Урок 5

Формирование блочной модели

Основные теги для верстки (div и span). Отступы элементов (margin и padding). Обтекаемые элементы. Позиционирование блоков.

Оглавление

- Свойство display
  - Значения свойства display
  - Схлопывания
- Формирование блочной модели
  - Обтекаемые элементы
- Позиционирование блоков
  - Абсолютное позиционирование
  - Относительное позиционирование
  - Фиксированное положение
  - Значение по умолчанию
  - Совмещенное значение
  - Z-index

Практика

Создание основной структуры сайта

Домашнее задание

Дополнительные материалы

Используемая литература

Свойство display

При помощи css можно изменить тип элемента, т.е. блочный тег можно сделать строчным, а строчный - блочным. Для этого существует css свойство - display. Вернёмся к предыдущему примеру и для элементов задать значение свойства display: inline;, а для - значение display: block;

```
div, span {
	border: 1px solid #000;
	width: 400px;
	height: 50px;
}

div {
	display: inline;
}

span {
	display: block;
}
```

В этом случае получается, что элементы поменялись местами, стал строчным элементом, и ему теперь невозможно задать ни ширину, ни высоту, а стал блочным, и ему теперь можно задать и ширину и высоту.

Значения свойства display

- none (скрыть);
- block (блочный);
- inline (строчный);
- inline-block (строчно-блочный);
- table-cell (ячейка таблицы);
- flex (гибкий).

Блочный элемент (display: block;) создает разрыв строки перед тегом и после него. Он образует прямоугольную область, по ширине занимающую всю ширину веб-страницы или блока-родителя, если для него не задано значение width.

Блочные элементы могут содержать внутри себя элементы любого типа. Нельзя размещать блочные элементы внутри строчных, за исключением элемента <img>. Для блочных элементов можно задавать margin и padding.

Свойства width и height устанавливают ширину и высоту области содержимого элемента. Фактическая ширина элемента складывается из ширины полей (внутренних отступов), границ и внешних отступов.

Строчные элементы (display: inline;) не создают блоки, они отображаются на одной строке с содержимым рядом стоящих тегов. Строчные элементы являются потомками блочных элементов. Они игнорируют верхние и нижние margin и padding, но если для элемента задан фон, он будет распространяться на верхний и нижний padding, заходя на соседние строки текста.

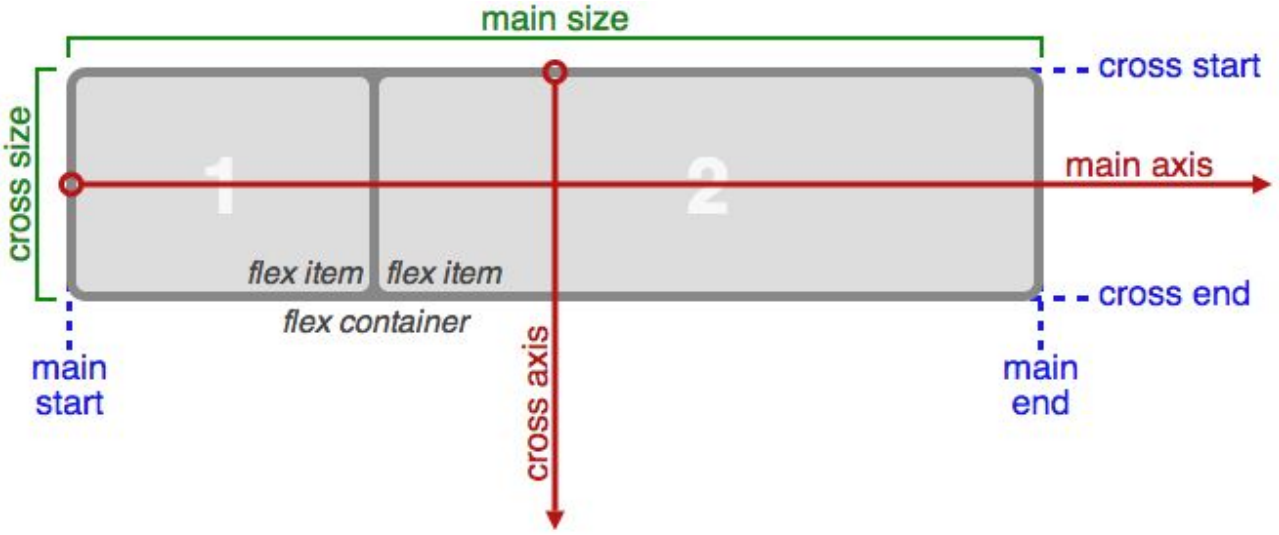
Ширина и высота строчного элемента зависит только от его содержания, задать размеры с помощью CSS нельзя. Можно увеличить расстояние между соседними элементами по горизонтали с помощью горизонтальных полей и отступов.

Существует ещё одна группа элементов, которые браузер обрабатывает как **строчно-блочные** (display: inline-block;). Такие элементы являются встроенным, но для них можно задавать поля, отступы, ширину и высоту.

Современные браузеры (IE8+) позволяют описывать таблицу любыми элементами, если поставить им соответствующие значения display. Это хорошо для семантической вёрстки и позволяет избавиться от лишних тегов.

Для блока возможно задать значение ячейки таблицы (table-cell). Внутри ячеек свойство vertical-align выравнивает содержимое по вертикали. CSS не требует, чтобы вокруг table-cell была структура таблицы: table-row и т.п. Может быть просто такой одинокий DIV, это допустимо. При этом он ведёт себя как ячейка TD, то есть подстраивается под размер содержимого и умеет вертикально центрировать его при помощи vertical-align.

flexbox — это целый модуль, а не просто единичное свойство, он объединяет в себе множество свойств. Некоторые из них должны применяться к контейнеру (родительскому элементу, так называемому flex-контейнеру), в то время, как другие свойства применяются к дочерним элементам или flex-элементам.



Если обычный layout основывается на направлениях потоков блочных и inline-элементов, то flex-layout основывается на «направлениях flex-потока». Ознакомьтесь с этой схемой из спецификации, разъясняющей основную идею flex-layout-ов.

В основном элементы будут распределяться либо вдоль главной оси (от main-start до main-end), либо вдоль поперечной оси (от cross-start до cross-end).

- main-axis - главная ось, вдоль которой располагаются flex-элементы. Обратите внимание, она необязательно должна быть горизонтальной, всё зависит от свойства justify-content (см. ниже).
- main-start | main-end - flex-элементы размещаются в контейнере от позиции main-start до позиции main-end.
- main size - ширина или высота flex-элемента в зависимости от выбранной основной величины. Основная величина может быть либо шириной, либо высотой элемента.
- cross axis - поперечная ось, перпендикулярная к главной. Её направление зависит от направления главной оси.
- cross-start | cross-end - flex-строки, заполняются элементами и размещаются в контейнере от позиции cross-start и до позиции cross-end.

- cross size - ширина или высота flex-элемента в зависимости от выбранной размерности равняется этой величине. Это свойство совпадает с width или height элемента в зависимости от выбранной размерности.

Элементы в контейнере поддаются выравниванию при помощи свойства justify-content вдоль главной оси. Это свойство принимает целых пять разных вариантов значений.

- flex-start (default): гибкие элементы выравниваются по началу главной оси;
- flex-end: элементы выравниваются по концу главной оси;
- center: элементы выравниваются по центру главной оси;
- space-between: элементы занимают всю доступную ширину в контейнере, крайние элементы вплотную прижимаются к краям контейнера, а свободное пространство равномерно распределяется между элементами;
- space-around: гибкие элементы выравниваются таким образом, что свободное пространство равномерно распределяется между элементами. Но стоит отметить, что пространство межу краем контейнера и крайними элементами будет в два раза меньше, чем пространство между элементами в середине ряда.

Мы также имеем возможность выравнивания элементов по cross оси. Применив свойство align-items, которое принимает также пять разных значений, можно добиться интересного поведения. Это свойство позволяет выравнивать элементы в строке относительно друг друга.

- flex-start: все элементы прижимаются к началу строки;
- flex-end: элементы прижимаются к концу строки;
- center: элементы выравниваются по центру строки;
- baseline: элементы выравниваются по базовой линии текста;
- stretch (default): элементы растягиваются, заполняя полностью строку;
- Еще одно похожее свойство на предыдущее это align-content. Только оно отвечает за выравнивание целых строк относительно гибкого контейнера. Оно не будет давать эффекта. если гибкие элементы занимают одну строку.

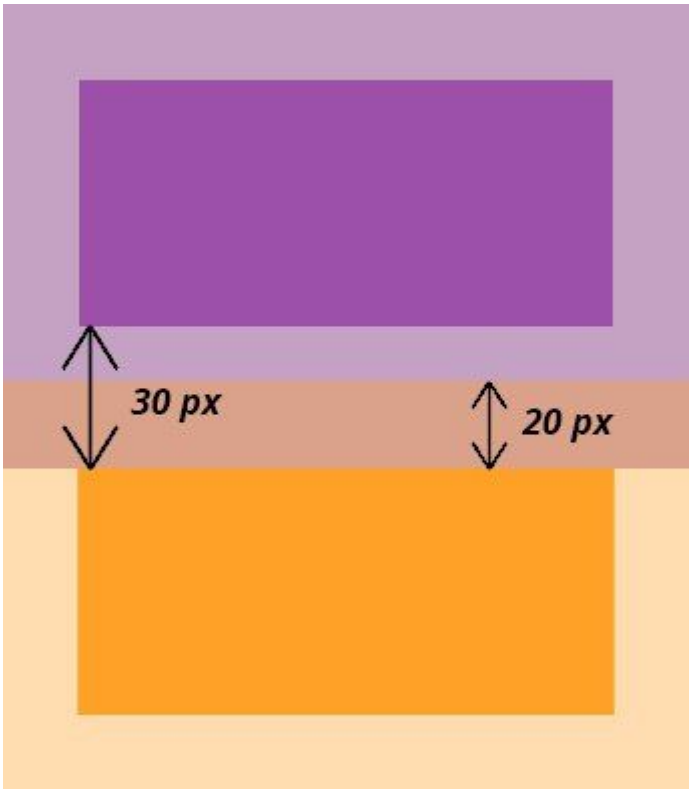
Свойство принимает шесть разных значений.

- flex-start: все линии прижимаются к началу cross-оси;
- flex-end: все линии прижимаются к концу cross-оси;
- center: Flex-элементы выравниваются по центру flex-контейнера.
- space-between: линии распределяются от верхнего края до нижнего, оставляя свободное пространство между строками, крайние же строки прижимаются к краям контейнера;
- space-around: линии равномерно распределяются по контейнеру;
- stretch (default): линии растягиваются, занимая все доступное пространство.

Одно из основных свойств является flex-basis. С помощью этого свойства мы можем указывать базовую ширину гибкого элемента. По умолчанию имеет значение auto. Это свойство тесно связано с flex-grow и flex-shrink, о которых будет рассказано чуть позже. Принимает значение ширины в px, %, em и остальных единицах. По сути, это не строго ширина гибкого элемента, это своего рода отправная точка, относительно которой происходит растягивание или усадка элемента. В режиме auto элемент получает базовую ширину относительно контента внутри него.

Схлопывания

Когда два или более вертикальных margin соприкасаются, они сливаются, при этом ширина общего отступа равна ширине большего из исходных отступов.



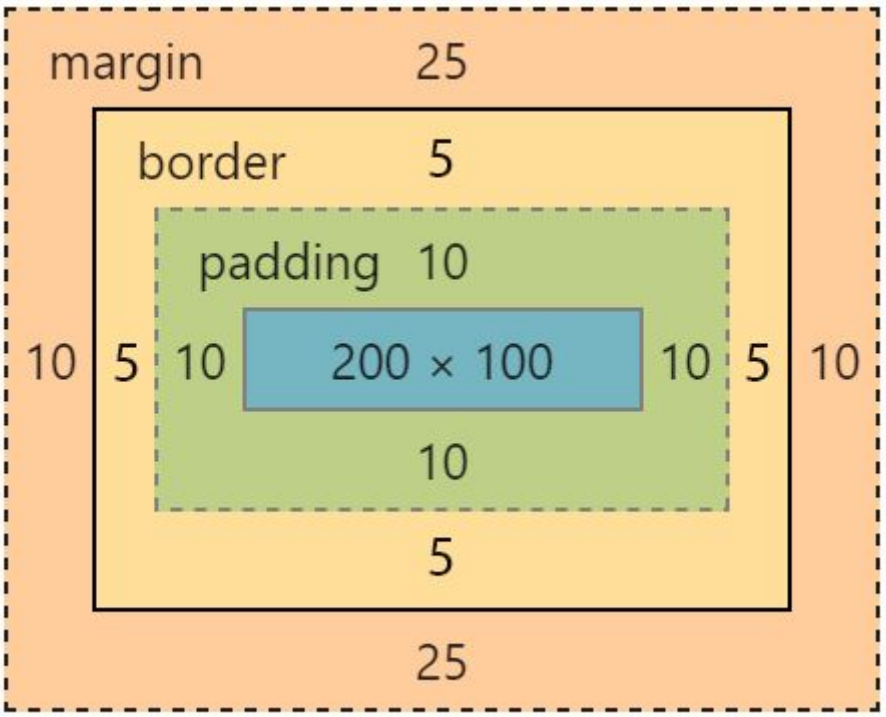
Слияние выполняется только для блочных элементов в нормальном потоке документа. Внешние вертикальные отступы строчных, плавающих и абсолютно позиционированных элементов не сливаются. Чтобы получить желаемый промежуток, можно задать, например, для верхнего элемента padding-bottom, а для нижнего элемента — margin-top.

Существуют исключения для схлопывания:

- с блоками, которым присвоено float;
- с основными элементами (html, body);
- для блоков, которым присвоено свойство и значение position:absolute;
- для строчных элементов.

Формирование блочной модели

На первый взгляд может показаться, что width – это окончательная ширина элемента, height - это окончательная высота элемента. На самом деле это не так, width и height - это не окончательные размеры элемента. Для того, чтобы вычислить размеры, необходимо учитывать следующие моменты.



Если внимательно ознакомиться с данной схемой, то можно сделать вывод, что ширина блока складывается из следующих свойств:

```
margin-left +
border-left +
padding-left +
width +
padding-right +
border-right +
margin-right
```

Соответственно, высота из следующих:

```
margin-top +
border-top +
padding-top +
height +
padding-bottom +
border-bottom +
margin-bottom
```

Внутренний отступ или поле элемента (padding) добавляет отступы внутри элемента, между его основным содержимым и его границей. Если для элемента задать фон, то он распространится также

и на поля элемента. Внутренний отступ не может принимать отрицательных значений, в отличие от внешнего отступа.

Внешний отступ (margin) добавляет отступы за границами элемента, создавая тем самым промежутки между элементами. Они всегда остаются прозрачными, и через них виден фон родительского элемента. Значения padding и margin задаются в следующем порядке: верхнее, правое, нижнее и левое.

Граница или рамка элемента задается с помощью свойства border. Если цвет рамки не задан, она принимает цвет основного содержимого элемента, например, текста. Если рамка имеет разрывы, то сквозь них будет проступать фон элемента.

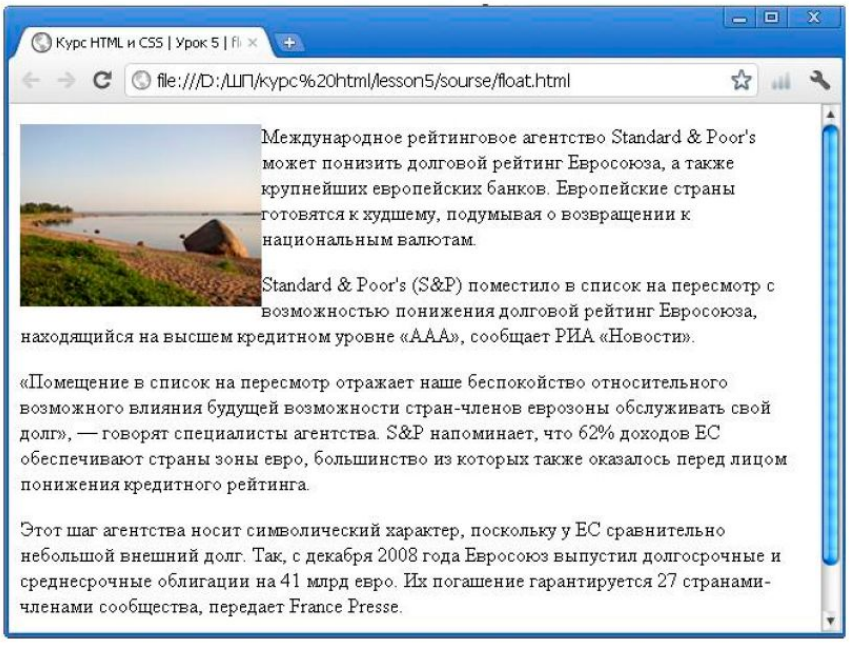
Внешние, внутренние отступы и рамка элемента не являются обязательными, по умолчанию их значение равно нулю. Тем не менее, некоторые браузеры добавляют к этим свойствам положительные значения по умолчанию на основе своих таблиц стилей.

Обтекаемые элементы

Обтекаемые элементы или как их ещё называют «плавающие», используются для реализации обтекания текстом изображений, создания врезок, и даже создания много-столбцовых компоновок.

Также обтекаемые элементы активно используются при верстке веб-страниц, и при помощи их возможно заменить табличную верстку на верстку слоями. Для того, чтобы задать обтекание, в CSS существует только одно свойство float, которое может принимать всего два значения - это left и right.

В следующем примере картинке, т.е. тегу <img />, заданно свойство float: left;



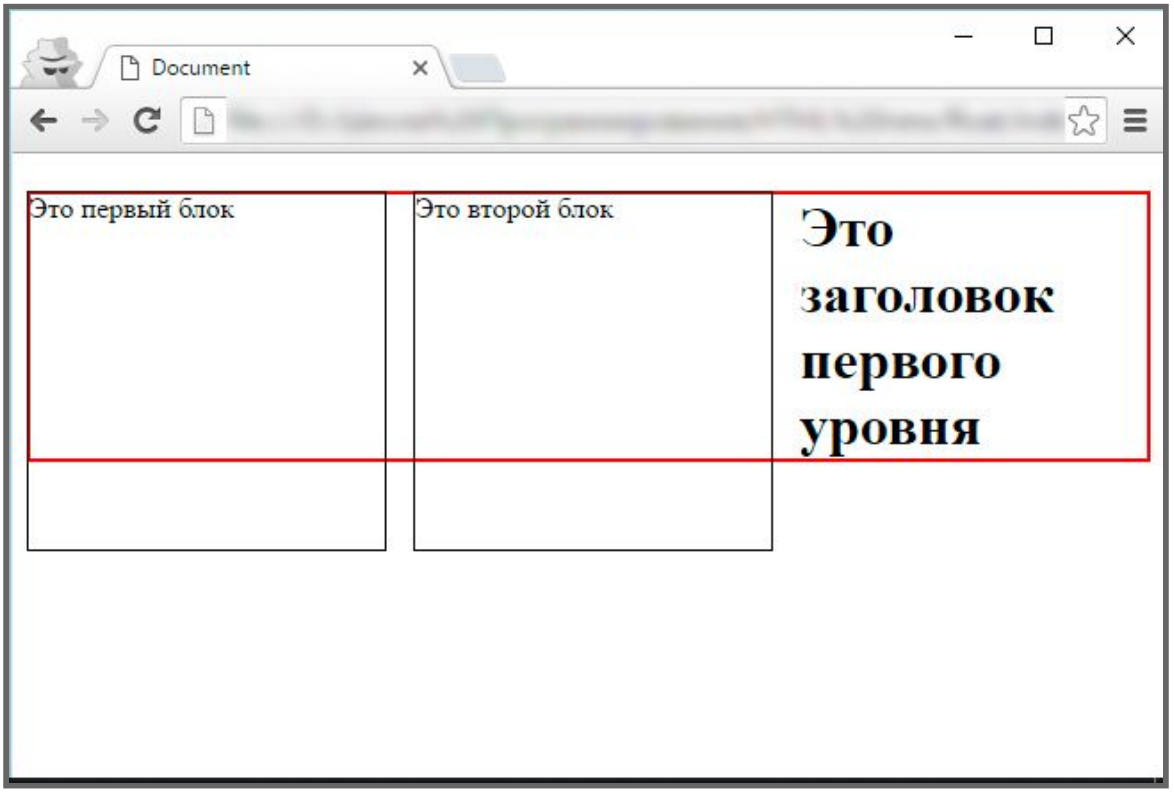
В этом случае картинка займет положение слева и позволит любым элементам, будь они строчные или блочные, обтекать себя справа.

Далее рассмотрим следующий пример. Создадим два элемента и один заголовок первого уровня <h1>

HTML	CSS

Это первый блок
 	
 Это второй блок
 	
 Это заголовок первого уровня
	div {
	width: 200px;
	height: 200px;
	margin-right: 15px;
	border: 1px solid #000;
	float: left;
	}
	h1 {
	border: 1px solid #f00;
	}

У обоих элементов задано свойство float: left;, т.е. они должны занимать левое положение и позволять обтекать себя справа. Посмотрим на работу этого примера в браузере.



Разберемся, что же произошло. Элементы находятся на одной линии по горизонтали, что и ожидаемо, т.к. у них задано свойство float: left;. Первый занял положение слева, позволил обтекать себя справа. Второй , соответственно, в свою очередь также позволил обтекать себя справа. Заголовок первого уровня находится справа второго элемента , но его рамка обрамляет также оба элемента . Это происходит потому, что у свойства float есть особенность: элементы, которым заданно это свойство, начинают притягивать к себе все близлежащие элементы и заставляют их тоже участвовать в обтекании. Но с этим можно бороться.

Рассмотрим две ситуации.

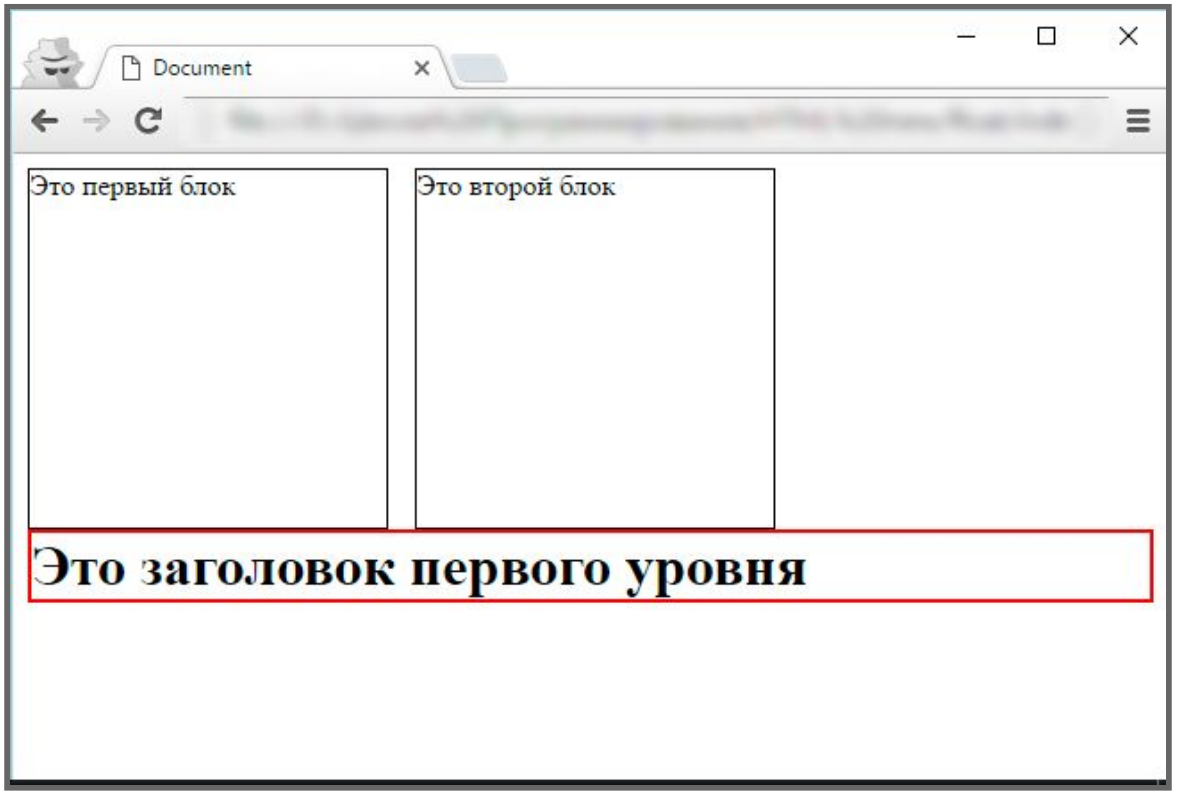
1. Заголовок не должен участвовать в обтекании и должен находиться под элементами .

В этом случае необходимо применить запрет на обтекание. Для этого в css существует свойство clear. Оно может принимать три значения - это left, отменяющее обтекание с левого края, right - с правого края, и значение both - которое отменяет обтекание с обеих сторон. Добавим свойство clear со значением both для заголовка первого уровня.

HTML	CSS

Это первый блок
 	
 Это второй блок
 	
 Это заголовок первого уровня
	div {
	width: 200px;
	height: 200px;
	margin-right: 15px;
	border: 1px solid #000;
	float: left;
	}
	h1 {
	border: 1px solid #f00;
	clear: both;
	}

Если запустить данный код в браузере, то заголовок уже не будет участвовать в обтекании, а будет находиться под элементами .



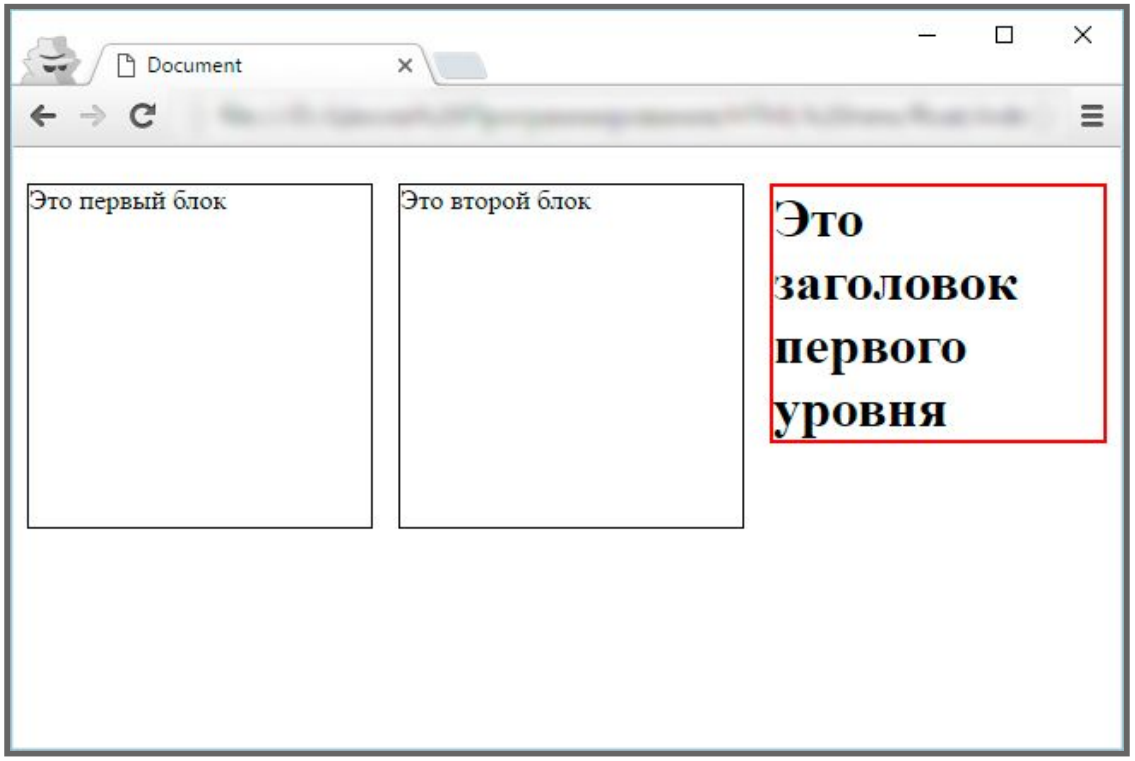
2. Заголовок остается на том же месте, где он сейчас находится, но рамка должна обрамлять только сам заголовок.

Для решения этой задачи, поможет css свойство overflow. Оно определяет, как будет вести себя блочный элемент в случае его переполнения, и при значении hidden отображает только содержимое этого элемента.

HTML	CSS

Это первый блок
 	
 Это второй блок
 	
 Это заголовок первого уровня
	div {
	width: 200px;
	height: 200px;
	margin-right: 15px;
	border: 1px solid #000;
	float: left;
	}
	h1 {
	border: 1px solid #f00;
	overflow: hidden;
	}

Если запустить этот пример в браузере, то заголовок остаётся на том же месте, и рамка теперь обрамляет только элемент <h1>.



Позиционирование блоков

Идея, лежащая в основе позиционирования, довольно проста. Позиционирование позволяет точно определить, где появятся блоки относительно другого элемента или относительно окна браузера. По

умолчанию все элементы располагаются последовательно один за другим в том порядке, в котором они определены в html-документе ({position: static}). Свойство не наследуется.

Блочный элемент (p, div, h1 и др.) занимает 100% ширины родительского элемента (по умолчанию — body). Поэтому блочные элементы отображаются один под другим в соответствии с разметкой страницы.

Строчный элемент (em, strong, span и др.) занимает ширину, которая соответствует ширине содержимого внутри него. Поэтому строчные элементы отображаются рядом друг с другом.

Свойство position вместе со значениями top, right, bottom и left отображает элемент с нарушением обычного порядка, смещая его на заданное расстояние. При позиционировании элементов можно использовать как положительные, так и отрицательные значения. Таким образом, существуют 4 вида позиционирования.

Абсолютное позиционирование

При абсолютном позиционировании элемент не существует в потоке документа, и его положение задаётся относительно краёв браузера. Задать этот тип можно через значение absolute свойства position. Координаты указываются относительно краёв окна браузера, называемого «видимой областью»

Для режима характерны следующие особенности:

- Ширина слоя, если она не задана явно, равна ширине контента плюс значения полей, границ и отступов.
- Слой не меняет своё исходное положение, если у него нет свойств right, left, top и bottom.
- Свойства left и top имеют более высокий приоритет по сравнению с right и bottom. Если left и right противоречат друг другу, то значение right игнорируется. То же самое касается и bottom.
- Если left задать отрицательное значение, то слой уйдёт за левый край браузера, полосы прокрутки при этом не возникнет. Это один из способов спрятать элемент от просмотра. То же относится и к свойству top, только слой уйдёт за верхний край.
- Если left задать значение больше ширины видимой области или указать right с отрицательным значением, появится горизонтальная полоса прокрутки. Подобное правило работает и с top, только речь пойдёт о вертикальной полосе прокрутки.
- Одновременно указанные свойства left и right формируют ширину слоя, но только если width не указано. Стоит добавить свойство width, и значение right будет проигнорировано. Аналогично произойдёт и с высотой слоя, только уже участвуют свойства top, bottom и height.
- Элемент с абсолютным позиционированием перемещается вместе с документом при его прокрутке.

Относительное позиционирование

relative (относительное позиционирование) - элемент будет смещаться относительно его определенного в настоящее время положения, и при этом его место будет оставаться не заполненным. Добавление свойств left, top, right и bottom изменяет позицию элемента и сдвигает его в ту или иную сторону от первоначального расположения. Положительное значение left определяет сдвиг вправо от левой границы элемента, отрицательное — сдвиг влево. Положительное значение top задаёт сдвиг элемента вниз, отрицательное — сдвиг вверх.

Свойства bottom и right производят обратный эффект. При положительном значении right сдвигает элемент влево от его правого края, при отрицательном — сдвигает вправо. При положительном значении bottom элемент поднимается вверх, при отрицательном опускается вниз.

Для относительного позиционирования характерны следующие особенности:

- Этот тип позиционирования не применим к элементам таблицы вроде ячеек, строк, колонок и др.;
- При смещении элемента относительно исходного положения, место, которое занимал элемент, остаётся пустым и не заполняется ниже или вышележащими элементами.

Фиксированное положение

Фиксированное положение слоя задаётся значением fixed свойства position и по своему действию похоже на абсолютное позиционирование. Но, в отличие от него, привязывается к указанной свойствами left, top, right и bottom точке на экране и не меняет своего положения при прокрутке веб-страницы. Ещё одна разница от absolute заключается в том, что при выходе фиксированного слоя за пределы видимой области справа или снизу от неё, не возникает полос прокрутки.

Применяется такой тип позиционирования для создания меню, вкладок, заголовков, в общем, любых элементов, которые должны быть закреплены на странице и всегда видны посетителю.

Значение по умолчанию

Если для элемента свойство position не задано или его значение static, элемент выводится в потоке документа как обычно. Иными словами, элементы отображаются на странице в том порядке, как они идут в исходном коде HTML.

Свойства left, top, right, bottom если определены, игнорируются.

Совмещенное значение

position:relative + position:absolute

Назначив родительскому блоку относительное позиционирование (position: relative), мы сможем позиционировать любые дочерние элементы относительно его границ. Если у элемента есть позиционированный предок, то position: absolute работает относительно него, а не относительно документа. Нужно пользоваться таким позиционированием с осторожностью, т.к. оно может перекрыть текст. Этим оно отличается от float.

Z-index

Любые позиционированные элементы на веб-странице могут накладываться друг на друга в определенном порядке, имитируя тем самым третье измерение, перпендикулярное экрану. Каждый элемент может находиться как ниже, так и выше других объектов веб-страницы, их размещением по z-оси и управляет z-index. Это свойство работает только для элементов, у которых значение position задано как absolute, fixed или relative.

В качестве значения используются целые числа (положительные, отрицательные и ноль). Чем больше значение, тем выше находится элемент по сравнению с теми элементами, у которых оно меньше. При равном значении z-index, на переднем плане находится тот элемент, который в коде HTML описан ниже. Хотя спецификация и разрешает использовать отрицательные значения z-index, но такие элементы не отображаются в браузере Firefox до версии 2.0 включительно.

Кроме числовых значений применяется auto — порядок элементов в этом случае строится автоматически, исходя из их положения в коде HTML и принадлежности к родителю, поскольку

дочерние элементы имеют тот же номер, что их родительский элемент. Значение inherit указывает, что оно наследуется у родителя.

Практика

Создание основной структуры сайта

```
<div class="container">
    <div class="header"></div>
    <div class="content"></div>
    <div class="footer"></div>
</div>
```

```
.container {
    width: 800px;
    margin: 0 auto;
}
.header {
    background-color: #2118FF;
    height: 100px;
}
.content {
    background-color: #E1E0E1;
    height: 400px;
}
.footer {
    background-color: #905BAE;
    height: 100px;
}
```

Домашнее задание

Создание блочной структуры сайта, позиционирование элементов

1. Главная страница:
    a. Создать родительский блок с классом container, задать значение ширины и расположить блок по центру экрана
    b. Задать блоку с классом header значение высоты.
    c. Отодвинуть логотип от левого края.
    d. Из вертикального меню сделать горизонтальное.
    e. Разместить меню справа от логотипа.
    f. Прижать меню к правому краю.
2. Footer (на всех страницах:)
    a. Задать высоту.
    b. Поменять цвет фона.
    c. Прижать текст “Все права защищены” к правому краю и отодвинуть от верхней и правой границы.
3. Страница просмотра товара каталога:
    a. Разместить краткое описание товара справа от картинки
    b. Добавить кнопку “Купить”
    c. Присвоить наведение и нажатие на кнопку “Купить”
4. Страница каталога
    a. Создать блоки для картинки и ссылки на просмотр товара каталога
    b. Расположить элементы каталога горизонтально.
5. Доделать то, что не успели в прошлых уроках.
6. *Расположить элементы на ваше усмотрение.
7. *Страница контактов:
    a. Ширина карты 100%,
    b. Высота карты 374px.
8. *Добавить фоновые изображения для header на каждой странице.

На данном этапе уже можно приступить к расположению элементов в соответствии с макетом.

Задачи со * предназначены для продвинутых учеников, которым мало сделать обычное ДЗ.

Дополнительные материалы

1. Свойство display
2. Статья про display
3. Формирование блочной модели
4. Позиционирование за 10 шагов

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. http://htmlbook.ru/css/display
2. https://learn.javascript.ru/display
3. http://htmlbook.ru/css/z-index
4. http://www.pvsm.ru/css3/73795

5. http://html5book.ru/css3-flexbox/
6. http://htmlbook.ru/samlayout/blochnaya-verstka/blochnaya-model
7. https://ru.wikibooks.org/wiki/CSS/%D0%91%D0%BB%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C
8. http://htmlbook.ru/samlayout/blochnaya-verstka/pozitsionirovanie-elementov