Web-страницы. Язык HTML и др.

Оформление документа. Стилевые файлы (CSS)

Архитектура документа.

1. Фреймы.
2. Табличный дизайн.
3. Блочный дизайн.

Фреймы

1. Фрейм (от англ. frame — рамка) — отдельный, законченный HTML-документ, который вместе с другими HTML-документами может быть отображён в окне браузера.
2. Фреймы по своей сути очень похожи на ячейки таблицы, однако более универсальны. Фреймы разбивают веб-страницу на отдельные миникадры, расположенные на одном экране, которые являются независимыми друг от друга. Каждое окно может иметь собственный адрес. При нажатии на любую из ссылок, расположенных в одном фрейме, можно продолжать видеть страницы в других окнах.
3. В настоящее время использование фреймов для публичных сайтов не рекомендовано. Главным образом это связано с принципом работы поисковых машин, которые приводят пользователя к HTML-документу, являющемуся согласно задумке лишь одним из фреймов того, что автору сайта хотелось бы представить. .

Табличная вёрстка

1. элементы сайта располагаются по ячейкам. Создаётся файл-шаблон с разметкой и используется как основа для всех остальных страниц. Фактически от файла к файлу меняется только основной контент. Шапка сайта, его низ и меню берутся из уже готового шаблона и обычно остаются неизменными.
2. проста и во всех браузерах выглядит одинаково. Но состоящие из таблиц страницы много весят, медленно загружаются и содержат избыточный код. К тому же структура таблиц позволяет создать только строгий дизайн.
3. вёрстка этого типа делается просто: в теле документа создаётся таблица (и ничего, кроме неё), в ячейки которой добавляется весь контент, вся страница состоит из таблиц, а вся выводимая информация находится внутри их ячеек. Даже область с основным содержимым страницы (например, со статьёй) — это всего лишь ячейка, текст из которой отображается на экране.

HTML-документ не обязательно должен состоять из одной таблицы — их можно вкладывать друг в друга, получая более интересное оформление.


Шапка
Меню 1
Текст страницы
Меню 2
Низ сайта

Блочная вёрстка

1. лишена недостатков табличной — поисковыми системами она индексируется лучше, её код не такой развесистый, да и блоки ** (**«слои»), изначально задумывались универсальными, то есть «для всего», тогда как **<table>** — это таблица, которую нужно использовать для отображения табличных данных и не более того.
2. ощутимый минус блочной вёрстки — сделанные на ней сайты могут по-разному отображаться в обозревателях. Чтобы этого избежать, нужно делать вёрстку «кроссбраузерной», то есть одинаково отображаемой любым обозревателем.

ПРИМЕР

Наиболее популярным является деление макетов по ширине и количеству колонок. Выделяют следующие типы макетов, связанных с шириной:

- фиксированные;
- резиновые;
- эластичные;
- адаптивные;
- комбинированные.

Фиксированный макет располагается по центру окна браузера, а его ширина ограничивается заданными размерами в пикселах.

При создании *резинового макета* задается в процентах ширина колонок таким образом, что макет занимает всю свободную ширину окна браузера.

Эластичный макет по своему виду может не отличаться от фиксированного или резинового макета. Размер элементов задаётся в em, привязанных к размеру шрифта. Значение em можно использовать не для всех элементов, оставляя ширину некоторых фиксированной.

Адаптивный макет подстраивается под разрешение монитора и окна браузера, меняя при необходимости ширину макета, число колонок, размеры изображений и текста. Для этого заготавливается несколько стилевых правил или файлов под разный диапазон разрешений, выбор правил происходит через скрипты или CSS, которые и определяют нужную для этого информацию о пользователе.

Комбинированный макет предполагает использование разной ширины для отдельных частей страницы, например, шапку и подвал делают резиновыми, а контент фиксированным.

Блоки (DIV и SPAN)

Эти теги играют особую роль для CSS. Они позволяют выделять в документе отдельные области, задавая для них специфические свойства. Все, что нужно сделать — это поместить теги, принадлежащие конструируемой области внутрь `<DIV>...</DIV>` или `<SPAN>...</SPAN>`. Разница между `<DIV>` и `<SPAN>` только в одном: после блока `<DIV>` браузер переходит на новую строку, а после блока `<SPAN>` — нет. Использование тега `<SPAN>` позволяет тем самым задавать стилевые свойства даже отдельным словам и буквам.

.Где

начало того конца,
которым оканчивается начало?

.Где начало того конца, которым
оканчивается начало?

Блоки могут отображать любое содержимое!

```
<div id="mix">
  <p>Lorem ipsum dolor sit amet,
  consectetur adipiscing elit. Nam nunc
  libero, semper ac feugiat sed,
  sollicitudin et mauris.
  </p>
  <img src="vasya.jpg">
  <table>
  <tr><td>1</td></td></tr>
  </table>
</div>
```

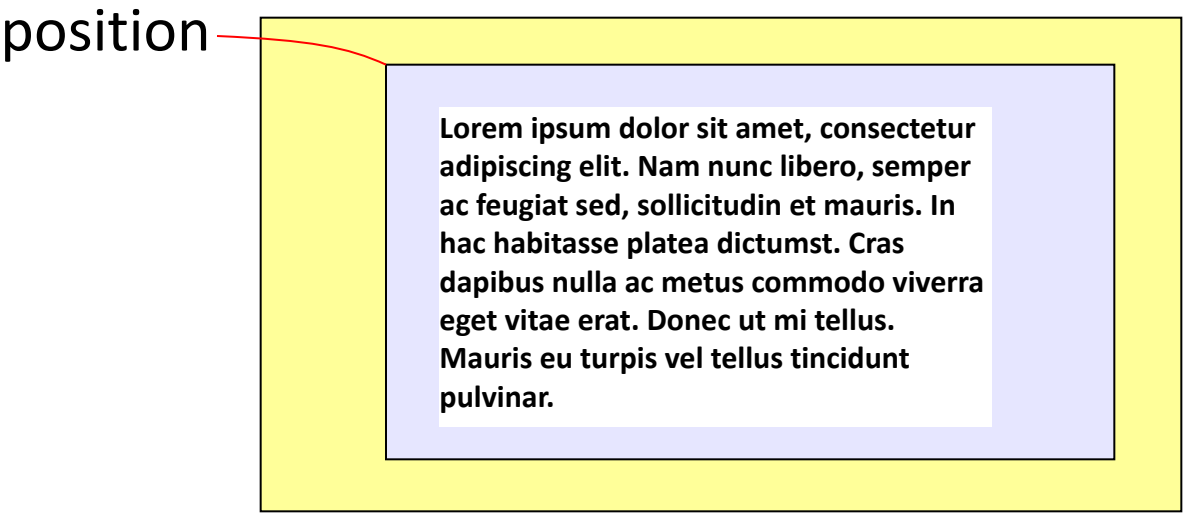
Позиционирование

Позиционирование — это управление координатами размещения элемента в окне браузера.

CSS предлагает для позиционирования свойство position. У этого свойства могут быть три значения absolute (абсолютное позиционирование), relative (относительное позиционирование) и static (статическое позиционирование).

Значение static размещает элемент на странице так, как он располагался бы без всякого позиционирования, поэтому использование этого значения не дает ничего нового.

Блоки (DIV) – позиционирование



При помощи CSS можно отображать элементы на экране, используя реальные координаты, отсчитываемые от левого верхнего угла окна браузера. Такую возможность предоставляет стилевое свойство `position` со значением `absolute`. Сами координаты задаются при помощи свойств `left` (горизонтальная координата) и `top` (вертикальная координата).

```
{
  position: absolute; top:10; left:10;
  color:red;font-weight:bolder;
  font-size:40pt; background:aqua
}
```

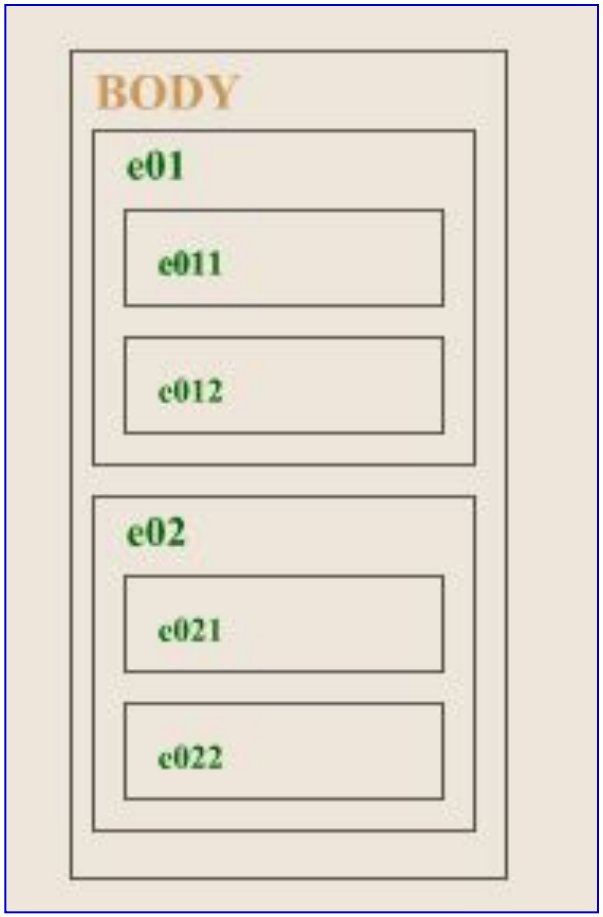
Иерархия кода страницы

Иерархия — это структура, в которой содержимое “упаковано” по пакетикам, вложенным друг в друга наподобие матрешек.
Все элементы страницы расположены внутри элемента BODY (блока <BODY>...</BODY>). Таким образом, элемент BODY является родителем всех других элементов страницы.

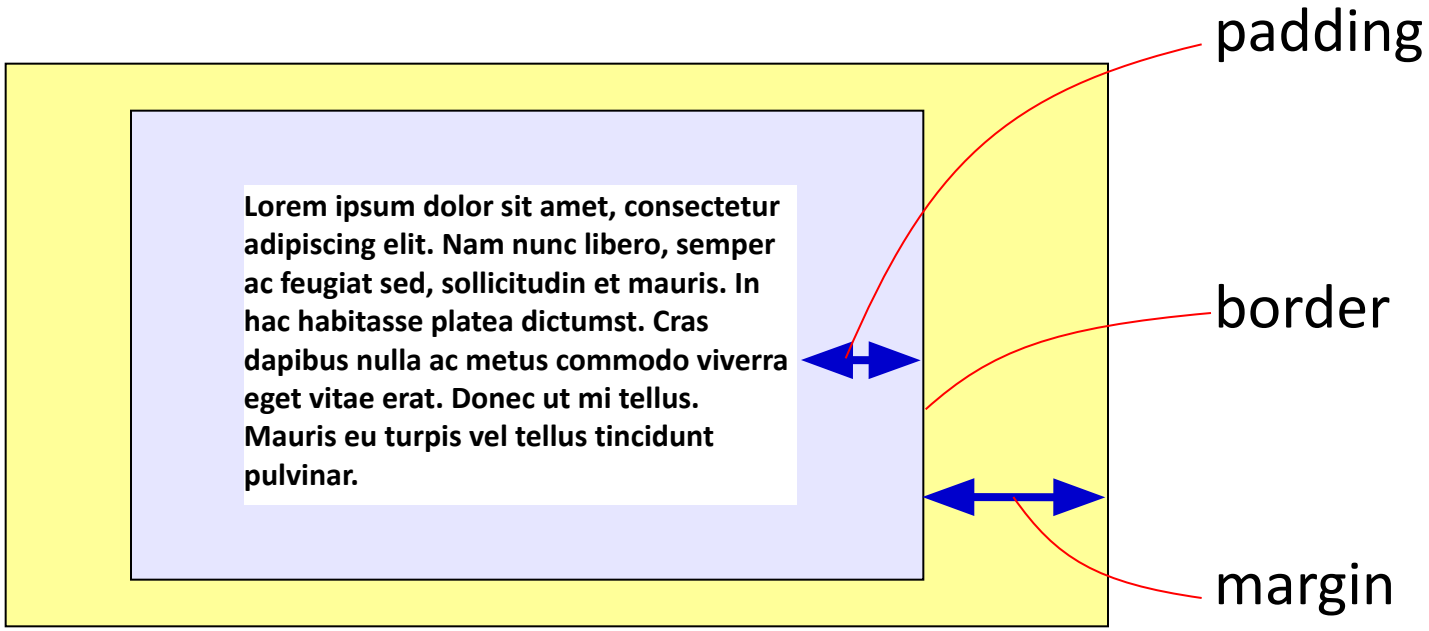
Позиционирование элементов выполняется с учетом иерархии кода. Вот почему так важно отчетливо представлять HTML-структуру документа. Опытные программисты всегда записывают код лесенкой (как в приведенном примере), и у них возникает гораздо меньше проблем при программировании, тестировании и отладке своих страниц.

Пример:

```
<BODY id=e0>
  <DIV id=e01>
    <DIV id=e011>
      ...
    </DIV>
    <DIV id=e012>
      ...
    </DIV>
  </DIV>
  <DIV id=e02>
    <DIV id=e021>
      ...
    </DIV>
    <DIV id=e022>
      ...
    </DIV>
  </DIV>
</BODY>
```



Блоки (DIV) – рамки и отступы




```
#qq {
  padding: 5px 10px;
  border: 1px solid green;
  margin: 5px 15px 5px 10px;
}
```

CSS свойства границ

border	Позволяет установить все свойства границ за одно определение.	CSS1
border-color	Позволяет определить цвет для всех границ элемента за одно определение.	CSS1
border-style	Позволяет определить стиль для всех границ элемента за одно определение.	CSS1
border-width	Позволяет установить ширину всех границ элемента за одно определение.	CSS1
border-radius `#el1 { border-radius:5px; }`	Позволяет определить форму всех углов элемента за одно определение.	CSS3

ПРИМЕР:

primer.htm

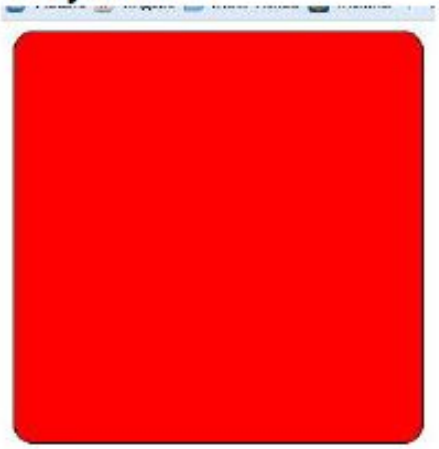
primer1.htm

Пример

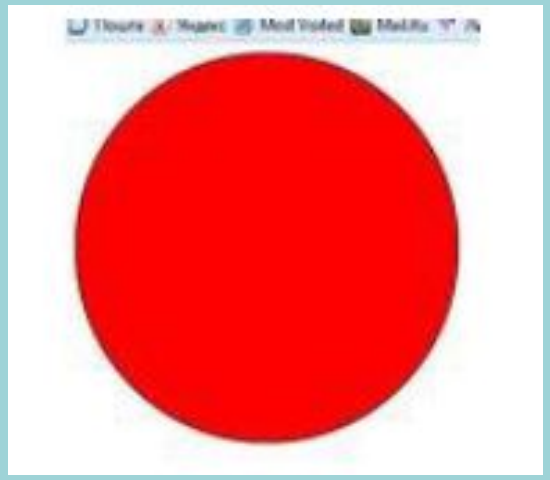
html	css

<html >
<head>
<link rel="stylesheet" href="css/radius.css"
type="text/css" />
<title>Радиус</title>
</head>
<body>
 	
</body>
</html>
	#radius {
	border: 1px solid #000000;
	width: 300px;
	height: 300px;
	background-color: red;
	border-radius: 15px;
	}

Результат:



Задание 1.
Из квадрата сделать круг.



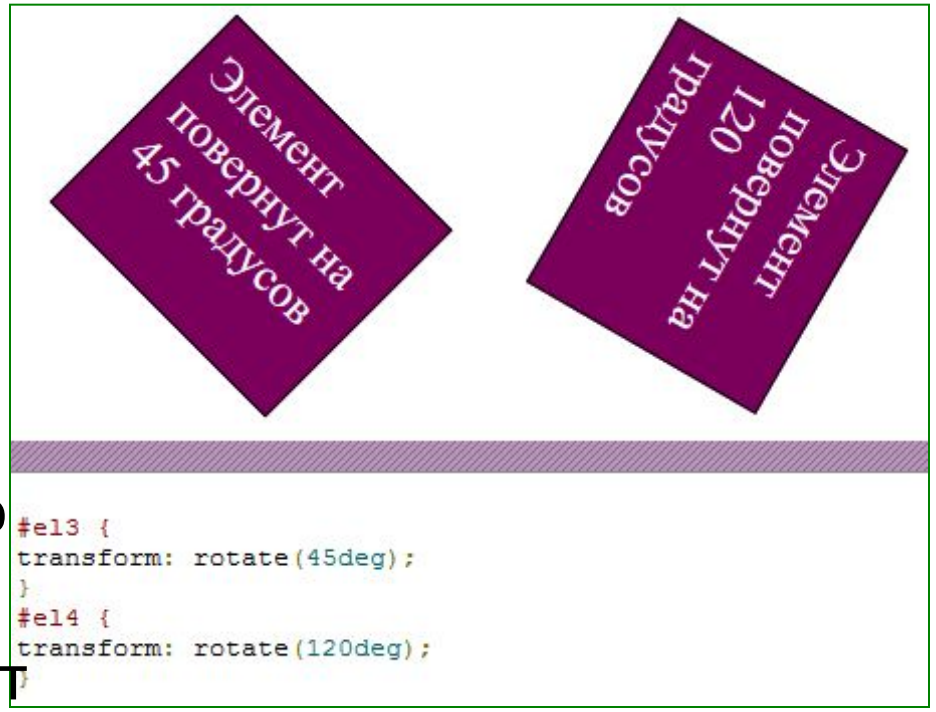
CSS3 трансформирование

translate(x,y) - сместить элемент на указанное количество пикселей по горизонтали и вертикали.

rotate(градусы) -повернуть элемент на указанное количество градусов (deg) по часовой стрелке.

scale(x,y) -растянуть элемент в ширину или высоту.

skew(x,y) -скосить элемент на указанное количество градусов по горизонтали и вертикали.



CSS3 функции трансформирования

Функция	Описание
translate(x,y)	Смещает элемент от изначальной позиции по горизонтали и вертикали.
translateX(x)	Смещает элемент по горизонтали.
translateY(y)	Смещает элемент по вертикали.
scale(x,y)	Растягивает элемент по вертикали и горизонтали.
scaleX(x)	Растягивает элемент по горизонтали.
scaleY(y)	Растягивает элемент по вертикали.
rotate(градусы)	Поворачивает элемент по часовой стрелке.
skew(x,y)	Скашивает элемент по горизонтали и вертикали.
skewX(x)	Скашивает элемент по горизонтали.
skewY(y)	Скашивает элемент по вертикали.
matrix(x,x,x,x,x,x)	Совмещает все перечисленные выше методы в один.

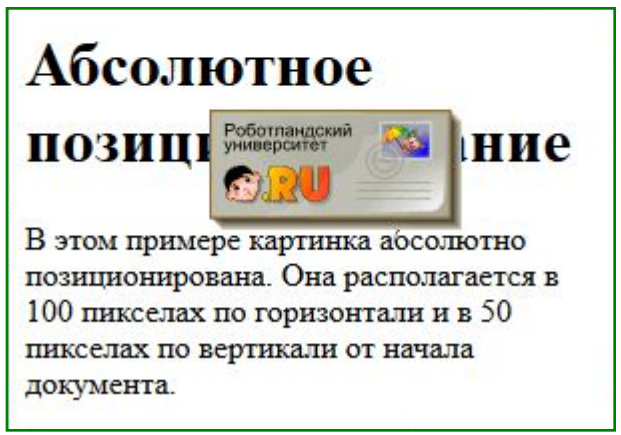
Задание 2.
Выполните трансформацию квадрата и круга 4 разными способами.

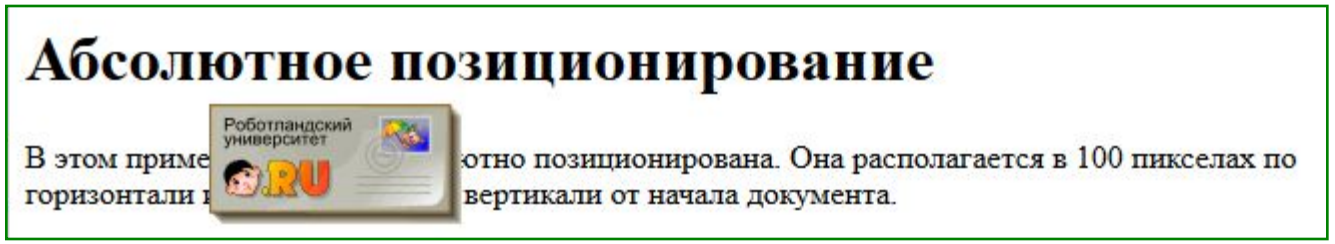
Абсолютное позиционирование

Абсолютное позиционирование задается стилевым указанием position:absolute.
При этом начало координат элемента находится в верхнем левом углу прямого предка, при условии, что он также позиционирован (абсолютно или относительно). Если родитель не позиционирован, то берется его родитель.
Если все предки не имеют указаний position, то в качестве точки отсчета принимается левый верхний угол экранного образа тега BODY, то есть, левый верхний угол документа.
Горизонтальная и вертикальная координата задаются свойствами left и top соответственно.

Пример, в котором картинка позиционируется в точку (100,50). Началом координат браузер считает начало документа.

```
<IMG  src=./pic/let.gif width=126 height=60 border=0
      alt="Роботландский университет"
      style="position:absolute; left:100px; top:50px;">
```





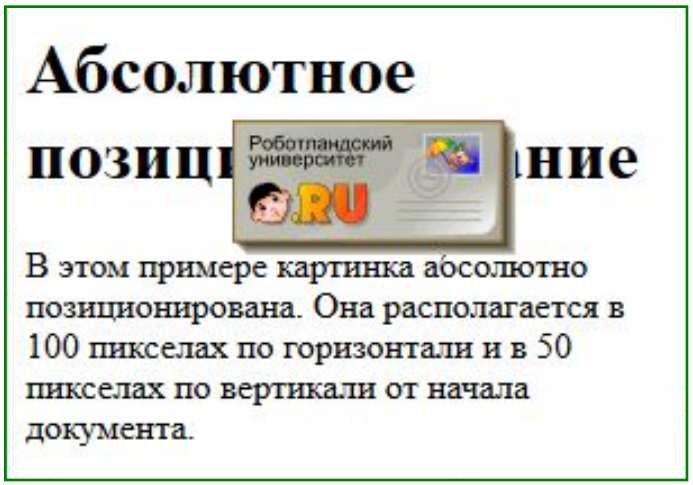
Приведенные выше примеры наглядно показывают, что абсолютно позиционированные элементы выпадают из процесса обычного последовательного форматирования.

Браузер не берет в расчет порядок следования кодов, а учитывает только вложенность для определения начала координат. Элементы выводятся в указанном месте, перекрывая при этом другие элементы страницы.

Порядок следования кодов абсолютно позиционируемых элементов становится важным, если они начинают перекрывать друг друга: “выше” оказывается тот элемент, код которого идет позже.

1 уровень

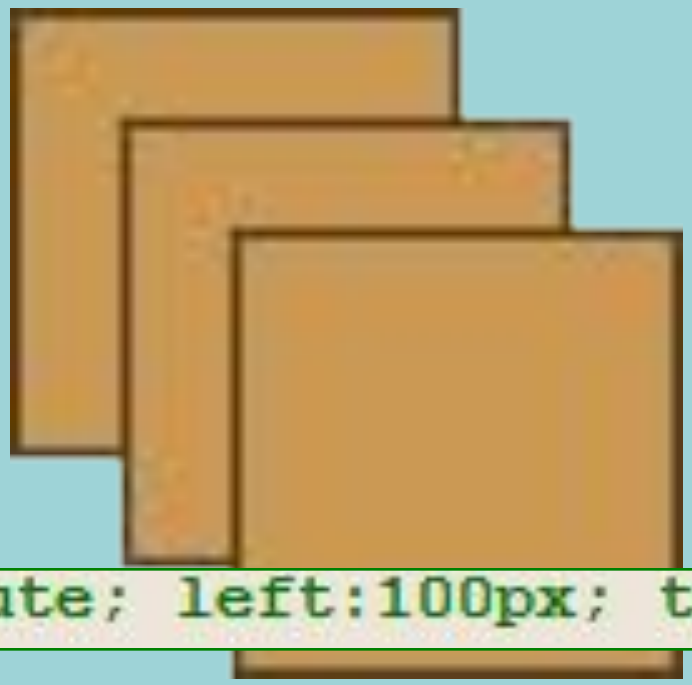
```
<BODY bgcolor=white text=black>

 <H1>Абсолютное позиционирование</H1>

 <P>
В этом примере картинка абсолютно позиционирована.
Она располагается в 100 пикселах по горизонтали и в 50
пикселах по вертикали от начала документа.

 <P>
Измените размеры окна и убедитесь, что картинка всегда
остается на прежнем месте.

 <IMG  src=./pic/let.gif width=126 height=60 border=0
       alt="Роботландский университет"
       style="position:absolute; left:100px; top:50px;">

</BODY>
```



2 уровень

Задание 3.

Создайте страницу, в которой одна и та же картинка выводится несколько раз со смещением вниз и вправо так, чтобы каждая следующая копия была выше предыдущей.



```
position:absolute; left:100px; top:50px;
```

ЭФФЕКТЫ ДИНАМИЧЕСКОГО ПЕРЕХОДЫ

```
#wrap1
{
border:1px #000 solid;
width:200px;
padding:10px;
font-size:1.5em;
transition: width 4s;
}
#wrap1:hover
{
width:500px;
}
```

transition -эффекты перехода.
Для создания переходов необходимо указать какое CSS свойство будет изменятся и скорость выполнения этих изменений в секундах.

transition: width 4s;

Для того, чтобы добавить эффект перехода к нескольким свойствам просто перечислите их названия через запятую.

transition: color 4s, width 4s, background-color 4s;

ЭФФЕКТЫ ДИНАМИЧЕСКОГО ПЕРЕХОДЫ

CSS3 свойства переходов

Свойство	Описание
transition	Позволяет задать значения четырех различных свойств перехода в одном определении.
transition-property	Позволяет указать имя CSS свойства, к которому будет применен эффект перехода.
transition-duration	Позволяет указать время выполнения перехода (*по умолчанию имеет значение 0*).
transition-timing-function	Позволяет задать функцию смягчения отвечающую за плавность выполнения перехода (*по умолчанию имеет значение 'ease'*).
transition-delay	Позволяет задать задержку перед началом выполнения перехода (*по умолчанию имеет значение 0*).

Z-индекс

Стилевое свойство **z-index** позволяет отойти от плоского представления документа на экране.

Его значением является целое число, номер плоскости, в которой будет размещаться элемент (вместе со своими потомками).

Основной текст имеет нулевой уровень **(z-index:0).**

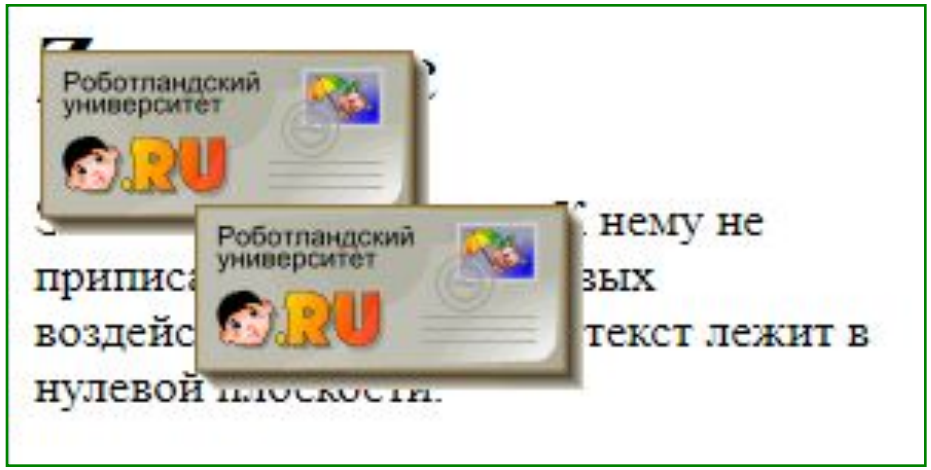
Положительный **z-индекс** размещает элементы над основным текстом, отрицательный — под ним.

Из двух плоскостей размещения та расположена выше, у которой **z-индекс** больше.

Если у элементов один уровень (задан явно или не задан вовсе), то тот из них будет располагаться выше, чей **HTML-код** идет позже.

Пример

```
<BODY bgcolor=white text=black>

 <H1>Z-индекс</H1>

 <P>
Этот текст – основной. К нему не приписано никаких
стилевых воздействий. Значит, этот текст лежит в нулевой
плоскости.

 <IMG src=./pic/let.gif width=126 height=60 border=0
      alt="Университет" align=left hspace=10
      style="position:absolute; left:0px; top:20px;
             z-index=-1;">

 <IMG src=./pic/let.gif width=126 height=60 border=0
      alt="Университет" align=left hspace=10
      style="position:absolute; left:50px; top:70px;
             z-index=-2;">

</BODY>
```



Задание 4.

Создайте структуру страницы. Размеры и позиции подберите приблизительно!

Пример кода:

Файл CSS

Блок «родитель»

Блоки «внутренние»

```
<html>
<head>
   <title>Блочная вёрстка</title>
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<div id="container">
      <div id="header">
      <h2>header (шапка сайта)</h2>
      </div>

      <div id="navigation">
      <h2>Блок навигации</h2>
      </div>

      <div id="sidebar">
      <h2>Левая панель</h2>
      </div>

      <div id="content">
      <h2>Основной контент страницы</h2>
      </div>

      <div id="clear">

      </div>

      <div id="footer">
      <h2>footer (низ сайта)</h2>
      </div>
</div>
</body>
</html>
```

Файл style.css

```
body {
	background: #FFF;
	color: #000;
	font-family: Arial, sans-serif;
	font-size: 14px;
}

#header {
	background: #F5DEB3;
	width: 100%;
	height: 55px;
}

#container {
	background: #FFD700;
	margin: auto auto;
	text-align: center;
	width: 80%;
	height: 400px;
}

#navigation {
	background: #FE9798;
	width: 100%;
	height: 25px;
}

#sidebar {
	background: #40E0D0;
	float: left;
	width: 20%;
	height: 280px;
}

#content {
	background: #DCDCDC;
	float: right;
	width: 80%;
```

«Плавающие» блоки и картинки

Если картинку поместить в абзац, то она ведет себя как символ, перемещаясь по экрану за своим местом (при изменении размера окна браузера).

Плавающая картинка

Этот роботландский лисенок помещен внутрь абзаца вот здесь:  . При изменении окна, картинка перемещается по экрану так, чтобы всегда отслеживать свое место в абзаце.

Плавающая картинка

Этот роботландский лисенок помещен внутрь абзаца вот здесь:  . При изменении окна, картинка перемещается по экрану так, чтобы всегда отслеживать свое место в абзаце.

```
<P>
Этот роботландский лисенок помещен внутрь абзаца вот здесь:
<IMG src=./pic/person1.gif width=44 height=76 border=0  alt="Роботландский лисенок">.
При изменении окна, картинка перемещается по экрану так, чтобы всегда отслеживать свое место в абзаце.
```

«Плавающий» блок с динамическим переходом

```
#wrap1:hover
{
color:#FFFFFF;
width:800px;
background-color:#880045;
top:55%;
left:55%;
}
```

«Плавающие» блоки

```
<div class="picture">
  <img src="dog.jpg">
  <p>На природе</p>
</div>
```


```
.picture {
  float: left;
  margin: 5px;
}
.picture p {
  margin: 0;
  text-align: center;
  font-family: sans-serif;
  font-size: 80%;
  font-weight: bold;
}
```

свойства блока

свойства абзаца внутри блока

Задание 5.

Создайте структуру страницы из примера. Размеры и позиции подберите приблизительно!

Замена рисунка при движении мыши

События:

onMouseOver – курсор мыши над объектом

onMouseOut – курсор мыши ушел с объекта

начальный рисунок

когда курсор мыши над рисунком

```
<IMG SRC="image1.gif"
     onMouseOver="this.src='image2.gif'"
     onMouseOut="this.src='image1.gif'">
```

после ухода мыши

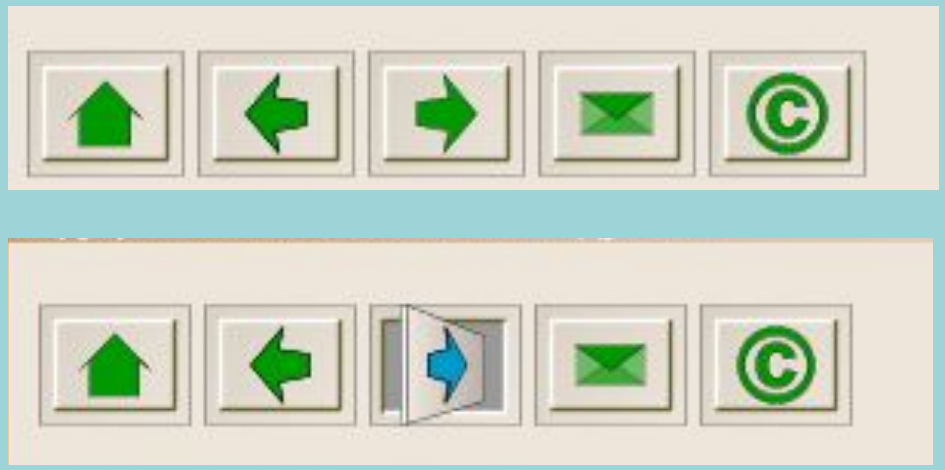
this – этот объект

this.src – свойство SRC этого объекта

image1.gif и image2.gif – две картинки

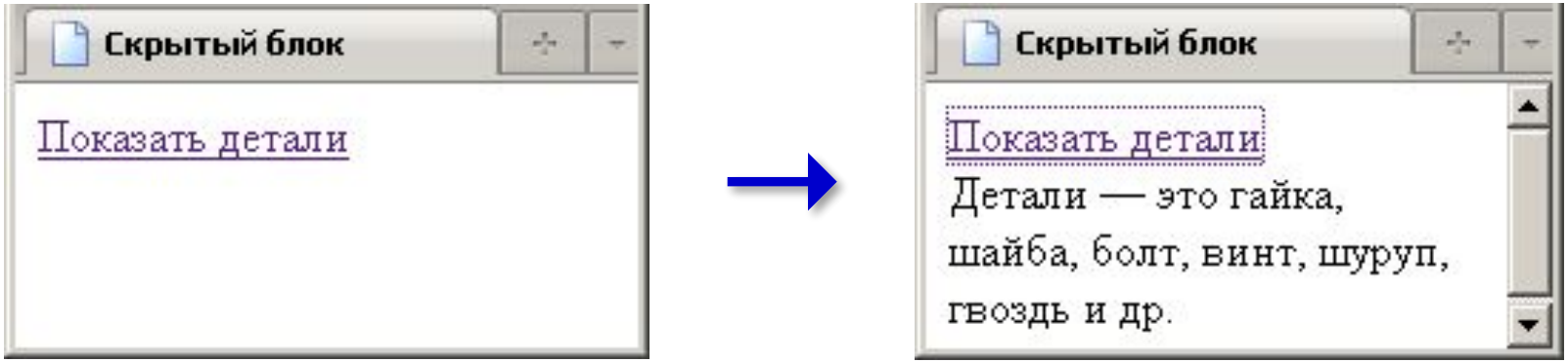
Задание 6.

Создайте в своем документе эффект открытия/закрытия объекта.



Нестандартное применение блоков

Скрытый блок

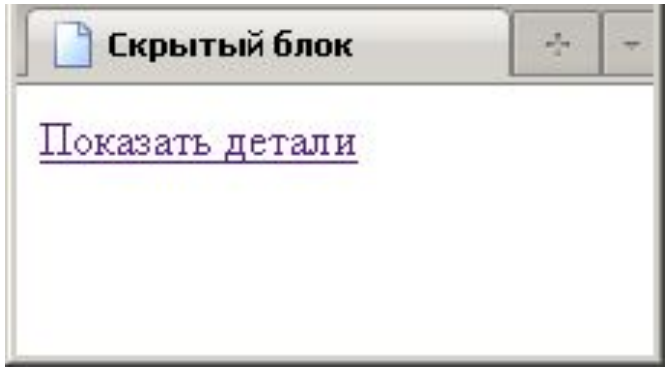



```
<div id="details" class="hidden">
Детали &#151; это гайка, шайба, болт,
винт, шуруп, гвоздь и др.
</div>
```

```
.hidden {
 display:none;
}
```

Скрытый блок: оформление ссылки

остаться на странице




```
<a href="#"
   onClick="show('details');return false;">
Показать детали
</a>
```

по щелчку вызвать функцию **show**

переход не выполнять

Скрытый блок: как его открыть?

```
function show ( name )
{
  var elem = document.getElementById(name);
  if ( elem )
    elem.style.display = "block";
}
```

найти блок по имени

изменить свойство **display**

test.js

```
<head>
  <script type="text/javascript"
          src="test.js">
  </script>
</head>
```

left	задает положение относительно левого края контейнера.
top	задает положение относительно верхнего края контейнера. Задаются в процентах или пикселях.
z-index	указывает на то, какой элемент должен располагаться выше при перекрытии. Измеряется в единицах.
visibility	определяет видимость элемента. Принимает значения visible (видимый), hidden (скрытый), inherit (наследуется от родительского элемента).
margin	устанавливает величину отступа от каждого края элемента. Отступом является пространство от границы текущего элемента до внутренней границы его родительского элемента. Разрешается использовать одно, два, три или четыре значения, разделяя их между собой пробелом.