3.12. Практическая работа №12. Динамический HTML. Управление текстом

Цель: Научиться с помощью свойств и методов DHTML управлять текстом, а также получить доступ к DOM-модели с помощью языка JavaScript.

Рассмотрим, каким образом можно получить доступ к DOM-модели с помощью яыка JavaScript.

Существует три способа реализовать эту задачу:

1. Использовать метод **getElementsByTagName (),** возвращающий

список узлов с указанным именем тега.

2. С помощью метода **getElementByld ()**, которому передается уникальный атрибут id необходимого элемента.

3. Воспользоваться свойством **DocumentElement** объекта Document и использовать древовидную структуру DOM для доступа к необходимому элементу.

Доступ к узлам.

Работа с текстом немного отличается от работы с другими элементами DOM. Каждый фрагмент текста на странице помещается в невидимый узел #text. Приходится учитывать специфику реализации работы с DOM-моделью различных браузеров. Алгоритм в каждом случае одинаковый. Свойство documentElement документа ссылается на объект Element, который представляет сам документ. Находим его дочерний узел, соответствующий элементу BODY в переменной BodyNode. Отличия связаны с тем, что браузеры Opera и Mozilla Firefox учитывают пустые текстовые строки между элементами заголовков, введенные для удобства чтения. Кроме того, внесение изменений в документ также изменит объекты, что повлечет за собой изменение и в коде. Таким образом, этот способ является довольно трудоемким в реализации.

Рассмотрим пример доступа к узлам:

```
<html>
<script>
function v()
        {
        var h2l=document.getElementsByTagName('h2')
        var he1=h2l.item(0);
        alert(he1.firstChild.nodeValue+' '+he1.tagName)
        }
function v2()
        {
        var he2=document.getElementById('h2Tag2')
        alert(he2.firstChild.nodeValue+' '+he2.tagName)
        }
function v3()
        {
        if (navigator.appName=='Netscpae')
                {
                var
BodyNode=document.documentElement.childNodes[2]
                var he3=BodyNode.childNodes[3]
                alert(he3.firstChild.nodeValue+'
'+he3.tagName)
                }
        if (navigator.appName=='Opera')
```

```
                    {
                    var
BodyNode=document.documentElement.childNodes[1]
                    var he3=BodyNode.childNodes[3]
                    alert(he3.firstChild.nodeValue+'
'+he3.tagName)
                    }
               if    (navigator.appName=='Microsoft    Internet
Explorer')
                    {
                    var
BodyNode=document.documentElement.childNodes[1]
                    var he3=BodyNode.childNodes[2]
                    alert(he3.firstChild.nodeValue+'
'+he3.tagName)
                    }
               }
        </script>
        <body>
        <h2 id='h2Tag1'>Первый заголовок</h2>
        <h2 id='h2Tag2'>Второй заголовок</h2>
        <h2 id='h2Tag3'>Третий заголовок</h2>
        <form>
        <input type=button value='Показать1' onClick='v()'>
        <input type=button value='Показать2' onClick='v2()'>
        <input type=button value='Показать3' onClick='v3()'>
        </form>
        </body>
        </html>
```

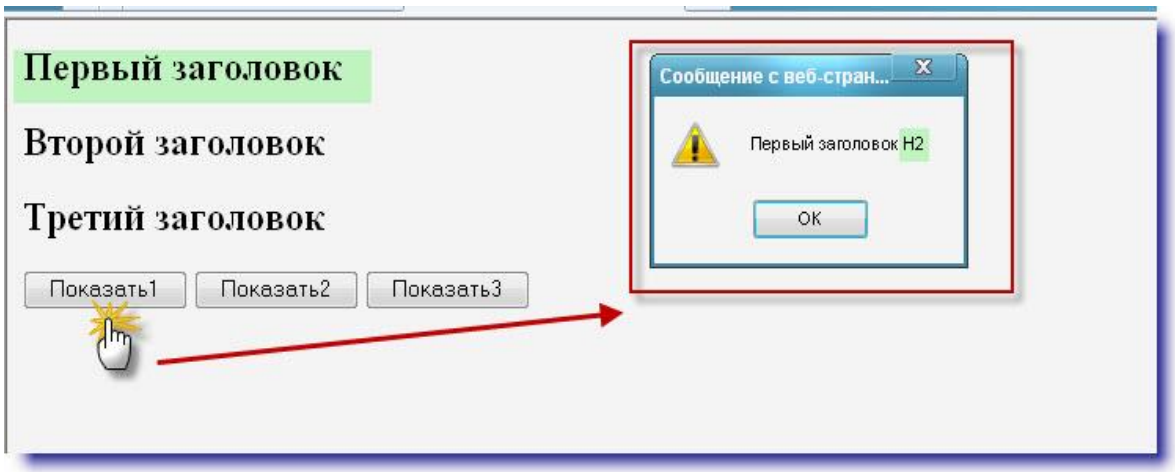

Рисунок 3.12.1. Доступ к узлам.

Доступ к атрибутам.

Атрибут элемента - это переменная и ее значение, которые указываются в теге элемента, то есть в угловых скобках (<>). Например, элемент img имеет атрибут src, указывающий на графический файл. Чтобы получить значение атрибута, необходимо воспользоваться методом getAttribute() объекта Element. Параметром данного метода должно выступать соответствующее имя атрибута элемента.

Рассмотрим пример доступа к атрибутам:

```
<html>
<body>
<h2 id='h2Tag' align='center'>Заголовок</h2>
<form>
Текст
<input type='text' id='text1' ><br>
<input type=submit id='But'>
</form>
<br>
<script>
var e=document.getElementById('h2Tag')
document.write('Элемент '+e.tagName+', id=')
document.write(e.getAttribute('id')+', выравнивание -
'+e.getAttribute('align'))
document.write('<br>')
Display(document.getElementById('text1'))
Display(document.getElementById('But'))
function Display(el)
    {
    document.write('Элемент '+el.tagName+', id=')

    document.write(el.getAttribute('id')+', тип -
'+el.getAttribute('type'))
    document.write('<br>')
    }
</script>
</body>
</html>
```

Рисунок 3.12.2. Доступ к атрибутам.

Изменение текста.

На странице можно изменять не только текст, но и его атрибуты. Для этого необходимо воспользоваться методом setAttribute () объекта Element. Параметрами данного метода должны выступать имя атрибута элемента и соответствующее ему значение. Рассмотрим на примере:

```
<html>
<script>
function ch()
	{
	var e=document.getElementById('h2Tag')
	e.setAttribute('align','left')
	var e=document.getElementById('iTag')
	e.firstChild.nodeValue='Текст2'
	}
</script>
<body>
<h2 id='h2Tag' align='center'>Заголовок</h2>
<i id='iTag'>Текст</i>
<form>
<input type=button id='But' value='Изменить'
onClick='ch()'>
</form>
</body>
</html>
```

Рисунок 3.12.3. Изменение текста до запуска сценария.

На данной странице нажатие кнопки вызывает функцию **Change()**, которая изменяет текст. С помощью метода **setAttribute()** назначается выравнивание по левой стороне для заголовка. При обращении же к дочернему узлу элемента I изменяется текст, выделенный курсивом.

Рисунок 3.12.4. Изменение текста после запуска сценария.

3.13. Практическая работа №13. Графика на веб-страницах

Цель: Научиться с помощью объекта Image и событий составлять сценарии в JavaScript.

Программирование графики в JavaScript опирается на объект Image. Наиболее часто используемой возможностью визуальных эффектов является использование событий.

Всего для визуальных эффектов накатов (rollover) используется четыре типа событий. Обработчики событий активизируются при следующих действиях:

1. onMouseDown — щелчок кнопкой мыши на элементе;
2. onMouseUp — отпускание кнопки мыши после щелчка на элементе;
3. onMouseOver — наведение указателя мыши на элемент;
4. onMouseOut — перемещение указателя мыши с элемента.

Рассмотрим пример демонстрации событий изображения, для этого необходимо найти 4 изображения:

```
<html>
<head>
<title> События</title>
</head>
<body>
<center>
<h2>Демонстрация событий изображения</h2>
<a href="javascript:void(0)"
onMouseOver="document.pic.src='1.jpeg'"
onMouseDown="document.pic.src='2.jpeg'"
onMouseUp="document.pic.src='3.jpeg'"
onMouseOut="document.pic.src='4.jpeg'">
<img src="4.jpeg" border=0 name="pic">
</a>
</body>
</html>
```

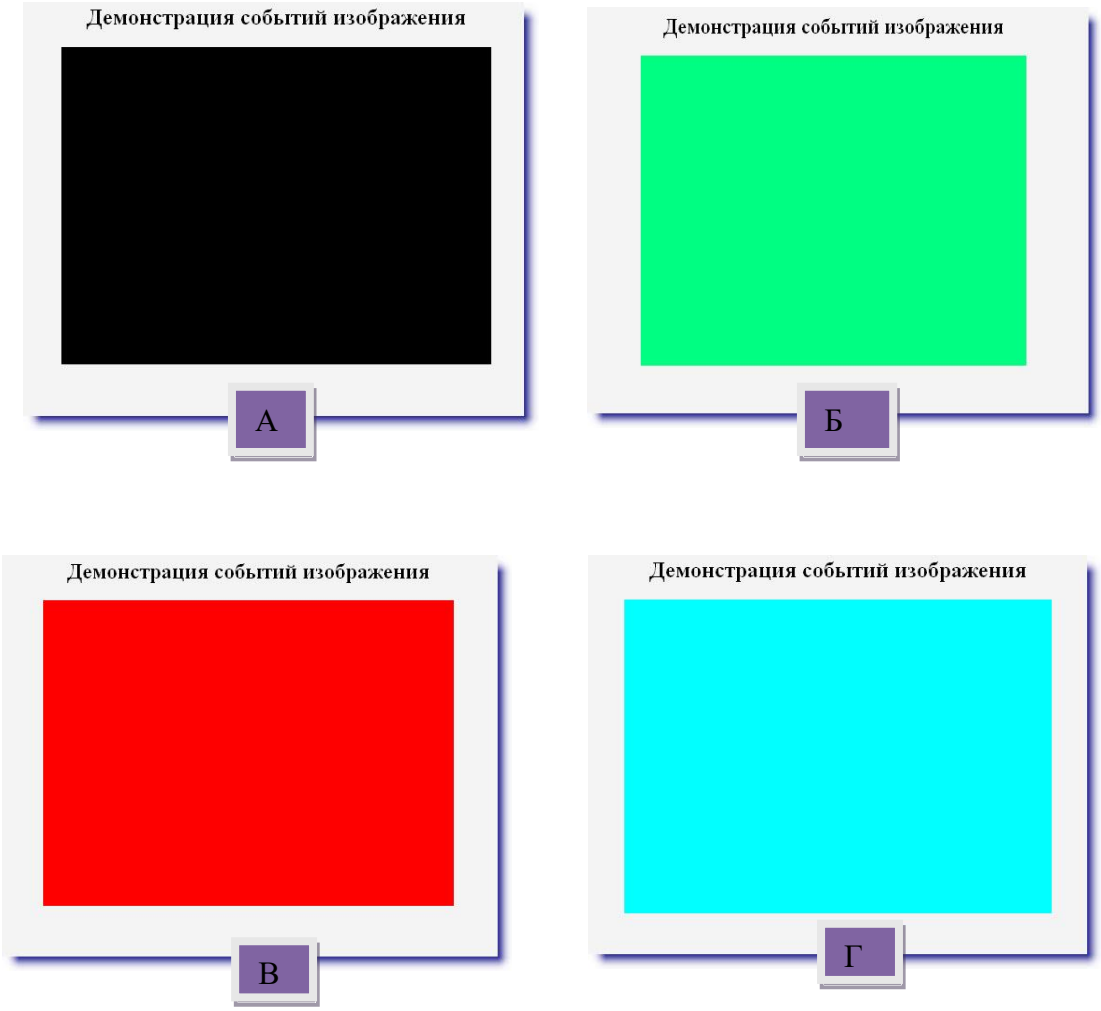

Рисунок 3.13.1. Демонстрация событий: А- onMouseOut, Б- onMouseUp, В-onMouseDown, Г-onMouseOver.

Рассмотрим пример по смене изображений:

```
<html>
<head>
<title>Смена изображений</title>
<script>
numerals=new Array(6);
letters=new Array(6);
for(var i=1;i<6;i++)
{
numerals[i] =new Image();
letters[i] =new Image();
numerals[i].src ="img"+i+".jpeg";
letters[i].src ="pic"+i+".jpeg";
}
function imageOut(img)
{
document.images[img-l].src=numerals[img].src;
}
function imageOver(img)
{
document. images[img-1].src=letters[img].src;
}
</script>
</head>
<body>
<h2>Смена изображений</h2>
<a href="javascript:void(0)"
onMouseOver="imageOver(0)"
onMouseOut="imageOut(0)">
<img src="0.jpeg" width=81 height=81>
</a>
<a href="javascript:void(0)"
onMouseOver="imageOver(1)"
onMouseOut="imageOut(1)">
<img src="1.jpeg" width=81 height=81>
</a>
<a href="javascript:void(0)"
onMouseOver="imageOver(2)"
onMouseOut="imageOut(2)">
<img src="2.jpeg" width=81 height=81>
</a>
<a href="javascript:void(0)"
onMouseOver="imageOver(3)"
onMouseOut="imageOut(3)">
```

```
<img src="3.jpeg" width=81 height=81>
</a>
<a href="javascript:void(0)"
onMouseOver="imageOver(4)"
onMouseOut="imageOut(4)">
<img src="4.jpeg" width=81 height=81>
</a>
</body>
</html>
```

Рисунок 3.13.2. Смена изображений

3.14. Практическая работа №14. Графика на веб-страницах

Цель: Научиться создавать сценарии с использованием слоев их свойств в языке JavaScript.

В настоящий момент для создания слоев обычно используется элемент DIV, являющийся блочным элементом разметки. Этот элемент просто определяет данные, но не указывает браузеру, как их отображать. Вообще-то с помощью любого HTML-элемента и применения некоторых свойств каскадных таблиц стилей можно управлять позиционированием, отображением и наложением элементов.

Для управления слоями необходимо использовать таблицы стилей, с помощью которых можно реализовать все возможности манипулирования слоями.

Рассмотрим пример, демонстрирующий использование свойства позиционирование слоев:

```
<html>
<body>
<h2>Позиционирование слоев</h2>
Это текст,
<div id=f style='background-color:orange;'>
1 слой - статический
</div>
внутри которого находится слой
<div id=s
style='position:absolute;top:200px;left:10px;background-
color:green;'>
2 слой - с абсолютными координатами
</div>
<div id=t
style='position:absolute;top:300px;left:60px;background-
color:red;'>
3 слой - с абсолютными координатами
</div>
<div id=t
style='position:relative;top:250px;left:40px;background-
color:violet;'>
4 слой - с относительными координатами
</div>
</body>
</html>
```

Для наглядности каждому слою назначен фоновый цвет с помощью свойства background-color. Кроме того, для позиционирования задаются значения top и left.

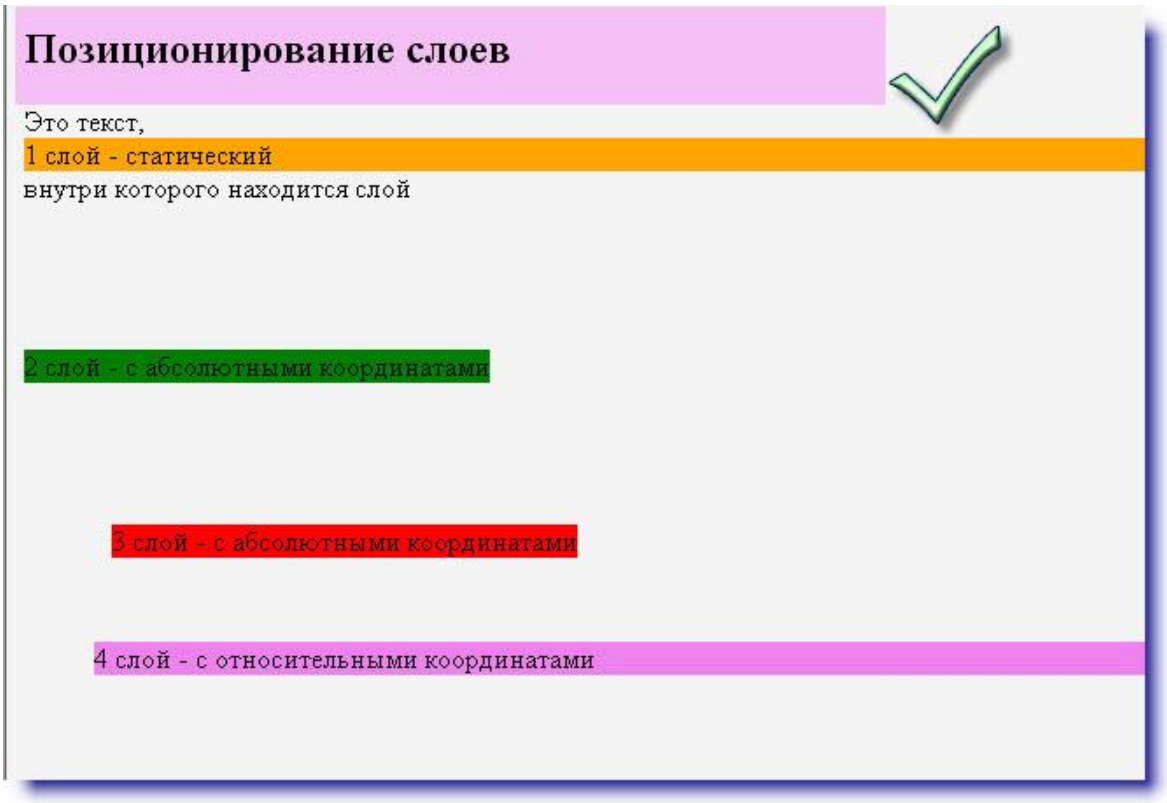

Рисунок 3.14.1. Позиционирование слоев.

Рассмотрим пример задания координат, высоты и ширины слоев:

```
<html>
<body>
<h2>Задание координат,высоты и ширины слоев</h2>

<div id=f
style='position:absolute;top:50px;left:50px;background-
color:orange;'>
1 слой
</div>

<div id=s
style='position:absolute;bottom:50px;left:50px;height:100px;widt
h:150px;background-color:green;'>
2 слой
</div>

<div id=t
style='position:absolute;bottom:50px;right:50px;height:50px;wid
```

```
th:30px;background-color:red;'>
        3 слой
        </div>

        <div id=fo
style='position:absolute;top:50px;right:50px;height:100px;width:
150px;background-color:blue;'>
        4 слой
        </div>

        <div id=fi
style='position:absolute;top:50%;left:50%;height:300px;width:30
0px;background-color:violet;'>
        5 слой
        </div>
        </body>
        </html>
```

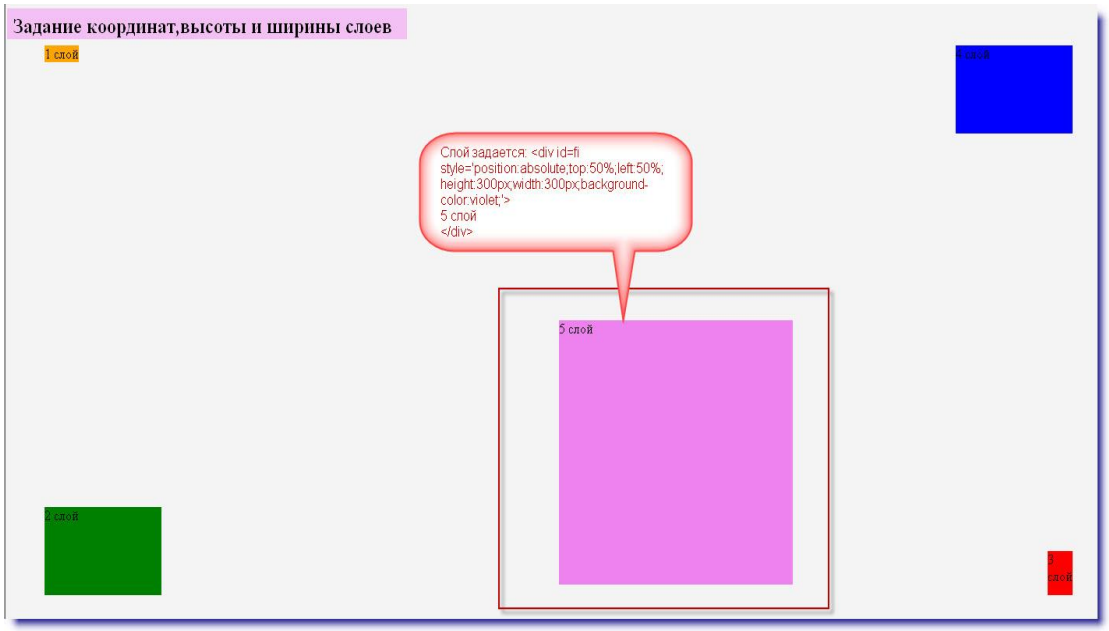

Рисунок 3.14.2. Задание координат, высоты и ширины слоев.