке 6.10.

Изображение выглядит как снимок экрана, Мультимедийное программное обеспечение, мультимедиа, мультфильм

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.10 - Свойство background-size в CSS 3

**6.5.3 Использование градиентов при оформлении фона элемента**

**6.5.3.1 Применение линейного градиента**

Градиенты представляют собой плавный переход от одного цвета к другому. CSS3 имеет ряд встроенных градиентов, которые можно использовать для создания фона элемента. Градиенты в CSS не представляют собой специального свойства, они просто создают значение, присваиваемое свойству background-image.

Линейный градиент проходит по прямой линии от одного конца элемента к другому, создавая плавный переход от одного цвета к другому. Для создания градиента необходимо указать его начало и несколько цветов.

**background-image: linear-gradient (left, black, white);**

В этом случае началом градиента является левый край элемента, заданный левым значением (left). Градиент: черно-белый. Это означает, что происходит плавный переход от черного к белому от левого края элемента к правому.

У использования градиентов есть один недостаток: разнообразие браузеров вынуждает использовать префикс провайдера:

**-webkit- /\* Для Yandex, Safari, Microsoft Edge, Opera выше 15 версии \*/**

**-moz- /\* Для Mozilla Firefox \*/**

**-o- /\* Для Opera старше 15 версии (Opera 12) \*/**

Для установки начала градиента используются значения: left (слева направо), right (справа налево), top (сверху вниз) или bottom (снизу вверх).

Далее представлен пример использования градиентов, с использованием свойства left.

**<head> <style>**

**div {width: 200px; height: 100px;**

**background-image: linear-gradient(left, #FFFF00, #008000);**

**background-image: -o-linear-gradient(left, #FFFF00, #008000);**

**background-image: -moz-linear-gradient(left, #FFFF00, #008000);**

**background-image: -webkit-linear-gradient(left, #FFFF00, #008000);}**

**p {margin: 0; padding-top: 30px; text-align: center; color: white;}**

**</style> </head>**

**<body> <div><p>Линейный желто-зеленый градиент</p>**

**</div> </body>**

Реализация данного кода представлена на рисунке 6.11.

Изображение выглядит как текст, снимок экрана, электроника, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.11 – Использование горизонтальных линейных градиентов

Если использовать свойство bottom, то градиент меняет направление на вертикальный. Например, далее представлен пример, в котором использовалось свойство bottom.

**background-image: linear-gradient (bottom, #FFFF00, #008000);**

Изображение выглядит как текст, снимок экрана, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.12 – Задание вертикального градиента в CSS3

Также можно задать диагональное направление с помощью двух значений.

**background-image: linear-gradient(top left, #FFFF00, #008000);**

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.13 - Настройка диагонального линейного градиента в CSS3

Помимо конкретных значений вроде top или left, также можно указать угол от 0 до 360, который будет определять направление градиента:

**background-image: linear-gradient(30deg, #FFFF00, blue);**

После значения углов указывается слово deg. Например, 0deg означает, что градиент начинается слева и движется вправо, а 45deg означает, что он начинается слева внизу и движется под углом 45° вверх вправо. Определив начало градиента, можно указать цвета или опорные точки, которые нужно применить. Количество цветов может быть больше:

**background-image: linear-gradient (top, #FFFF00, #ccc, blue);**

Все нанесенные цвета распределяются равномерно. Можно указать определенные положения фона для цветовых точек. Для этого после цвета добавляется второе значение, определяющее положение точки.

**background-image: linear-gradient (left, #ccc, #FFFF00 20%, red 80%, #ccc);**

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.14 – Задание углов в качестве параметров градиента

**6.5.3.2 Применение радиального градиента**

Радиальные градиенты в отличие от линейных распространяются от центра наружу по круговой схеме. Для создания радиального градиента достаточно указать цвет, который будет в центре градиента, и цвет, который должен быть снаружи. Эти цвета передаются в функцию radial-gradient().

**<head> <style>**

**div {width: 200px; height: 200px; border-radius: 100px;**

**background-color: #eee; background-image: radial-gradient(red, blue);**

**background-image: -moz-radial-gradient(red, blue);**

**background-image: -webkit-radial-gradient(red, blue);}**

**p {margin: 0; padding-top: 60px; text-align: center; color: white;} </style> </head> <body> <div><p>Радиальный градиент</p></div></body>**

Реализация данного кода представлена на рисунке 6.15.

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.15 - Радиальный градиент в CSS3

Как и в случае с линейным градиентом здесь также надо использовать префиксы для поддержки браузерами. Радиальный градиент может иметь две формы: круговую и эллиптическую. Эллиптическая форма представляет распространение градиента в виде эллипса и задается с помощью ключевого слова ellipse.

**background-image: radial-gradient(ellipse, red, blue);**

Поскольку это значение для градиента по умолчанию, то оно может опускаться при использовании. Круговая форма представляет распространение градиента в виде кругов от центра во вне. Для этого используется ключевое слово circle:

**background-image: radial-gradient(circle, red, blue);**

Как правило, центр радиального градиента расположен в центре элемента, но это поведение можно переопределить, указав значение для параметра background-position (рисунок 6.16).

**background-image: radial-gradient(25% 30%, circle, red, blue);**

Числа 25% 30% в данном случае означают, что центр градиента будет находиться на расстоянии в 25% от левой границы и в 30% от верхней границы элемента.

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.16 - Позиционирование радиального градиента в CSS3

С помощью специальных дополнительных значений можно задать размер градиента:

* closest-side: градиент распространяется из центра только до ближайшей к центру стороне элемента. То есть градиент остается внутри элемента;
* closest-corner: ширина градиента вычисляется по расстоянию из его центра до ближайшего угла элемента, поэтому градиент может выйти за пределы элемента;
* farthest-side: градиент распространяется из центра до самой дальней стороны элемента;
* farthest-corner: ширина градиента вычисляется по расстоянию из его центра до самого дальнего угла элемента.

Далее представлен пример задания размера градиента.

**background-image: radial-gradient (25% 30%, circle farthest-corner, red, blue);**

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.17 – Задание размеров градиенту

**6.6 Позиционирование блоковых элементов**

CSS предоставляет возможности размещения элементов, что означает, что есть возможность разместить элемент в определенном месте на странице.

Основным свойством, управляющим позиционированием в CSS, является свойство position. Это свойство может принимать одно из следующих значений:

* absolute: элемент позиционируется относительно границ содержащего его элемента, в качестве точки отсчета используется левое верхняя точка окна браузера,
* relative: элемент позиционируется относительно предыдущего элемента. В качестве точки отсчета используется левая верхняя точка окна браузера,

- fixed: элемент позиционируется фиксировано в окне браузера, это позволяет создавать фиксированные элементы, не меняющие положение при прокрутке,

* static: размещение элемента по умолчанию, значение по умолчанию.

При определении позиции элемента используются следующие параметры:

* left – расстояние от начала координат (от левой верхней точки окна браузера) по горизонтали, аналогично параметру х,
* top – расстояние от начала координат (от левой верхней точки окна браузера) по вертикали, аналогично параметру х,

- z-индекс – определяет порядок наложения элементов друг на друга.

Далее представлены примеры работы с различными видами позиционирования.

**6.6.1 Абсолютное позиционирование**

Область отображения браузера имеет верхнее, нижнее, правое и левое поля. У каждого из этих четырех краев есть соответствующее свойство CSS: left, right, top и bottom. Значения этих свойств задаются в пикселях, em или процентах. Нет необходимости задавать значения для всех четырех сторон. Как правило задаются только два значения - отступ от верхнего края top и отступ от левого края left.

Изображение выглядит как текст, доска, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.18 – Отправная точка для позиционирования элементов

Далее представлен пример применения абсолютного позиционирования для блокового элемента.

**<head> <style>**

**.mystyle {position: absolute;**

**left: 100px; top: 50px; width: 430px; border=3;**

**background-color: MediumAquamarine;} </style> </head> <body>**

**<div class="mystyle"> <p>Казань – город на юго-западе России, расположенный на берегах Волги и Казанки. В столице полуавтономной Республики Татарстан находится древний кремль – крепость, известная своими музеями и святыми местами. Башня Сююмбике, синие и золотые купола Благовещенского собора и яркая джума-мечеть Кул-Шариф – одни из самых интересных достопримечательностей кремля. </p </div> </body>**

Реализация данного кода представлена на рисунке 6.19.

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.19 – Абсолютное позиционирование блоковых элементов

Здесь абсолютно позиционированный элемент div будет находится на 100 пикселей слева от края окна просмотра и на 50 пикселей снизу.

Помимо форматирования блоковых элементов, в CSS имеется возможность также позиционировать текст.

**<head> <style> .mystyle { position: absolute;**

**left: 100px; top: 50px;**

**width: 430px;**

**height: 100px;**

**color: SkyBlue;}**

**</style> </head>**

**<body>**

**<p class="mystyle"> Город-крепость Оренбург был построен в 1743 году, расположен на слиянии двух рек - Урала и Сакмары, является главным связующим звеном между Европой и Азией. До 1743 года Оренбург дважды закладывали в других местах. Открытие газового месторождения, которое пришлось на 70-е годы XX века, явилось вторым рождением города Оренбурга.</p </div> </body>**

Реализация данного кода представлена на рисунке 6.20.

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.20 – Пример абсолютного позиционирования текста

**6.6.2** **Относительное позиционирование**

Относительное позиционирование также задается с помощью свойства position со значением relative. Далее приведен пример относительного позиционирования.

**<head> <style>**

**.outer {position: relative; left: 80px; top: 40px; width: 300px;**

**height: 100px; border: 1px solid; background-color: IndianRed ; color: white;}**

**.inner{ position: absolute; left: 80px; top: 40px; width: 300px;**

**height: 100px; border: 1px solid; background-color: SteelBlue; color: white;} </style> </head>**

**<body> <div class="outer">**

**<p> Москва - столица России и крупнейший город страны. Это огромный мегаполис, который является историческим, политическим и духовным сердцем Российской Федерации. Москва - крупнейшая столица Европы, наполненная достопримечательностями, памятниками истории и культуры, а также музеями мирового уровня**.**</p>**

**<div class="inner">**

**<p> Санкт-Петербург -** [**второй по численности населения**](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D1%80%D0%BE%D0%B4%D0%B0-%D0%BC%D0%B8%D0%BB%D0%BB%D0%B8%D0%BE%D0%BD%D0%B5%D1%80%D1%8B_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B8)**город**[**России**](https://ru.wikipedia.org/wiki/%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D1%8F)**. Русский портовый город на побережье Балтийского моря. Был основан в 1703 году Петром I, которому воздвигнут знаменитый памятник "Медный всадник". Город по праву считается культурным центром страны.** **</p></div></div> </body>**

Реализация данного кода представлена на рисунке 6.21.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.21 – Пример относительного позиционирование блоков

Аналогично, как и с абсолютным позиционированием, свойство относительного позиционирования может применяться и к текстовым элементам.

Далее представлен пример объемных заголовков с помощью абсолютного позиционирования текстовых элементов. В данном случае создается HTML-документ создаются три наложенных друг на друга надписей. Одна надпись предназначена для эффекта тени (задний план), вторая располагается над первой (передний план), еще одна используется для создания эффекта подсветки.

**<head> <style> p {font-family:sans-serif;**

**font-size:72;**

**font-weight:800;**

**color:DarkTurquoise}**

**p.highlight {color:silver;}**

**p.shadow {color:darkred;} </style></head>**

**<body bgcolor= LightCyan>**

**<div style=“position:absolute; top:5; left:5”>**

**<p class=shadow>Объемный заголовок</p></div>**

**<div style=“position:absolute; top:0; left:0”>**

**<p class=highlight> Объемный заголовок </p>**

**<div style=“position:absolute; top:2; left:2”>**

**<p>Объемный заголовок </p></div></body>**

Реализация данного кода представлена на рисунке 6.22.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.22 – Абсолютное позиционирование текста в CSS3

Таким образом, язык CSS позволяет создавать различные объемные объекты без использования графики.

**6.6.3 Свойство z-index**

По умолчанию, когда два элемента имеют одинаковую границу, последний элемент, определенный в разметке HTML, отображается над другим. Однако свойство z-index позволяет изменять порядок элементов при их перекрытии. Элементы с большим значением z-индекса будут отображаться над элементами с меньшим значением z-индекса. Можно поменять значения этого параметра, тем самым позволяя элементам, объявленным позднее перекрывать элементы, объявленные ранее.

Далее представлен пример работы с параметром z-index. В данном примере представлено использование перекрытие как блоковых, так и текстовых элементов.

**<head> <style>**

**.content {position: relative;**

**top: 10px; left: 25px;**

**width: 260px; height: 190px;**

**background-color: #eee;**

**border: 2px solid;}**

**.LightPinkBlock {position: absolute; top: 10px; left:40px; width: 70px;**

**height: 70px; background-color: LightPink; }**

**.MediumVioletRedBlock { position: absolute; top: 70px; left: 70px; width: 70px; height: 70px; background-color: MediumVioletRed;}**

**.Block1{ position: absolute; top: 10px; left:40px; color: LightPink; }**

**.Block2{ position: absolute; top: 15px; left: 20px; color: MediumVioletRed; } </style></head>**

**<body style="display: flex;"> <div class="content">**

**<div class="LightPinkBlock">**

**<h1> Lamborghini Aventador</h2></div>**

**<div class=" MediumVioletRedBlock">**

**<h1>Mercedes-Benz Maybach</h2></div> </div>**

**<div class="content" style="margin-left: 10px;">**

**<div class="Block1"> Lamborghini Aventador - это истинное произведение искусства в мире автомобилестроения. Он оснащен мощным 6,5-литровым двигателем V12, который развивает 700 лошадиных сил. Благодаря этому двигателю, Aventador способен разгоняться до 100 километров в час всего за 2,9 секунды.</div>**

**<div class="Block2">Mercedes-Benz Maybach - роскошный седан, который олицетворяет собой высшую эстетику и элегантность, возведённые в ранг абсолюта. Полноприводная версия модели оснащена 6-ти литровым 630-сильным мотором с двумя турбокомпрессорами. </div> </div></body>**

Реализация данного кода представлена на рисунке 6.23.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.23 – Применение свойства Z-index в CSS3 для первого блока

Теперь к стилю блока добавляется новое правило:

**.LightPinkBlock { z-index: 100; position: absolute; top: 10px; left:40px; width: 70px;**

**height: 70px; background-color: LightPink; }**

Второй блок имеет неопределенный z-индекс, то для LightPinkBlock можно установить для свойства z-index любое значение, большее нуля. В данном случае 100. Теперь первый блок будет перекрывать второй, а не наоборот, как было вначале (рисунок 6.24).

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.24 – Применение свойства Z-index в CSS3 для второго блока

**6.6.4 Фиксированное позиционирование**

Фиксированное позиционирование - это распространенный способ сохранить определенные элементы в области просмотра браузера путем создания фиксированной панели навигации, не меняющей своего положения при прокрутке.

Для фиксированного позиционирования элементов необходимо установить для свойства позиции значение “fixed”. Затем можно использовать стандартные свойства left, right, чтобы определить конкретное положение фиксированного элемента. Чтобы растянуть фиксированный блок от левой до правой границы страницы, устанавливаются три свойства: **top: 0; left: 0; right: 0;**

Далее приведен пример создания фиксированной панели навигации.

**<head> <style>**

**.toolbar{ position: fixed; top: 0;left: 0; right: 0;**

**background-color: #4169E1;**

**border-bottom: 1px solid #ADD8E6;}**

**.toolbar a{ color: white; display: inline-block; padding: 10px; text-decoration: none;**

**font-family: Verdana;}.**

**content{ margin-top: 50px; padding: 10px;} </style> </head>**

**<body> <div class="toolbar">**

**<a href="#">С++ </a> <a href="#">Java</a>**

**<a href="#"> PHP</a> <a href="#"> Kotlin</a></div>**

**<div class="content"> Сайт обучения программированию для начинающих, который предоставляет различные образовательные ресурсы. Предоставляет множество языков программирования для изучения и является подходящим местом как для компаний, так и для разработчиков. </div> </body>**

Реализация данного кода представлена на рисунке 6.25.

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.25 – Работа с фиксированными блоками в CSS

**6.6.5 Статическое позиционирование**

В стандартном потоке статически позиционированные элементы ведут себя аналогично относительно позиционированным: они отображаются непосредственно сразу после предыдущего элемента в потоке. Единственное их отличие от относительно позиционируемых заключается в том, что для них нельзя установить значения свойств top и left, и тем самым сместить их, например, со строки абзаца вверх или вниз.

Для того, чтобы в CSS реализовать “резиновую” верстку, были придуманы плавающие блоки. Их нельзя позиционировать с точностью до пикселя, как, например, абсолютно позиционированные. Они могут свободно перемещаться и прижиматься к краю своего контейнера. Данные блоки имеют ряд свойств, таких как float, overflow, clear, display, visibility. Рассмотрим каждый из них.

**6.6.5.1 Свойство float**

Плавающие блоки в CSS определяются свойством float. Данное свойство определяет, будет ли блок плавающим, и в какую сторону он будет перемещаться. Свойство может принимать следующие значения:

- left - структурный блок перемещается влево. Остальное содержимое документа будет выводиться вдоль правой стороны блока, начиная с самого верха.

* right - структурный блок перемещается вправо. Остальное содержимое документа выводится вдоль левой стороны блока, начиная с самого верха.
* none - блок не перемещается, т.е. позиционируется согласно алгоритму, заданному свойству position.

Далее представлен пример простого плавающего блока. Создан блок и задано свойство float со значением “right”.

**<head><style>**

**.сomment { background: PaleTurquoise;**

**border: 1 px solid;**

**padding: 5 px;**

**width: 150 px; float: right;}**

**</style>**

**</head>**

**<body>**

**<div class= “comment”> Среди самых дорогих картин Эрмитажа "Мадонна с Младенцем" Рафаэля Санти претендует на верхнюю ступеньку пьедестала. Ранняя работа Мастера куплена Александром II у графа Конестабиле в подарок любимой жене за 310 тыс. франков. Собственность императрицы Марии Александровны сразу попала в выставочный зал Эрмитажа. </div>**

**<p>** **Государственный Эрмитаж является одним из самых крупных музеев изобразительного искусства в мире. Музей состоит из шести зданий, связанных между собой, - Зимний дворец, Запасной дом Зимнего дворца, Малый Эрмитаж, Большой (Старый) Эрмитаж, Новый Эрмитаж и Эрмитажный театр. Эрмитаж возник как частная коллекция произведений искусства, приобретавшихся в частном порядке российской императрицей Екатериной II с начала её царствования. Первоначально это собрание размещалось в главной императорской резиденции в специальном дворцовом флигеле - Эрмитаже (в современном комплексе - Малый Эрмитаж) - откуда и закрепилось общее название будущего музея </p> </body>**

Реализация данной HTML-страницы в браузере представлена на рисунке 6.26.

Изображение выглядит как текст, электроника, снимок экрана, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.26 – Использование плавающих блоков,float: right

Как видно из рисунка 5.26, текст может обтекать не только рисунки, но и другие текстовые блоки. Если изменить значение свойства float с “right” на “left”, то блок переместится к левому краю контейнера (рисунок 6.27).

Изображение выглядит как текст, электроника, снимок экрана, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.27 – Использование плавающих блоков,“float: left”

Если необходимо вставить несколько одинаковых плавающих блоков, то каждый следующий блок будет устанавливаться правее исходного (рисунок 6.28). Как видно из рисунка второй блок располагается правее первого, несмотря на то, что также является левосторонним, но прикрепился к нему левым боком.

Изображение выглядит как текст, электроника, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.28 – Использование двух плавающих блоков

**6.6.5.2 Прокрутка элементов. Свойство overflow**

Нередко при создании веб-страниц можно столкнуться с ситуацией, когда содержимое блока занимает гораздо больше места, чем сам определено шириной и высотой блока. В этой ситуации по умолчанию браузер все равно отображает содержимое, даже если оно выходит за границы блока.

Однако свойство overflow позволяет настроить поведение блока в подобной ситуации и добавить возможность прокрутки. Это свойство может принимать следующие значения:

* auto: если контент выходит за границы блока, то создается прокрутка. В остальных случаях полосы прокрутки не отображаются
* hidden: отображается только видимая часть контента. Контент, который выходит за границы блока, не отображается, а полосы прокрутки не создаются
* scroll: в блоке отображаются полосы прокрутки, даже если контент весь помещается в границах блока, и таких полос прокрутки не требуется
* visible: значение по умолчанию, контент отображается, даже если он выходит за границы блока.

Далее представлен пример использования свойства overflow с различными значениями.

**<head> <style>**

**.part1 {width: 300px; height: 140px; margin: 10px; border: 2px solid;**

**overflow: auto; background-color:Thistle;}**

**.part2 { width: 300px; height: 140px; margin: 10px; border: 2px solid;**

**overflow: hidden; background-color:Thistle;}**

**.part3 {width: 300px; height: 140px; margin: 10px; border: 2px solid;**

**overflow: scroll; background-color:Thistle;}**

**.part4 {width: 300px; height: 140px; margin: 10px; border: 2px solid;**

**overflow: visible; background-color:Thistle;}**

**</style> </head>**

**<body style="display: flex">**

**<div> <div class="part1"> <p> Я не знаю, что сказать тебе при встрече. Не могу найти хотя бы пары слов. А недолгий вечер, а недолгий вечер. Скоро станет ночью темною без снов. А недолгий вечер, а недолгий вечер. Станет ночью темною без снов. </p> </div>**

**<div class="part2"> <p> И снова седая ночь. И только ей доверяю я. Знает седая ночь не все мои тайны. Но даже и ты помочь. Не можешь, и темнота твоя. Мне одному совсем, совсем ни к чему. И снова седая ночь. И только ей доверяю я. Знает седая ночь не все мои тайны. Но даже и ты помочь. Не можешь, и темнота твоя. Мне одному совсем, совсем ни к чему. </p> </div> </div>**

**<div> <div class="part3"> <p> Знаешь ты без слов, тебе давно все ясно. Только прячешь взгляд своих счастливых глаз. И опять напрасно, и опять напрасно. Собираюсь я все рассказать сейчас. И опять напрасно, и опять напрасно. Собираюсь все сказать сейчас.</p>**

**</div> <div class="part4"> <p> И снова седая ночь. И только ей доверяю я. Знает седая ночь не все мои тайны. Но даже и ты помочь. Не можешь, и темнота твоя. Мне одному совсем, совсем ни к чему. И снова седая ночь. И только ей доверяю я. Знает седая ночь не все мои тайны. Но даже и ты помочь. Не можешь, и темнота твоя. Мне одному совсем, совсем ни к чему. </p> </div> </div>**

**</body>**

Реализация данного кода представлена на рисунке 6.29.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.29 - Создание прокрутки в CSS

Свойство overflow управляет полосами прокрутки как по вертикали, так и по горизонтали. С помощью дополнительных свойств overflow-x и overflow-y можно определить прокрутку соответственно по горизонтали и по вертикали. Данные свойства принимают те же значения, что и overflow:

**overflow-x: auto;**

**overflow-y: hidden;**

**6.6.5.3 Свойство clear**

Свойство clear определяет, какие стороны перемещаемых блоков не могут соседствовать с другими перемещаемыми блоками. Свойство применяется только для элементов блокового уровня и может принимать следующие значения:

-left - блок должен размещаться ниже всех левосторонних плавающих блоков,

- right - блок должен размещаться ниже всех правосторонних плавающих блоков,

- both - блок должен размещаться ниже всех плавающих блоков,

- none - никаких ограничений на положение блока относительно перемещаемых объектов не накладывается.

Если у некоторого элемента существуют перемещаемые потомки, свойство clear на них не распространяется.

Далее представлен пример HTML-кода с двумя блоками, зададим для второго блока свойство “clear: left”.

**<head>**

**<style>**

**.сomment {background: SkyBlue; color: white;**

**border: 1 px solid; padding: 5 px;**

**width: 150 px; float: right;}**

**</style>**

**</head>**

**<body> <div class=“comment”> Одной из достопримечательностей Стамбула является Топкапы -  главный дворец Османской империи до середины XIX века. Расположен в историческом центре Стамбула, на мысе Сарайбурну, в месте впадения Босфора и Золотого Рога в Мраморное море. </div>**

**<div class=“comment” style=“clear:left”> Еще одной достопримечательностью Стамбула является голубая мечеть. Голубая мечеть, или Мечеть Султанахмет, - первая по значению мечеть Стамбула. Насчитывает шесть минаретов: четыре, как обычно, по сторонам, а два чуть менее высоких - на внешних углах. Считается выдающимся образцом исламской и мировой архитектуры. </div>**

**<p> Самым красивым городом в Турции по праву считается Стамбул. Стамбул - удивительный город: огромный, шумный и очень колоритный. Он манит своей многовековой историей, смешением цивилизаций, сплетением европейского и азиатского. Уникальность географического расположения Стамбула заключается в том, что это ближайший к Европе азиатский город и ближайший к Азии европейский город. Стамбул одарил своей красотой не только Турцию, но и весь мир, став исторической сокровищницей, привлекающей внимание всего человечества. Стамбул – жемчужина для туристов, множество исторических мест, музеев, художественных галерей и экскурсий, а также хамам – все это причины, чтобы посетить этот чудесный город. </p>**

**</body>**

Реализация данного кода представлена на рисунке 6.30.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.30 – Действие свойства “clear: lеft” на плавающий блок

Как видно из рисунка второй блок прикрепился к левому плавающему блоку снизу. Получилось два блока: один сверху, другой снизу.

Помимо этого, плавающие блоки имеют еще одну особенность. Несколько плавающих блоков будут располагаться на одной линии, если их суммарная ширина будет меньше ширины контейнера или будет равна ей. Если в качестве контейнера рассмотреть окно браузера, то таким образом можно легко реализовать верстку в несколько колонок. Ширину блоков при этом можно указать в процентах.

Далее представлен пример трех плавающих левосторонних блока так, что их суммарная ширина не превышала 100% ширины окна.

**<head><style> #block1 {background: PaleVioletRed; border: 1px solid;**

**padding: 5 px; width: 30%; float: left;}**

**#block2 {background: PaleVioletRed; border: 1px solid;**

**padding: 5 px; width: 30%; float: left;}**

**#block3 {background: PaleVioletRed; border: 1px solid;**

**padding: 5 px; width: 30%; float: left;}**

**</style> </head> <body>**

**<div id=“block1”>** **Dart - это язык программирования, используемый для создания кроссплатформенных приложений с помощью фреймворка Flutter. </div>**

**<div id=“block2”>Java - один из популярных языков программирования для Android. Это объектно-ориентированный язык, который используется для создания приложений с высокой производительностью.</div>**

**<div id=“block3”>** **Kotlin – высокоуровневый язык программирования общего применения, который является надстройкой над Java. Язык используется в качестве основного языка для разработки приложений на Android в официальной SDK – Android Studio.</div></body>**

Реализация данного HTML-кода представлена на рисунке 6.31.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.31 – Использование нескольких блоков, высота каждого зависит от его содержимого

Как видно из рисунка 6.31, высота колонок будет зависеть от содержимого блоков. Чтобы подровнять колонки по высоте, следует в CSS-правилах задать свойство height:100%. Это позволяет реализовать “резиновую” (рисунок 6.32).

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.32 – Выравнивание высоты колонок с помощью свойства height:100%

Как видно на рисунке 6.32 высота колонок выровнена.

**6.6.5.4 Свойство display**

Кроме свойства float, которое позволяет изменять позицию элемента, в CSS есть еще одно важное свойство - display. Оно позволяет управлять блоком элемента и также влиять на его позиционирование относительно соседних элементов.

Это свойство может принимать следующие значения:

* inline: элемент становится строчным, подобно словам в строке текста;
* block: элемент становится блочным, как параграф;
* inline-block: элемент располагается как строка текста;
* list-item: элемент позиционируется как элемент списка обычно с добавление, маркера виде точки или порядкового номера;
* run-in: тип блока элемента зависит от окружающих элементов;
* flex: позволяет осуществлять гибкое позиционирование элементов;
* table, inline-table: позволяет расположить элементы в виде таблицы;
* none: элемент не виден и удален из разметки html.

Итак, значение block позволяет определить блочный элемент. Такой элемент визуально отделяется от соседних элементов переносом строки, как, например, элемент параграфа p или элемент div, которые по умолчанию являются блочными и при визуализации веб-страницы визуально переносятся на новую строку. Однако элемент span в отличие от элемента div блочным не является. Далее приведены примеры работы свойства display c различными значениями.

Использование свойства display со значением block.

**<head> <style>**

**span { color: RoyalBlue;}**

**.blockSpan { display: block; border: 2px;}**

**</style> </head>**

**<body>**

**<div>Черное <span>море</span> -** [**внутреннее море**](https://ru.wikipedia.org/wiki/%D0%92%D0%BD%D1%83%D1%82%D1%80%D0%B5%D0%BD%D0%BD%D0%B5%D0%B5_%D0%BC%D0%BE%D1%80%D0%B5)**бассейна**[**Атлантического океана**](https://ru.wikipedia.org/wiki/%D0%90%D1%82%D0%BB%D0%B0%D0%BD%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BE%D0%BA%D0%B5%D0%B0%D0%BD)**. Проливом**[**Босфор**](https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D1%81%D1%84%D0%BE%D1%80)**соединяется с**[**Мраморным**](https://ru.wikipedia.org/wiki/%D0%9C%D1%80%D0%B0%D0%BC%D0%BE%D1%80%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BE%D1%80%D0%B5)**морем, далее, через пролив**[**Дарданеллы**](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D1%80%D0%B4%D0%B0%D0%BD%D0%B5%D0%BB%D0%BB%D1%8B)**с**[**Эгейским**](https://ru.wikipedia.org/wiki/%D0%AD%D0%B3%D0%B5%D0%B9%D1%81%D0%BA%D0%BE%D0%B5_%D0%BC%D0%BE%D1%80%D0%B5)**и**[**Средиземным**](https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B5%D0%B4%D0%B8%D0%B7%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BE%D1%80%D0%B5)**морями. </div>**

**<div>Средиземное<span class="blockSpan">море</span> - межматериковое море**[**Атлантического океана**](https://bigenc.ru/c/atlanticheskii-okean-4e914d)**, между Европой, Африкой и Азией.  Оно названо так благодаря своему положению среди земель, составлявших весь известный древним мир.</div> </body>**

Реализация данного кода представлена на рисунке 5.32.

Изображение выглядит как электроника, текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.32 - Использование свойства display со значением block

В данном примере определено два элемента span, но один из них является блочным, так как к нему применяется стиль display: block;. Поэтому этот элемент span переносится на новую строку.

1. Использование свойства display со значением inline. В отличие от блочных элементов строчные встраиваются в строку, так как имеют для свойства display значение inline. Элемент span как раз по умолчанию имеет стиль display: inline, поэтому и встраивается в строку, а не переносится на следующую, как параграфы или div. Далее реализована обратная процедура - сделать блочный элемент div строчным.

**<head> <style>**

**div{ display: inline; } </style> </head>**

**<body color: FireBrick; > <h3>Интернет магазин быстрой еды</h3>**

**<div>Гамбургеры</div>**

**<div>Картошка фри</div>**

**<div>Нагетсы</div>**

**<div>Напитки</div></body>**

Реализация данного кода представлена на рисунке 6.33.

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.33 - Использование свойства display со значением inline

1. Использование свойства display со значением inline-block. Еще одно значение - inline-block - представляет элемент, который обладает смесью признаков блочного и строчного элементов. По отношению к соседним внешним элементам такой элемент расценивается как строчный. То есть он не отделяется от соседних элементов переводом строки. Однако по отношению к вложенным элементам он рассматривается как блочный. И к такому элементу применяются свойства width, height, margin.

**<head> <style>**

**span{ width: 100px; height: 30px; background-color**: **LightSeaGreen; }**

**.inineBlockSpan{ display: inline-block;} </style> </head>**

**<body> <p> Хвойный лес <span> - лес</span>, состоящий из деревьев одной или нескольких хвойных пород: сосны, ели, пихты, лиственницы и другие.</p>**

**<p> Лиственный лес<span class="inineBlockSpan">-** [**лес**](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D1%81)**</span>, состоящий из**[**лиственных**](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%BF%D0%BE%D1%80%D0%BE%D0%B4%D1%8B)[**пород**](https://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B5%D0%B2%D0%B5%D1%81%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D1%80%D0%BE%D0%B4%D0%B0)[**деревьев**](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D1%8F)**и**[**кустарников**](https://ru.wikipedia.org/wiki/%D0%9A%D1%83%D1%81%D1%82%D0%B0%D1%80%D0%BD%D0%B8%D0%BA%D0%B8) **</p> </body>**

Реализация данного кода представлена на рисунке 6.34.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.34 – Использование свойства display со значением inline-block

Первый элемент span является строчным, у него значение inline, поэтому для него бессмысленно применять свойства width и height. А вот второй элемент span имеет значение inline-block, поэтому к нему также применяются и ширина, и высота, и при необходимости еще можно установить отступы.

1. Использование свойства display со значением run-in. Значение run-in определяет элемент, который зависит от соседних элементов. И здесь есть три возможных варианта:

* Элемент окружен блочными элементами, тогда фактически он имеет стиль display: block, то есть сам становится блочным.
* Элемент окружен строчными элементами, тогда фактически он имеет стиль display: inline, то есть сам становится строчным.

Во всех остальных случаях элемент считается блочным.

5) Использование свойства display со значением table. Значение table, по сути, превращает элемент в таблицу. Для этого у элемента списка устанавливается стиль display: table-cell.

**<head> <style>**

**ul { display: table; margin: 0; }**

**li { list-style-type: none; display: table-cell; padding: 10px; } </style> </head>**

**<body color: BlueViolet;> <h2>Корзина покупок:</h2>**

**<ul> <li>Товар</li> <li>Цена</li> <li>Количество</li>**

**<li>Печенье</li> <li>50 р.</li> <li>3 пачки</li>**

**<li>Чай</li> <li>100 р.</li> <li>1 пачка.</li>**

**<li>Конфеты</li> <li>700 р.</li> <li>1 кг.</li> </ul> </body>**

Реализация данного кода представлена на рисунке 6.35.

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.35 – Использование свойства display со значением table

6) Использование свойства display со значением none. Значение none позволяет скрыть элемент, которого как будто нет на веб-странице.

**<head> <style>**

**.invisible { display: none; } </style> </head>**

**<body color: Teal;> <h2> Содержание романа “Капитанская дочка”</h2>**

**<p>**[**Глава I. Сержант гвардии**](https://ilibrary.ru/text/107/p.1/index.html)**.</p>**

**<p> Глава II.** [**Вожатый**](https://ilibrary.ru/text/107/p.2/index.html)**. </p>**

**<p> Глава III. Крепость. </p>**

**<p> Глава IV. Поединок. </p>**

**<p> Глава V. Любовь. </p>**

**<p class="invisible"> Глава VI. Пугачевщина. </p>**

**<p> Глава VII. Приступ. </p> </body>**

Реализация данного кода представлена на рисунке 6.36.

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.36 - Использование свойства display со значением none

**6.6.5.5 Свойство visibility**

Как следует из названия, этого свойство управляет видимостью блока. Оно может принимать только два значения:

- visible – обычное состояние блока (по умолчанию);

* hidden – блок становится прозрачным, т.е. невидимым.

Свойство visibility:hidden очень похоже на свойство display:none. Однако свойство visibility:hidden делает элемент прозрачным, но при этом продолжает присутствовать на странице, а свойство display:none отображается так, как будто его никогда не было. Далее представлен пример использования свойства visibility.

**<head> <body color: SlateBlue;> <h2>Мир сказок.</h2><hr>**

**<h3> Сказки А.С. Пушкина:</h3>**

**<p> Сказка о рыбаке и рыбке.</p>**

**<p> Сказка о царе Салтане.</p>**

**<p style=“display:none”> Сказка о мёртвой царевне и о семи богатырях</p>**

**<p> Сказка о золотом петушке</p>**

**<p>Сказка о попе и о работнике его Балде</p><hr>**

**<h3> Сказки В.И. Даля: </h3><hr>**

**<p>Сказка “Девочка Снегурочка”. </p>**

**<p> Сказка “Лучший певчий”. </p>**

**<p style=“visibility:hidden”>Сказка “О дятле”.</p>**

**<p>Сказка “Лучший певчий”. </p> </body>**

Реализация данной HTML-страницы в браузере представлена на рисунке 6.37.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.37 – Визуальное отличие свойств “visibility:hidden” и “display:none”

Как видно из рисунка 6.37, в первой группе второй абзац с установленным свойством “display:none” просто отсутствует на экране, поэтому сразу после первого абзаца идет третий. Во второй группе абзац с установленным свойством “visibility:hidden” также отсутствует на экране, но его место остается незанятым. Иначе говоря, он также остается на своем месте, но является прозрачным, вследствие чего невидимым. С помощью свойства visibility можно организовывать раскрывающиеся меню. В начальном состоянии видимость всех пунктов этого меню устанавливается как “visibility:hidden”. Когда пользователь щелкает по команде главного меню, то включается видимость нужного пункта, с помощью объявления “visibility:visible”.

**6.****7 Создание панели навигации**

Панель навигации играет важную роль на сайте, поскольку позволяет перемещаться между страницами сайта или на внешние ресурсы. По сути, панель навигации представляет собой набор ссылок, часто в виде ненумерованного списка. Панели навигации бывают разных форм: вертикальные и горизонтальные, одноуровневые и многоуровневые, но в каждом случае элемент <a> находится в центре каждой навигации. Поэтому при создании панели навигации вы можете столкнуться с рядом трудностей из-за ограничений элемента ссылки. А именно, элемент <a> является встроенным, что означает, что вы не можете указать для него ширину, высоту или отступы. Ширина ссылки автоматически занимает необходимое ей пространство

**6.7.1 Вертикальное меню**

Для создания вертикального меню необходимо ссылку блочным элементом с помощью значения свойства display: block.

**<head> <style>**

**ul.nav{ margin-left: 1px; padding-left:1px; list-style: none;background-color: Lavender; border: 2px solid; border-color: MediumOrchid; }**

**ul.nav a { display: block; width: 8em; padding:8px;**

**background-color: Lavender; border: 2px solid; border-color: MediumOrchid;**

**text-decoration: none; }**

**ul.nav li:last-child a { border-bottom: 2px; }**

**</style> </head>**

**<body> <h2>Интернет-магазин женской обуви</h2>**

**<ul class="nav">**

**<li><a href="#">Сапоги</a></li>**

**<li><a href="#">Ботильоны</a></li>**

**<li><a href="#">Туфли</a></li>**

**<li><a href="#">Лоферы </a></li>**

**<li><a href="#">Босоножки</a></li> </ul> </body>**

Реализация данного кода представлена на рисунке 6.38.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.38 - Стилизация ссылок и навигация в CSS3

После установки свойства display: block можно определить у блока ссылки ширину, отступы и т.д.

**6.7.2 Горизонтальное меню**

Существует два метода создания горизонтального меню. Первый - использовать свойство float и создавать плавающие элементы из ссылок, которые обтекают друг друга слева. И второй способ - создать соединительную линию (строку ссылок), задав свойство display:inline-block.

**6.7.2.1 Использование float**

Алгоритм создания панели навигации с использованием float делится на два этапа. На первом этапе элемент li, содержащий ссылку, имеет значение float: left;. Это позволяет выровнять все элементы списка по ширине, так что правый элемент списка обтекает левый элемент списка.

Второй этап - настройка ссылки display:block на элемент, что дает возможность задавать ширину, отступы и вообще все особенности, характерные для блочных элементов.

**<head> <style> ul.nav{ margin-left: 1px; padding-left: 1px; list-style: none;}**

**.nav li {float: left;}**

**ul.nav a {display: block; width: 6em; padding:12px;**

**margin: 16px; background-color: LightCyan;**

**border: 2px; text-decoration: none; border-color: DarkTurquoise; text-align: center; }**

**ul.nav a:hover{background-color: Cyan; border-color: DarkTurquoise; } </style> </head>**

**<body> <h2>Магазин сумок</h2>**

**<ul class="nav"> <li><a href="#">Клатч</a></li>**

**<li><a href="#">Рюкзак</a></li>**

**<li><a href="#">Офисная сумка</a></li>**

**<li><a href="#">Пляжная сумка</a></li> </ul> </body>**

Реализация данного кода представлена на рисунке 6.39.

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.39 - Горизонтальный ряд ссылок в CSS3

**6.7.2.2 Inline и inline-block**

Чтобы создать горизонтальную панель навигации, нужно написать каждый элемент li строчным, т.е. установить для него display:inline. Затем для элемента link, который находится внутри элемента li, нужно установить display: inline-block.

**<head> <style> ul.nav{ margin-left: 1px; padding-left: 1px; list-style: none; }**

**.nav li { display: inline; }**

**ul.nav a {display: inline-block; width: 6em; padding:12px;**

**background-color: MistyRose; border: 2px; border-color: MediumVioletRed;**

**text-decoration: none; text-align: center; }**

**ul.nav a:hover {background-color: HotPink; border-color:MediumVioletRed; border: 2px;}**

**</style> </head> <body> <h2> Интернет-магазин спортивной обуви</h2>**

**<ul class="nav"> <li><a href="#"> Кроссовки /a></li>**

**<li><a href="#">Кеды</a></li> <li><a href="#"> Слипоны</a></li>**

**<li><a href="#">Мокасины</a></li> </ul> </body>**

Реализация данного кода представлена на рисунке 6.40.

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.40 - Горизонтальная панель навигации в CSS3

**6.8 Использование свойства box-sizing**

При работе с плавающими элементами и свойством float может возникнуть распространенная проблема: плавающие элементы выпадают со страницы. Рассмотрим наглядно эту проблему на конкретном примере.

**<head> <style>**

**#sidebar {float: left; width: 25%; padding: 10px; color: BlueViolet; }**

**#main{ border-left: 1px solid #ccc; width:75%; padding: 15px;**

**margin-left: 25%; color: DarkViolet; } </style> </head>**

**<body> <div id="sidebar"><h2>Новости театра музыкальной комедии города Оренбурга</h2>**

**<p>В оренбургском театре музыкальной комедии в феврале и марте пройдут спектали мьюзикла “Ромео и Джульетта”, основанные на трагедии, написанной Уильямом Шекспиром в 1594 году. В ней рассказывается о любви юноши и девушки из двух враждующих веронских родов - Монтекки и Капулетти.</p> </div>**

**<div id="main"> <h2>О мьюзикле Ромео и Джульета</h2>**

**<p> Действие спектакля разворачивается в Вероне, где непримиримо враждуют два рода – Монтекки и Капулетти. Однако любовь - искренняя и настоящая - всегда находит путь к сердцу. Так и случилось – юный Ромео влюбляется в прекрасную Джульетту. Мюзикл рассказывает о том, что кровная вражда, взаимная ненависть, столкновение мировоззрений всегда будут бессильны перед лицом истинной бесконечной Любви. Так было много лет назад, так происходит и сейчас. История о Ромео и Джульетте особенно актуальна “в наше скупое на искренние чувства время”.</p> </div></body>**

Реализация данного кода представлена на рисунке 6.41.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.41 - Выпадение плавающих блоков в HTML5

Как видно на рисунке 6.41 буквы вылезают из плавающего блока за границу. Это обусловлено тем, что для свойства box-sizing по умолчанию используется значение content-box, то есть при определении ширины и высоты элемента браузер будет прибавлять к значению свойств width и height также и внутренние отступы padding и ширину границы. В итоге это может привести к выпадению плавающих элементов из тех блоков, которые для них предназначены. Рекомендуется устанавливать для свойства box-sizing значение border-box, чтобы все элементы измерялись одинаково, а их ширина представляла только значение свойства width. Поэтому нередко в стилях добавляется следующий стиль.

**\* {box-sizing: border-box;}**

То есть значение box-sizing: border-box; устанавливается для всех элементов, и все они интерпретируются браузером одинаково. К примеру, добавляется этот стиль в выше определенную страницу и получится уже несколько другой результат (рисунок 6.42).

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.42 – Использование свойства border-box в CSS3

**6.9 Создание макета веб-страницы из блоковых элементов**

На основе информации, полученной в предыдущих темах, далее представлено описание создания макета простейшей веб-страницы. На первых этапах определяется базовая структура веб-страницы (рисунок 6.43).

**<head> <link href="styles.css" rel="stylesheet">**

**<div id="header"> <h1> Туристическая компания “White Lily”</h1>**

**<div id="nav"> <ul><li><a href="#">Поиск туров</a></li>**

**<li><a href="#">Поиск отелей</a></li> <li><a href="#">Горящие туры </a></li>**

**<li><a href="#">Выбрать турагентство </a></li> <li><a href="#">О нас </a></li> </ul> </div> </div> <div class="wrapper"> <div id="sidebar1" class="aside"> <h2>Направления туризма</h2> <p> Туристическая компания “White Lily”, являясь одним из признанных лидеров российского рынка путешествий и туризма, предлагает отдых в самых красивых местах**[**Турции**](https://pegast.ru/turkey-about)**,**[**России**](https://pegast.ru/russia-about)**,**[**Египта**](https://pegast.ru/egypt-about)**,**[**Туниса**](https://pegast.ru/tunisia-about)**,**[**Греции**](https://pegast.ru/greece-about)**,**[**Индии**](https://pegast.ru/india-about)**,**[**Таиланда**](https://pegast.ru/thailand-about)**,**[**Индонезии**](https://pegast.ru/indonesia-about)**,**[**Кипра**](https://pegast.ru/cyprus/about)**,**[**Кубы**](https://pegast.ru/cuba-about)**,**[**ОАЭ**](https://pegast.ru/uae-about)**. Идет постоянная работа по открытию новых направлений. </p></div>**

**<div id="sidebar2" class="aside"> <h2>Преимущества компании “White Lily”</h2> <p> Количество туристов, выбирающих отдых с “White Lily”, постоянно растет, а это лучшее подтверждение высококачественного обслуживания и профессионализма наших сотрудников. Мы подбираем самые популярные среди российских туристов отели всех ценовых категорий, обеспечиваем надежные и качественные авиаперевозки. </p>**

**<div> <div id="article"> <h2> О туристической компании “White Lily”</h2>**

**<p> Бренд “White Lily” представлен на рынке с 2001 года. Сегодня т компания занимает лидирующие позиции в туристической отрасли и позиционируется как марка надежности и качества. Компания организует групповые и индивидуальные туры на базе собственных чартерных программ и регулярных рейсов, занимается развитием конгресс-, спортивного и других видов туризма, а также активно продает авиабилеты в онлайн. Мы помогаем найти и приблизить счастье от путешествия вашей мечты. </p></div></div> <div id="footer">**

**<p>Contacts: WhiteLily @mail.ru</p><p>Copyright © WhiteLily.com, 2024</p> </div></body>**

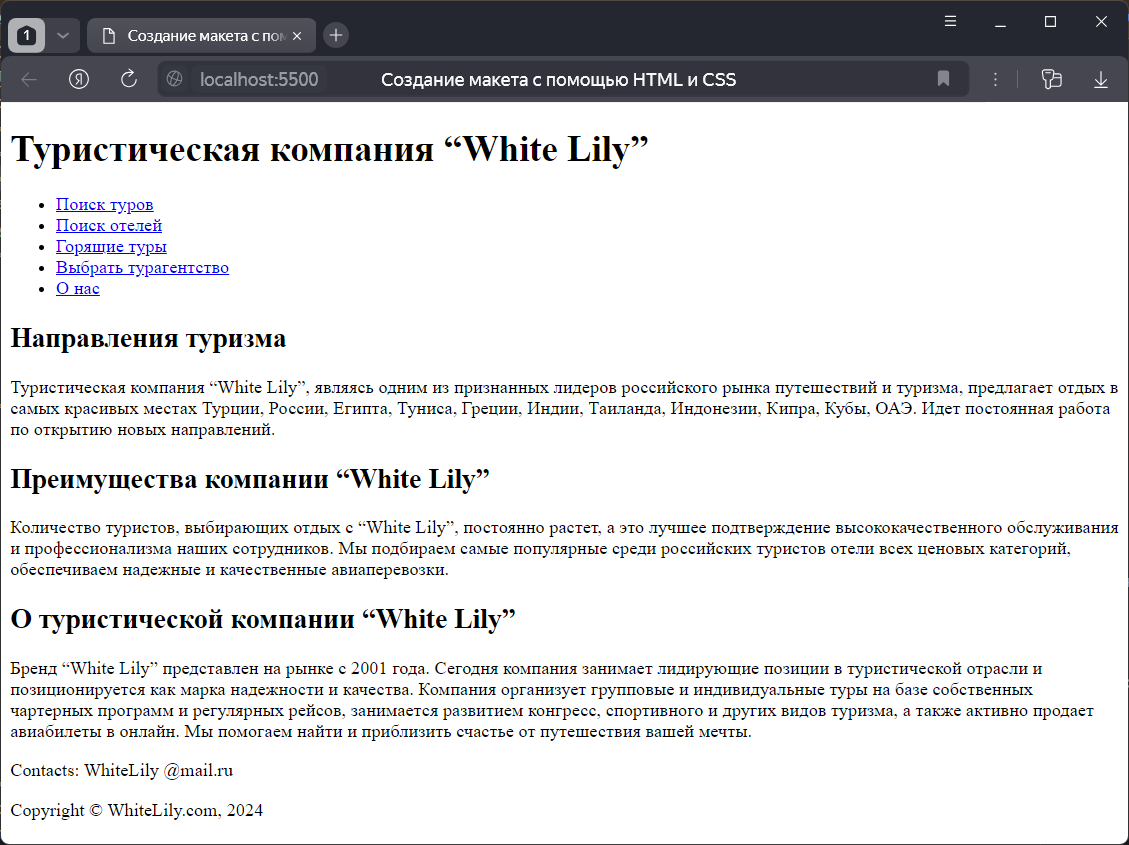


Рисунок 6.43 - Создание макета с помощью HTML и CSS

Вначале находится шапка сайта - блок с header, содержащий заголовок страницы и панель навигации. Далее идет блок wrapper, который содержит две боковые панели и блок основного контента страницы. Боковые панели также содержат некоторый контент, но главное, что они определены до основного блока. А в самом низу есть небольшой футер.

В начале веб-страницы определяется подключение файла styles.css, который будет стилизовать веб-страницу. Поэтому создается файл styles.css в том же каталоге, что и веб-страница, и определяется в нем следующее содержимое.

**\* { box-sizing: border-box;}**

**html, body, div, span, h1, h2, h3, h4, h5, h6, p, a, ul, li { margin: 0; padding: 0; border: 0; font-size: 100%; vertical-align: baseline; }**

**body { font-family: Verdana, Arial, sans-serif; background-color: #bcfafa; }**

**#header { background-color: LightBlue;}**

**#header h1 { font-size: 24px; text-transform: uppercase; font-weight: bold; padding: 30px 30px 30px 10px; clear: both;}**

**#nav { background-color: LightSkyBlue; border-top: 1px solid DarkTurquoise; border-bottom: 1px solid DarkTurquoise; }**

**#nav li {float: left; list-style: none; }**

**#nav a { display: block; color: black; padding: 10px 25px; text-decoration: none; border-right: 1px solid DarkTurquoise; }**

**#nav li:last-child a { border-right: none;}**

**#nav a:hover { font-weight: bold; }**

**#nav:after { content: " "; display: table; clear: both; }**

**.wrapper { background-color: #b4ebf6; }**

**.aside h2 { font-size: 0.95em; margin-top: 15px; }**

**.aside h3 { font-size: 0.85em; margin-top: 10px; }**

**.aside p, .aside li {font-size: .75em; margin-top: 10px; }**

**.aside li { list-style-type: none; }**

**#sidebar1 { float: left; width: 20%; padding: 0 10px 0 20px; }**

**#sidebar2 { float: right; width: 20%; padding: 0 20px 0 10px; }**

**#article { background-color: #9aeff1; border-left: 1px solid Cyan; border-right: 1px solid Cyan; margin-left: 20%; margin-right: 20%; padding: 15px; width: 60%; }**

**#article:after { clear: both; display: table; content: ''; }**

**#article h2 { font-size: 1.3em; margin-bottom: 15px; }**

**#article p { line-height: 150%; margin-bottom: 15px; }**

**#footer { border-top: 1px solid Azure; font-size: .8em; text-align: center; padding: 10px 10px 30px 10px; }**

**#nav ul, #header h1, .wrapper, #footer p { max-width: 1200px; margin: 0 auto; }**

**.wrapper, #nav, #header, #footer { min-width: 768px;}**

Реализация данного кода представлена на рисунке 6.44.

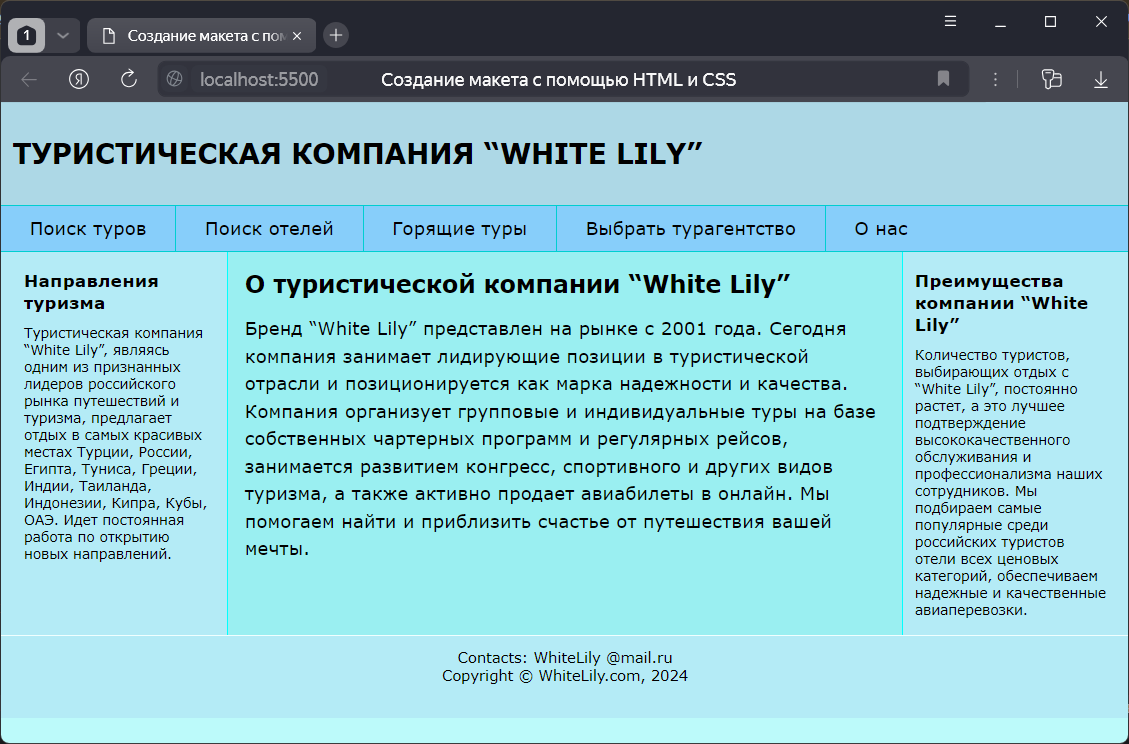
****

Рисунок 6.44 - Итоговое форматирование веб-страницы

Первые три стиля сбрасывают настройки стиля по умолчанию для используемых элементов, а также устанавливают стиль элемента body.

Следующая пара стилей управляет отображением заголовка и заголовка страницы:

**#header { background-color: LightBlue;}  
#header h1 { font-size: 24px; text-transform: uppercase; font-weight: bold; padding: 30px 30px 30px 10px; clear: both;}**

Следующий набор стилей управляет созданием горизонтальной панели навигации:

**#nav { background-color: LightSkyBlue; border-top: 1px solid DarkTurquoise; border-bottom: 1px solid DarkTurquoise; }  
#nav li {float: left; list-style: none; }  
#nav a { display: block; color: black; padding: 10px 25px; text-decoration: none; border-right: 1px solid DarkTurquoise; }  
#nav li:last-child a { border-right: none;}  
#nav a:hover { font-weight: bold; }  
#nav:after { content: " "; display: table; clear: both; }**

Далее создается горизонтальная панель навигации: элементам <li> присваивается float (float: left;), который размещает их в ряд, и каждая ссылка становится блочным элементом (display: block;). Далее настраивается средняя часть страницы, особенно боковые панели:

**.wrapper { background-color: #b4ebf6; }  
.aside h2 { font-size: 0.95em; margin-top: 15px; }  
.aside h3 { font-size: 0.85em; margin-top: 10px; }  
.aside p, .aside li {font-size: .75em; margin-top: 10px; } .aside li { list-style-type: none; }  
#sidebar1 { float: left; width: 20%; padding: 0 10px 0 20px; }  
#sidebar2 { float: right; width: 20%; padding: 0 20px 0 10px;}**

Стиль класса wrapper позволяет установить цвет фона для боковых панелей. Для каждой боковой панели ширина установлена ​​на уровне 20% ширины страницы. Проценты позволяют автоматически подстраивать ширину блоков под ширину окна браузера при его расширении или сжатии.

Ниже приведены стили для основного блока контента и нижнего колонтитула:

**#article { background-color: #9aeff1; border-left: 1px solid Cyan; border-right: 1px solid Cyan; margin-left: 20%; margin-right: 20%; padding: 15px; width: 60%; }  
#article:after { clear: both; display: table; content: ''; }  
#article h2 { font-size: 1.3em; margin-bottom: 15px; }  
#article p { line-height: 150%; margin-bottom: 15px; }  
#footer { border-top: 1px solid Azure; font-size: .8em; text-align: center; padding: 10px 10px 30px 10px; }**

Поскольку боковые панели имеют ширину в 20% каждая, то для главного блока устанавливается ширина в 60% и отступы справа и слева в 20%.

И в конце идет пара довольно важных стилей:

**#nav ul, #header h1, .wrapper, #footer p { max-width: 1200px; margin: 0 auto; }  
.wrapper, #nav, #header, #footer { min-width: 768px;}**

Вначале для ряда селекторов определена максимальная ширина 1200 пикселей. Это означает, что основные элементы страницы не будут выходить за пределы 1200 пикселей. А автоматические левые и правые поля позволят центрировать содержимое элементов. То есть при ширине браузера 1400 пикселей эти элементы шириной 1200 пикселей будут располагаться как бы посередине, а справа и слева будут отступы шириной (1400-1200)/2= 100 пикселей.

Второй стиль позволит создать фиксированную минимальную ширину для ряда элементов. То есть в результате при сжатии окна браузера боковые панели и основной блок будут выглядеть более или менее, а при сжатии окна до размеров менее 768 пикселей образуется полоса прокрутки.