

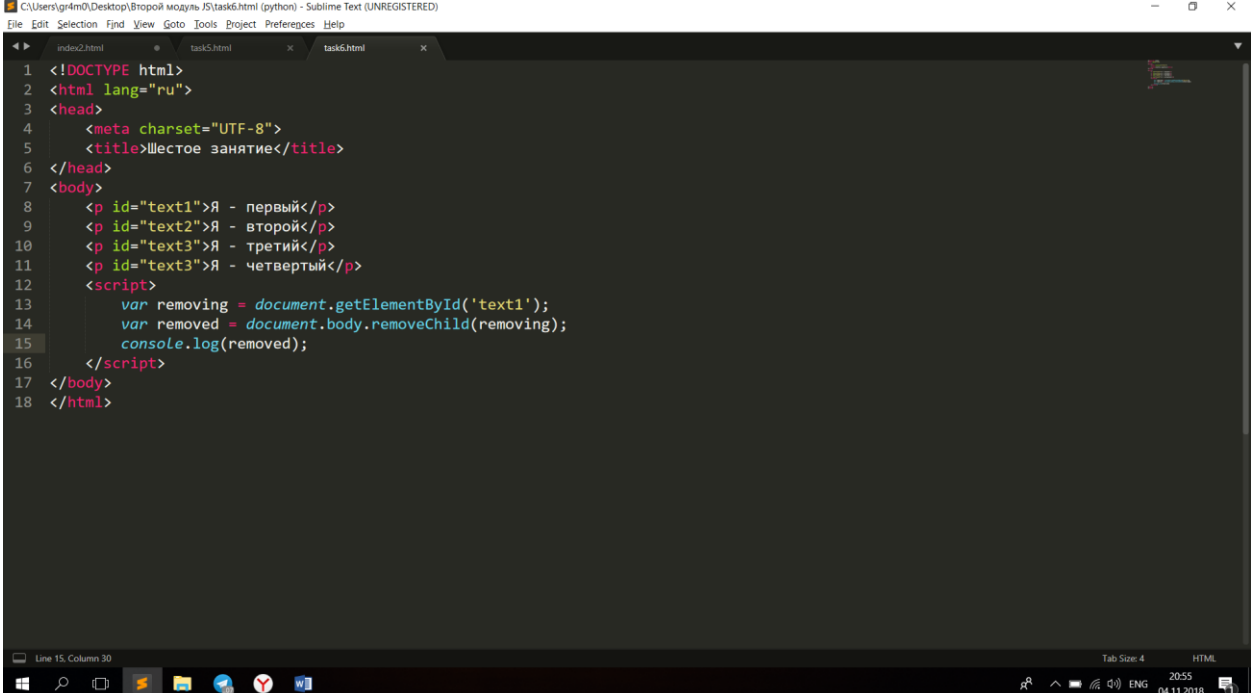
Программирование на JS

Второй модуль

Второе занятие

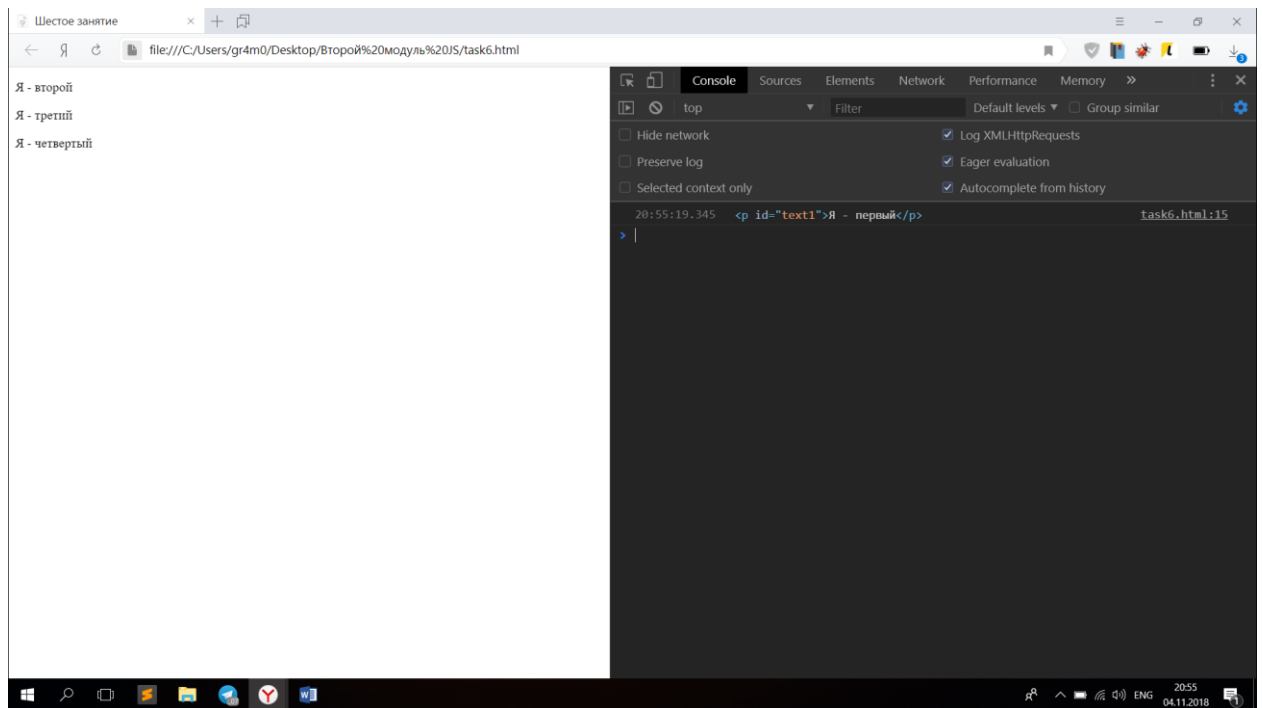
Продолжаем работать с моделью документа в браузере (DOM – Document Object Model ниже я буду использовать эту аббревиатуру для сокращения).

Кроме добавления элементов в DOM, мы можем их оттуда удалять с помощью метода `removeChild`. Метод `removeChild` возвращает значение удаленного узла DOM

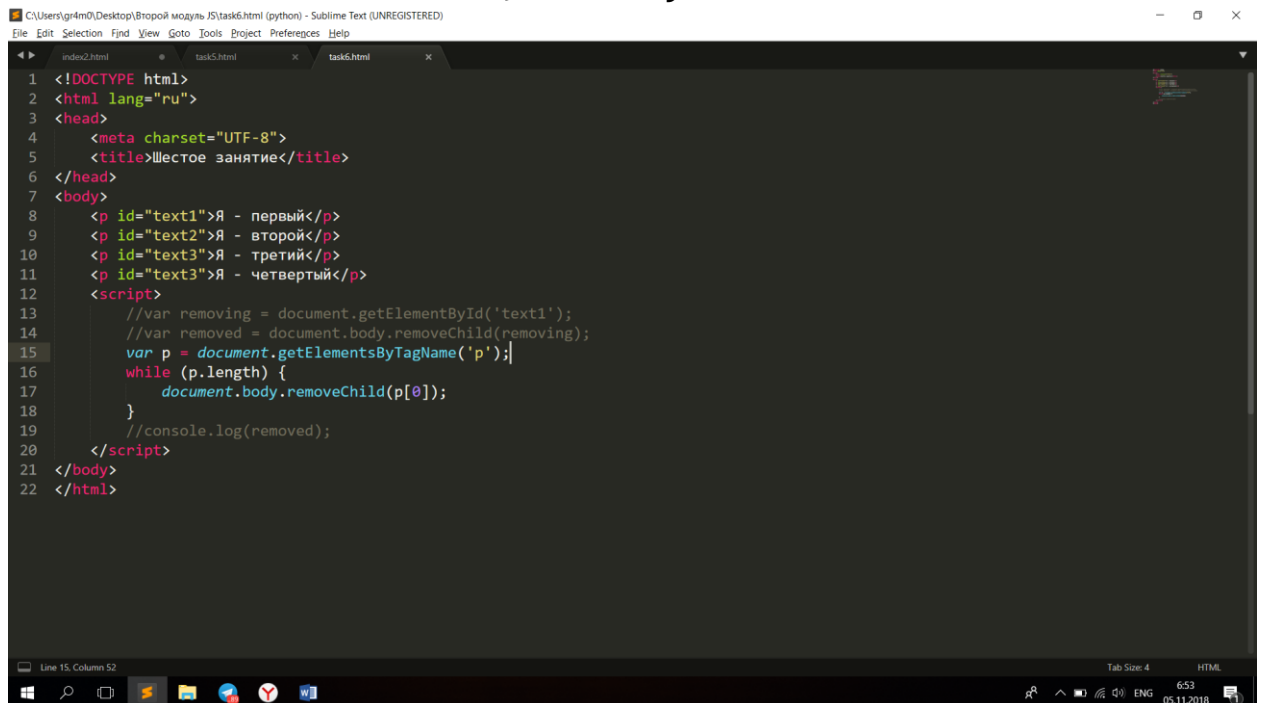


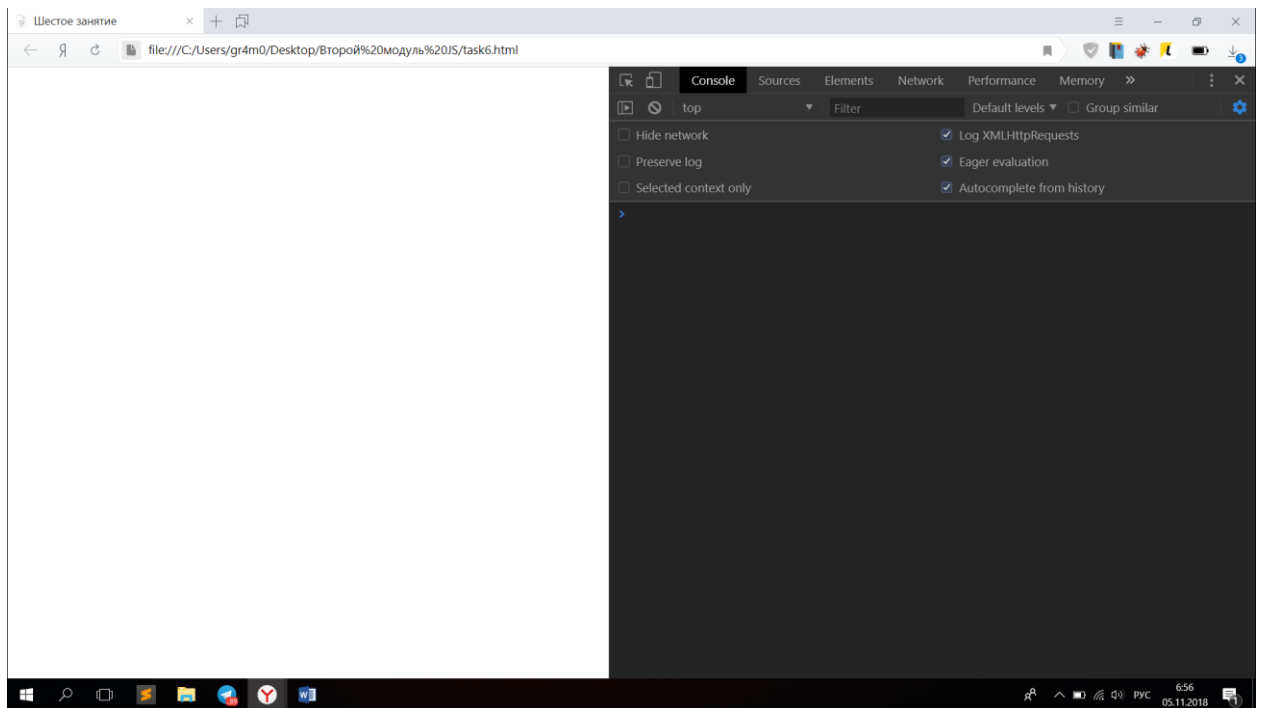
```
C:\Users\gr4m0\Desktop\Второй модуль JS\task6.html (python) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

1 <!DOCTYPE html>
2 <html lang="ru">
3 <head>
4   <meta charset="UTF-8">
5   <title>Шестое занятие</title>
6 </head>
7 <body>
8   <p id="text1">Я - первый</p>
9   <p id="text2">Я - второй</p>
10  <p id="text3">Я - третий</p>
11  <p id="text3">Я - четвертый</p>
12  <script>
13    var removing = document.getElementById('text1');
14    var removed = document.body.removeChild(removing);
15    console.log(removed);
16  </script>
17 </body>
18 </html>
```

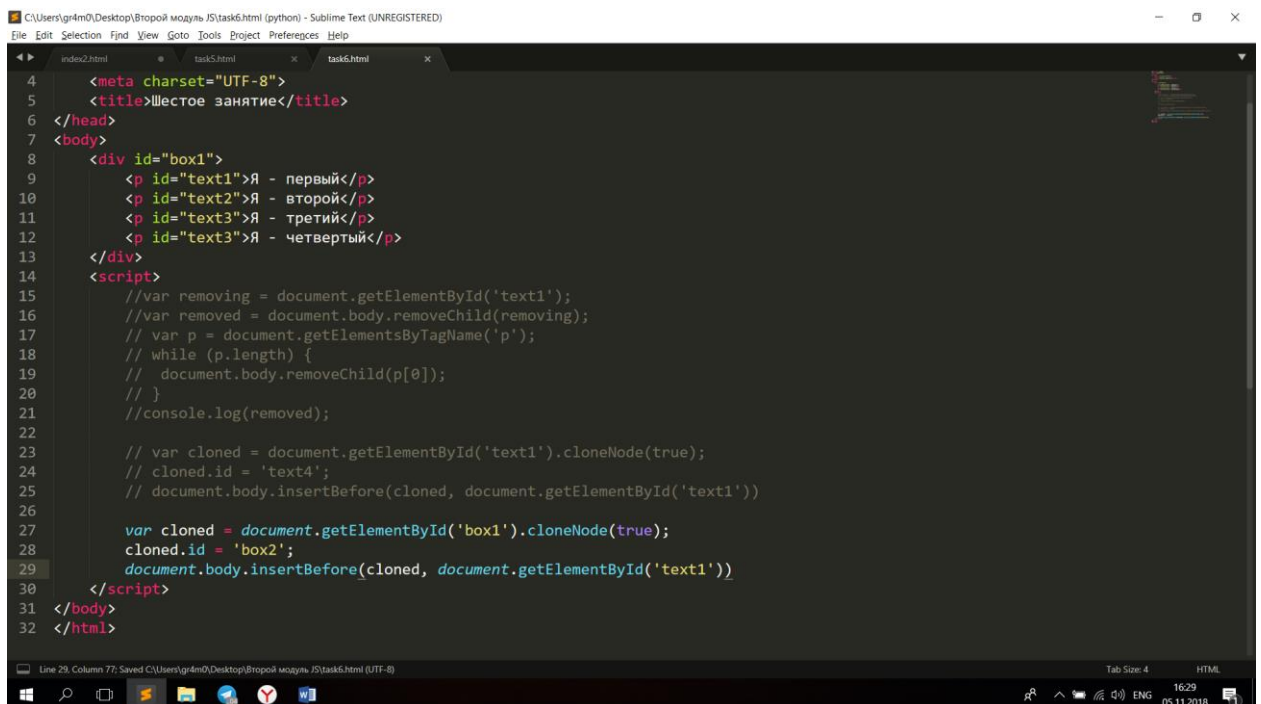


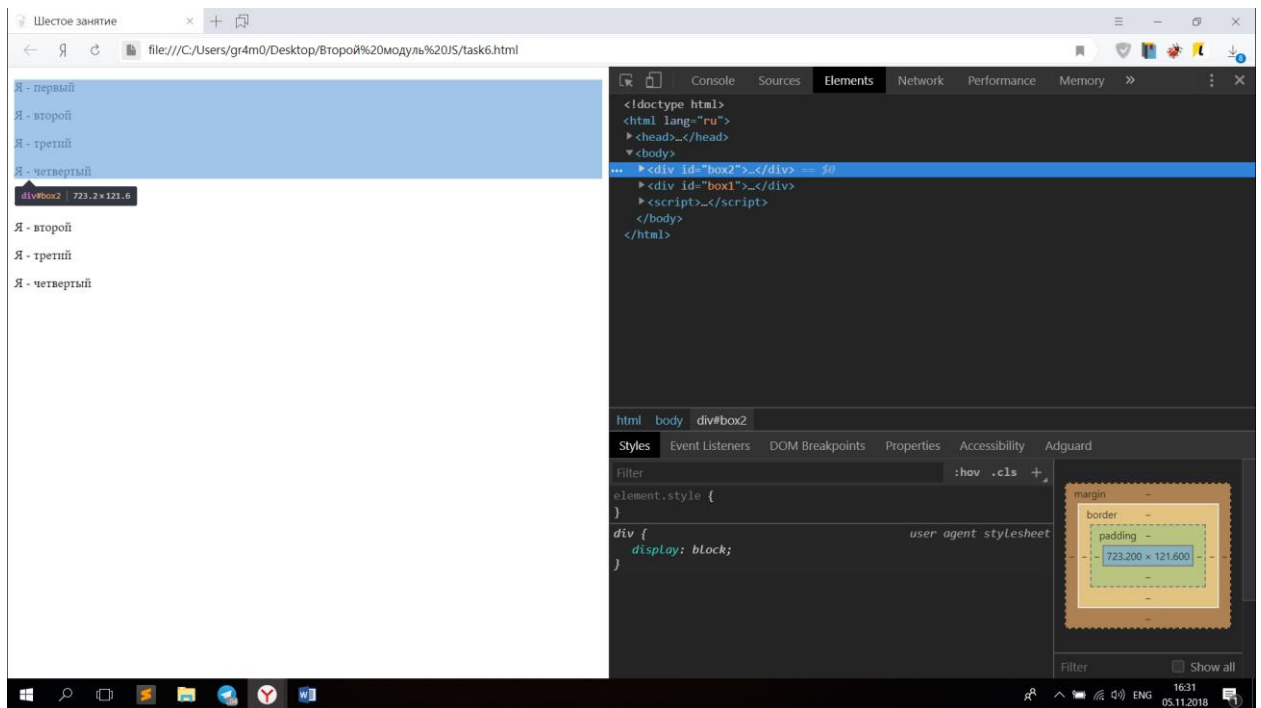
Для того чтобы удалить все параграфы, мы должны вызвать `removeChild` в цикле `while`. Этот цикл читается так: пока в коллекции `p` есть элементы, удаляем первый из них. Дело в том, что при удалении элемента индексы в коллекции автоматически сдвигаются, поэтому цикл `for` нам не подойдет.



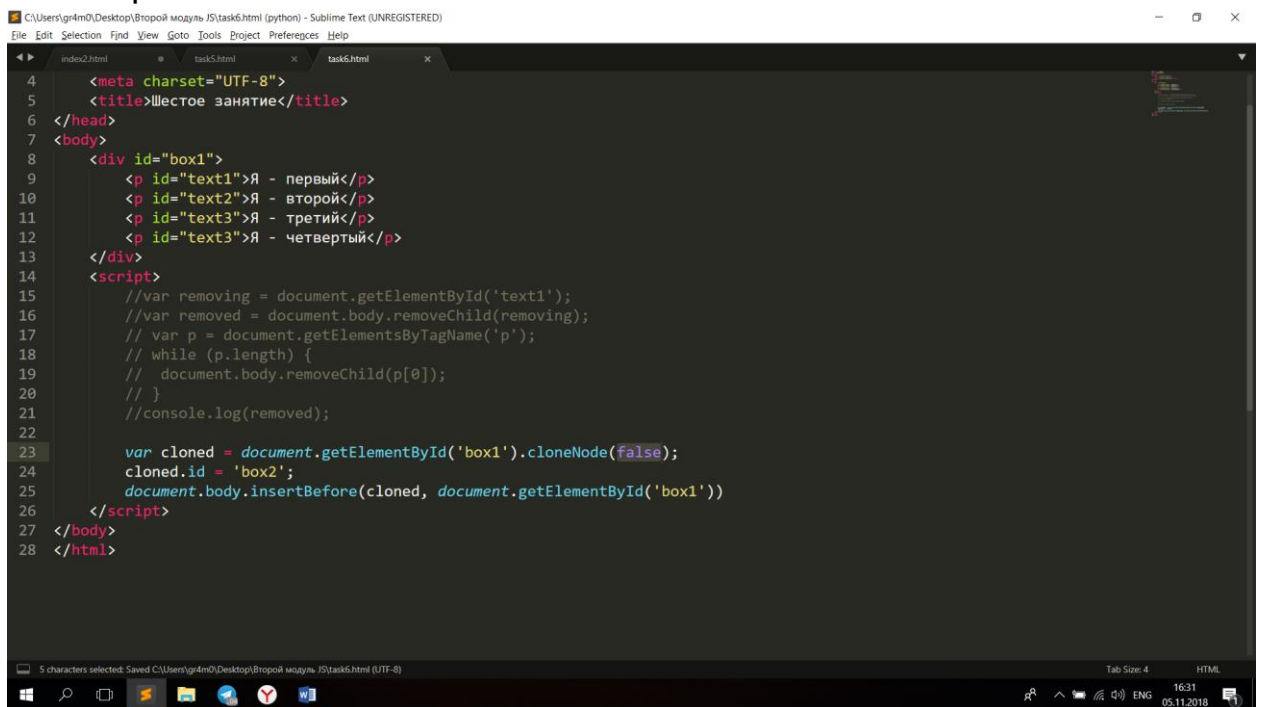


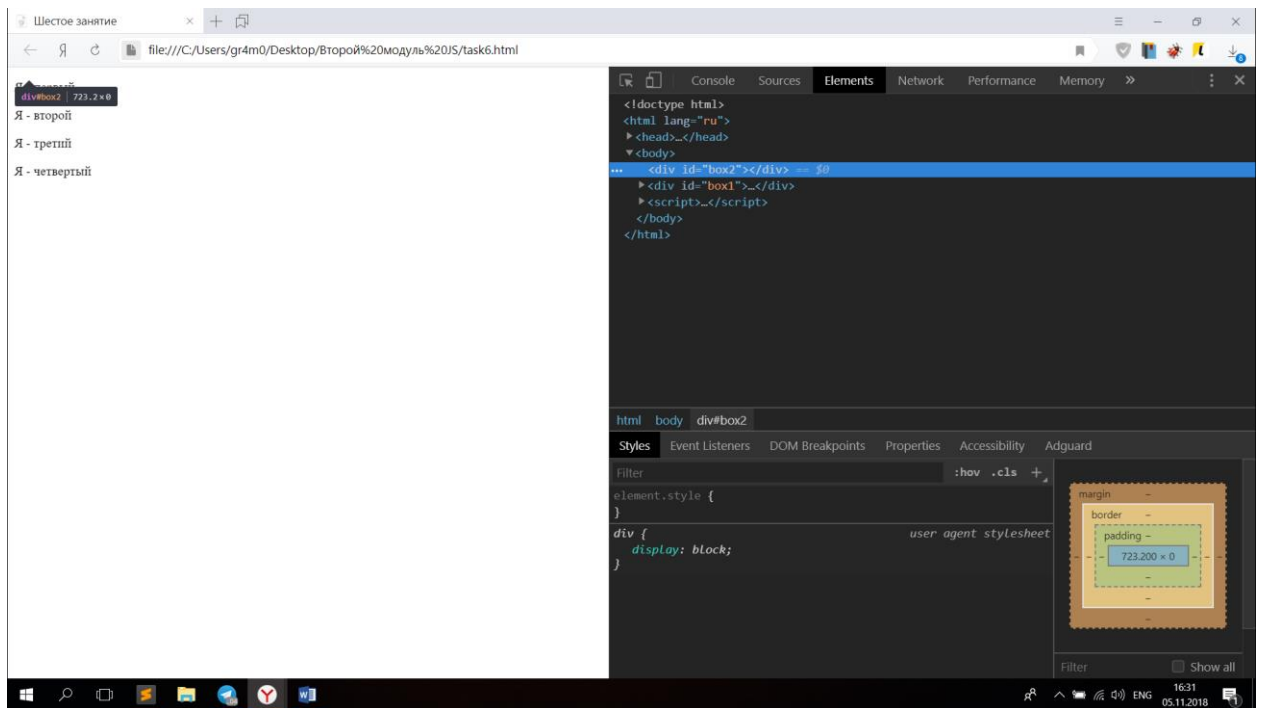
С помощью метода `cloneNode` мы можем копировать узлы DOM. Этот метод принимает один параметр – `true` или `false`, который определяет глубину копирования элемента: `true` – с содержимым, `false` – без. Обернем наши параграфы в `div`





А теперь с false





Во втором случае div скопировался без содержимого.

Рассмотренные методы вставки и удаления работают всегда и везде. Сравнительно недавно в стандарте появились методы, которые позволяют вставить что угодно и куда угодно.

Синтаксис:

- `node.append(...nodes)` – вставляет `nodes` в конец `node`,
- `node.prepend(...nodes)` – вставляет `nodes` в начало `node`,
- `node.after(...nodes)` – вставляет `nodes` после узла `node`,
- `node.before(...nodes)` – вставляет `nodes` перед узлом `node`,
- `node.replaceWith(...nodes)` – вставляет `nodes` вместо `node`.

Вот ссылка на поддержку браузерами данных методов:

<https://caniuse.com/#search=append>

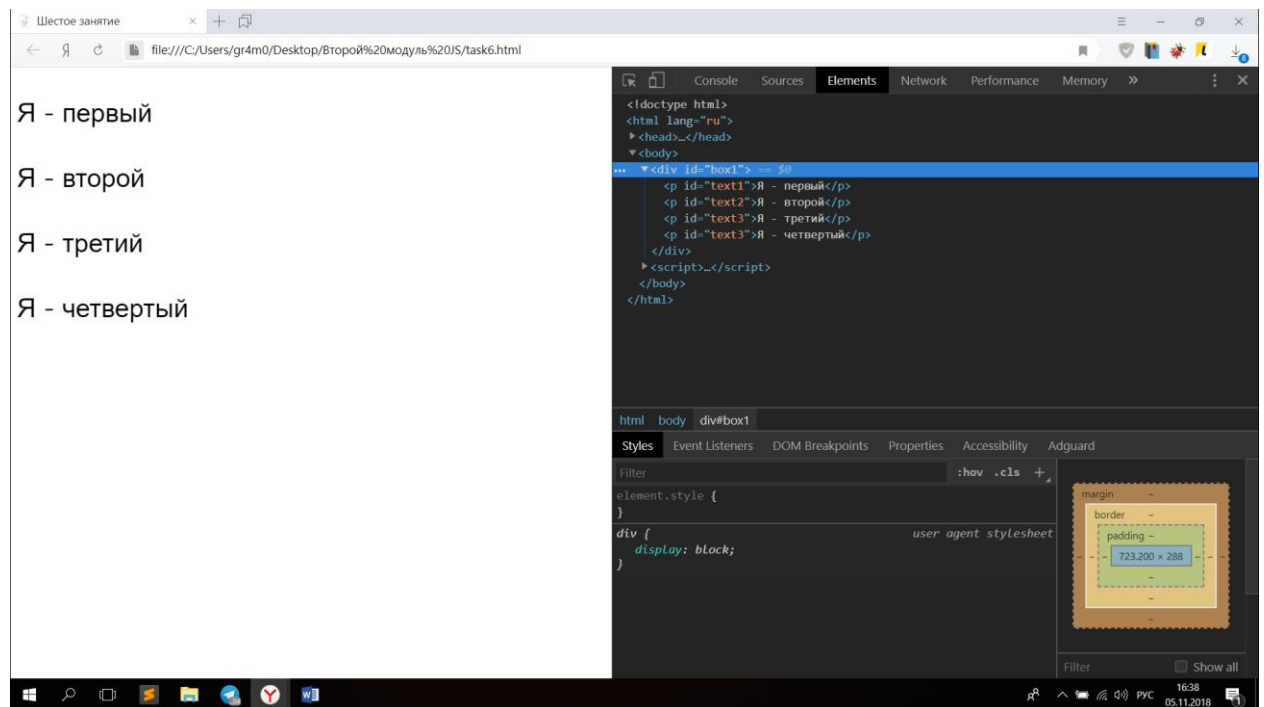
Рассмотрим их по очереди.

Я немного изменил стили документа

```

1 <!DOCTYPE html>
2 <html lang="ru">
3 <head>
4   <meta charset="UTF-8">
5   <title>Шестое занятие</title>
6   <link href="https://fonts.googleapis.com/css?family=M+PLUS+Rounded+1c" rel="stylesheet">
7   <style>
8     p {
9       font-size: 32px;
10      font-family: 'M PLUS Rounded 1c', sans-serif;
11      line-height: 1.5;
12    }
13  </style>
14 </head>
15 <body>
16   <div id="box1">
17     <p id="text1">Я - первый</p>
18     <p id="text2">Я - второй</p>
19     <p id="text3">Я - третий</p>
20     <p id="text3">Я - четвертый</p>
21   </div>
22   <script>
23     //var removing = document.getElementById('text1');
24     //var removed = document.body.removeChild(removing);
25     // var p = document.getElementsByTagName('p');
26     // while (p.length) {
27     //   document.body.removeChild(p[0]);
28     // }
29     //console.log(removed);
30   </script>

```



Проиллюстрируем работу этих методов

Метод append для вставки текста

```
C:\Users\gr4m0\Desktop\Второй модуль\JS\task6.html (python) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index2.html task5.html task6.html
16 <div id="box1">
17   <p id="text1">Я - первый</p>
18   <p id="text2">Я - второй</p>
19   <p id="text3">Я - третий</p>
20   <p id="text3">Я - четвертый</p>
21 </div>
22 <script>
23   //var removing = document.getElementById('text1');
24   //var removed = document.body.removeChild(removing);
25   // var p = document.getElementsByTagName('p');
26   // while (p.length) {
27   //   document.body.removeChild(p[0]);
28   // }
29   //console.log(removed);
30
31   // var cloned = document.getElementById('box1').cloneNode(false);
32   // cloned.id = 'box2';
33   // document.body.insertBefore(cloned, document.getElementById('box1'))
34
35   var p = document.createElement('p'),
36       beforeP = document.getElementById('text1');
37   p.append('Я добавленный текст с помощью метода append!');
38   p.id = 'before';
39   p.style.color = 'red';
40   beforeP.before(p);
41 </script>
42 </body>
43 </html>

Line 40, Column 27 Tab Size: 4 HTML 16:46 05.11.2018
```

Шестое занятие

file:///C:/Users/gr4m0/Desktop/Второй%20модуль%20JS/task6.html

Я добавленный текст с помощью метода append!

Я - первый

Я - второй

Я - третий

Я - четвертый

Console Sources Elements Network Performance Memory

<!doctype html>
<html lang="ru">
<head>...</head>
<body>
...<div id="box1"> ...</div>
</body>
</html>

html body div#box1

Styles Event Listeners DOM Breakpoints Properties Accessibility Adguard

Filter: :hov .cls +
element.style {
}
div {
 display: block;
}

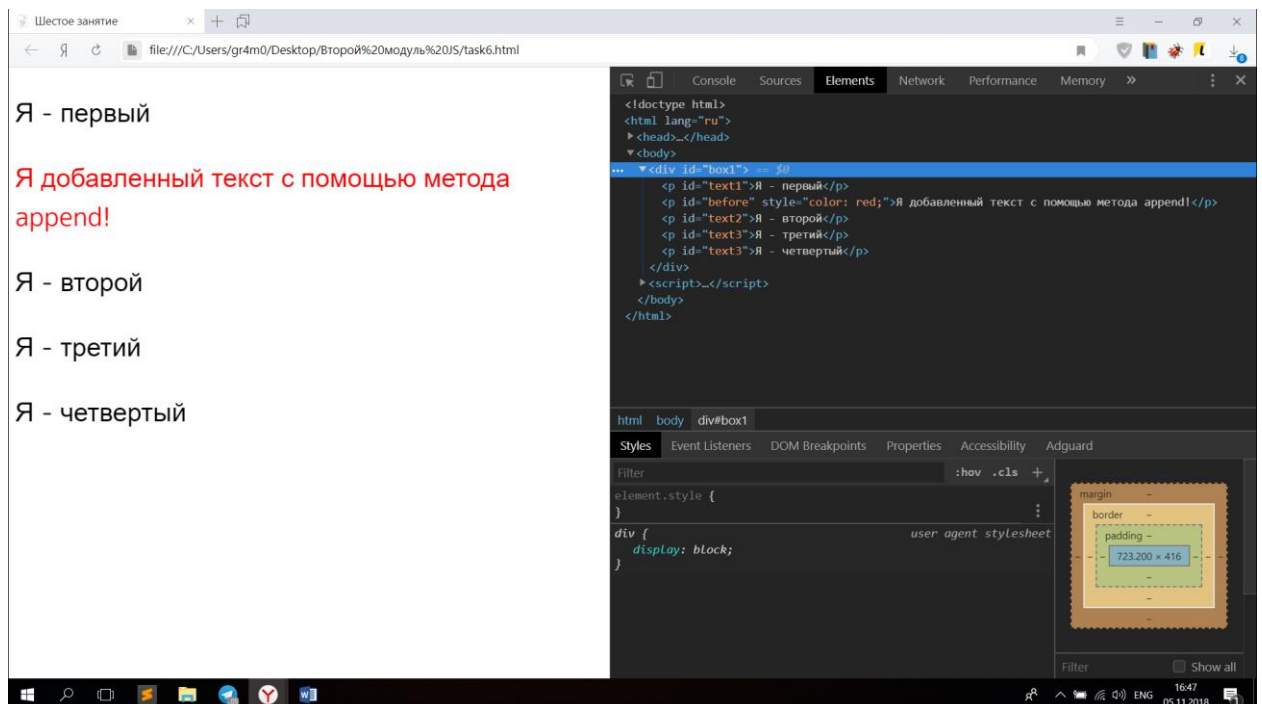
margin -
border -
padding -
723.200 x 416
user agent stylesheet

Filter Show all

Метод after

```
C:\Users\gr4m0\Desktop\Второй модуль\JS\task6.html (python) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

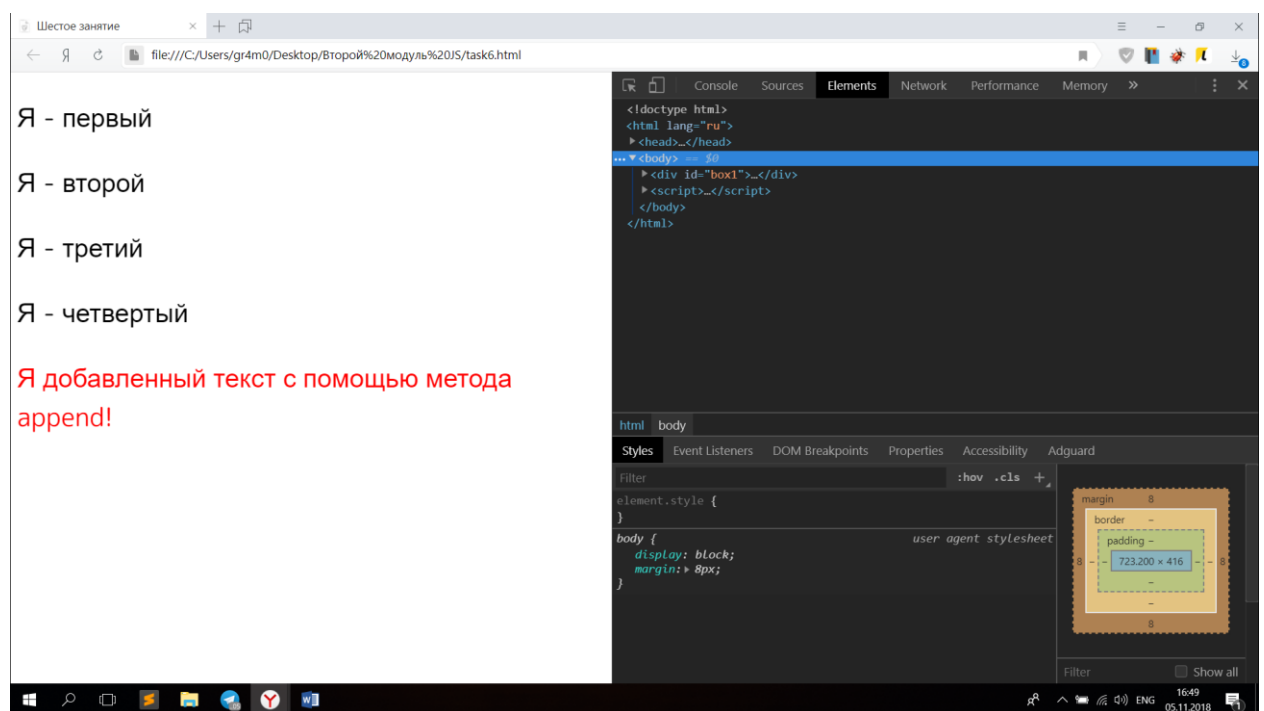
index2.html task5.html task6.html
16 <div id="box1">
17 <p id="text1">Я - первый</p>
18 <p id="text2">Я - второй</p>
19 <p id="text3">Я - третий</p>
20 <p id="text3">Я - четвертый</p>
21 </div>
22 <script>
23 //var removing = document.getElementById('text1');
24 //var removed = document.body.removeChild(removing);
25 // var p = document.getElementsByTagName('p');
26 // while (p.length) {
27 //   document.body.removeChild(p[0]);
28 // }
29 //console.log(removed);
30
31 // var cloned = document.getElementById('box1').cloneNode(false);
32 // cloned.id = 'box2';
33 // document.body.insertBefore(cloned, document.getElementById('box1'))
34
35 var p = document.createElement('p'),
36     beforeP = document.getElementById('text1');
37 p.append('Я добавленный текст с помощью метода append!');
38 p.id = 'before';
39 p.style.color = 'red';
40 // beforeP.before(p);
41 beforeP.after(p);
42
43 </script>
44 </body>
45 </html>
Line 41, Column 22
Tab Size: 4 HTML
```



Метод `append` для вставки узла, он работает аналогично `appendChild`

```
C:\Users\gr4m0\Desktop\Второй модуль JS\task6.html (python) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index2.html task5.html task6.html
16 <div id="box1">
17   <p id="text1">Я - первый</p>
18   <p id="text2">Я - второй</p>
19   <p id="text3">Я - третий</p>
20   <p id="text3">Я - четвертый</p>
21 </div>
22 <script>
23   //var removing = document.getElementById('text1');
24   //var removed = document.body.removeChild(removing);
25   // var p = document.getElementsByTagName('p');
26   // while (p.length) {
27   //   document.body.removeChild(p[0]);
28   // }
29   //console.log(removed);
30
31   // var cloned = document.getElementById('box1').cloneNode(false);
32   // cloned.id = 'box2';
33   // document.body.insertBefore(cloned, document.getElementById('box1'))
34
35   var p = document.createElement('p'),
36       beforeP = document.getElementById('text1');
37   p.append('Я добавленный текст с помощью метода append!');
38   p.id = 'before';
39   p.style.color = 'red';
40   // beforeP.before(p);
41   // beforeP.after(p);
42   var box1 = document.getElementById('box1');
43   box1.append(p);
44
45
Line 43, Column 24: Saved C:\Users\gr4m0\Desktop\Второй модуль JS\task6.html (UTF-8)
Tab Size: 4 HTML
16:48 05.11.2018
```



Метод prepend

```
C:\Users\gr4m0\Desktop\Второй модуль JS\task6.html (python) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index2.html task5.html task6.html
22 <script>
23 //var removing = document.getElementById('text1');
24 //var removed = document.body.removeChild(removing);
25 // var p = document.getElementsByTagName('p');
26 // while (p.length) {
27 //   document.body.removeChild(p[0]);
28 // }
29 //console.log(removed);
30
31 // var cloned = document.getElementById('box1').cloneNode(false);
32 // cloned.id = 'box2';
33 // document.body.insertBefore(cloned, document.getElementById('box1'))
34
35 var p = document.createElement('p'),
36     beforeP = document.getElementById('text1');
37 p.append('Я добавленный текст с помощью метода append!');
38 p.id = 'before';
39 p.style.color = 'red';
40 // beforeP.before(p);
41 // beforeP.after(p);
42 var box1 = document.getElementById('box1');
43 box1.prepend(p);
44
45
46 </script>
47 </body>
48 </html>
```

Шестое занятие

Я добавленный текст с помощью метода append!

Я - первый

Я - второй

Я - третий

Я - четвертый

file:///C:/Users/gr4m0/Desktop/Второй%20модуль%20JS/task6.html

Console Sources Elements Network Performance Memory

```
<!doctype html>
<html lang="ru">
  <head>_</head>
  <body>_
    <div id="box1">_</div>
    <script>_</script>
  </body>
</html>
```

html body

Styles Event Listeners DOM Breakpoints Properties Accessibility Adguard

Filter :hov .cls +

```
element.style {
}

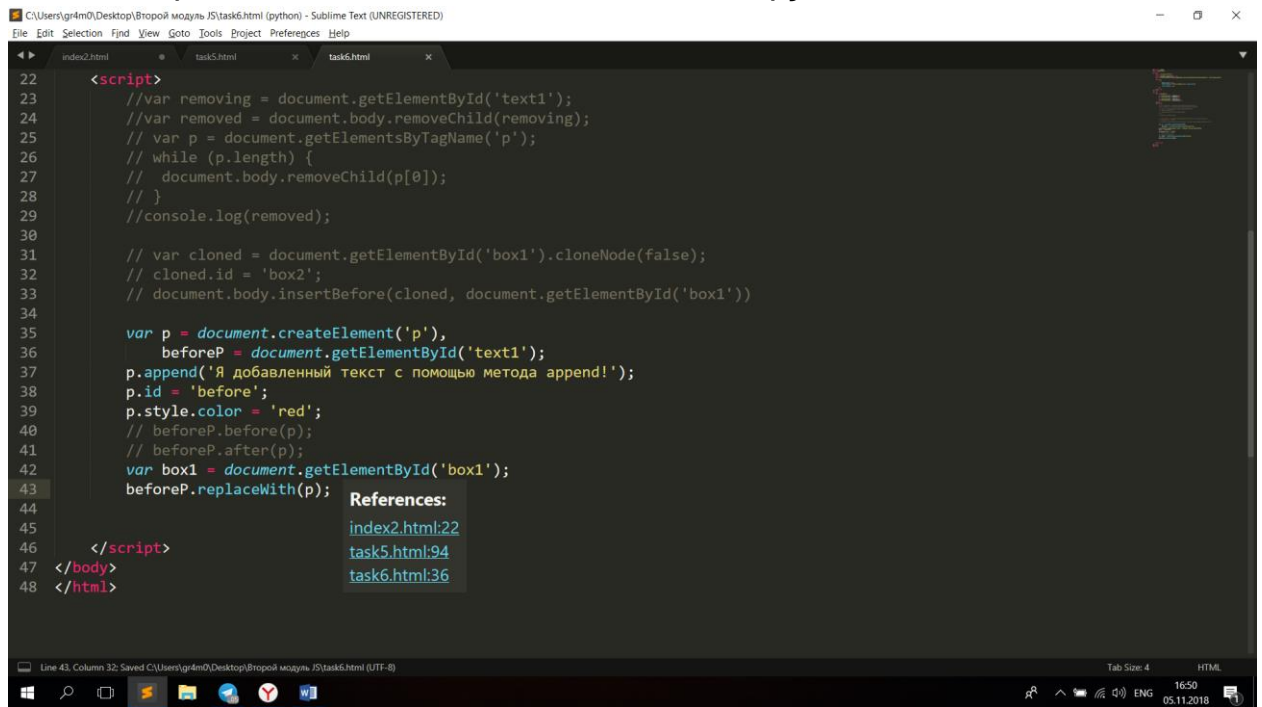
body {
  display: block;
  margin: 8px;
}
```

user agent stylesheet

margin 8
border -
padding -
723.200 x 416
8

Filter Show all

Метод `replaceWith` заменяет один тег другим.

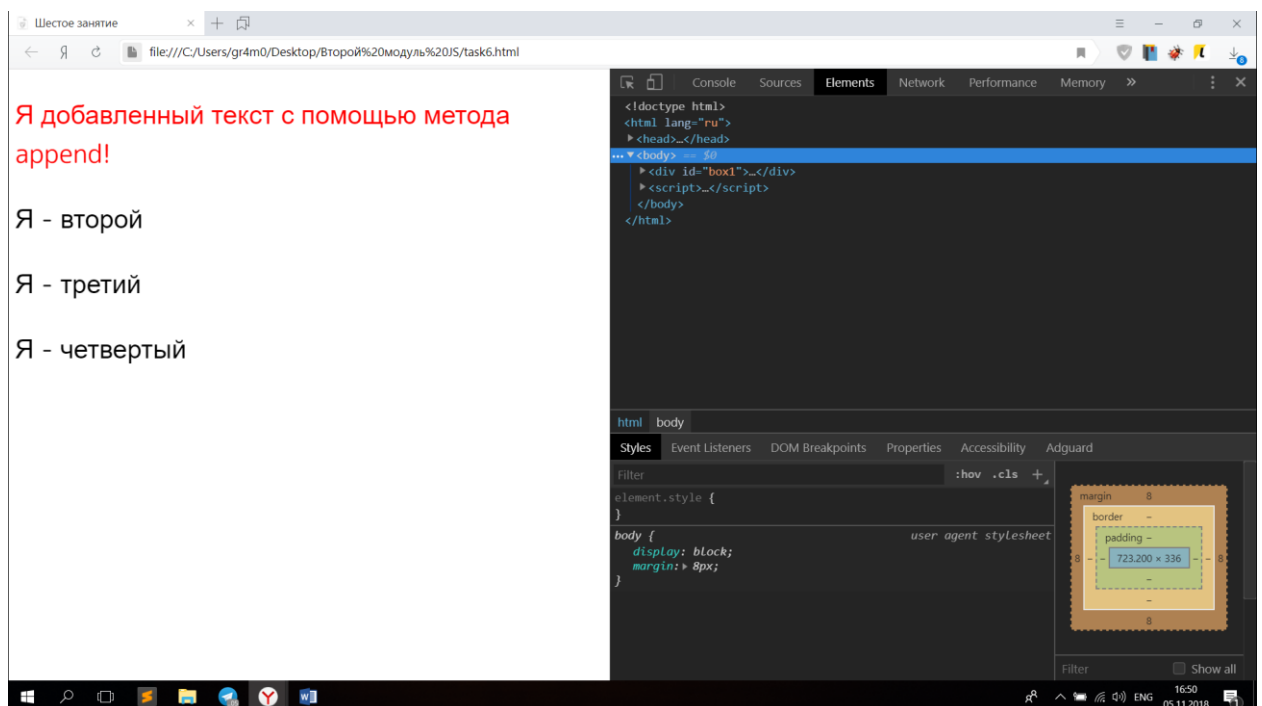


The screenshot shows the Sublime Text editor with a file named `task6.html`. The code is as follows:

```
22 <script>
23 //var removing = document.getElementById('text1');
24 //var removed = document.body.removeChild(removing);
25 // var p = document.getElementsByTagName('p');
26 // while (p.length) {
27 //   document.body.removeChild(p[0]);
28 // }
29 //console.log(removed);
30
31 // var cloned = document.getElementById('box1').cloneNode(false);
32 // cloned.id = 'box2';
33 // document.body.insertBefore(cloned, document.getElementById('box1'))
34
35 var p = document.createElement('p'),
36     beforeP = document.getElementById('text1');
37 p.append('Я добавленный текст с помощью метода append!');
38 p.id = 'before';
39 p.style.color = 'red';
40 // beforeP.before(p);
41 // beforeP.after(p);
42 var box1 = document.getElementById('box1');
43 beforeP.replaceWith(p);
44
45 </script>
46 </body>
47 </html>
```

References:

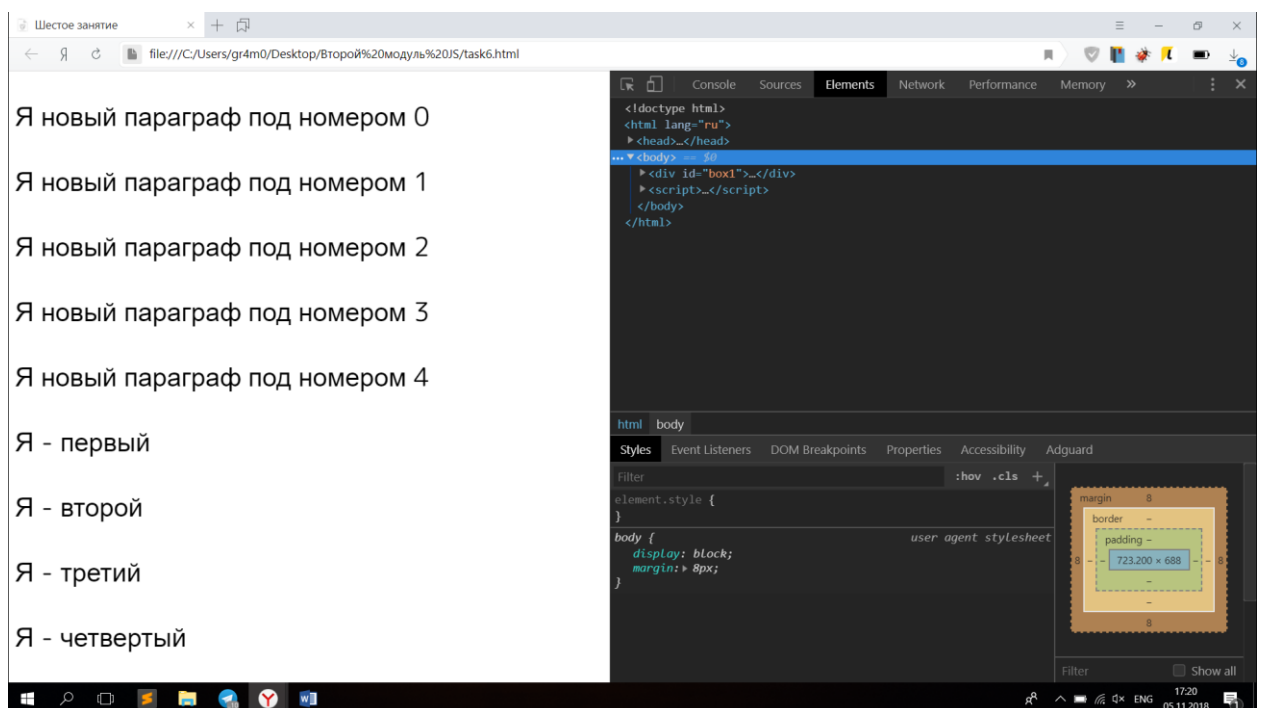
- [index2.html:22](#)
- [task5.html:94](#)
- [task6.html:36](#)



Также мы можем вставлять несколько узлов с помощью этих методов.

```
C:\Users\gr4m0\Desktop\Второй модуль\JS\task6.html (python) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index2.html task5.html task6.html
32 // cloned.id = 0000;
33 // document.body.insertBefore(cloned, document.getElementById('box1'))
34
35 // var p = document.createElement('p');
36 // beforeP = document.getElementById('text1');
37 // p.append('Я добавленный текст с помощью метода append!');
38 // p.id = 'before';
39 // p.style.color = 'red';
40 // // beforeP.before(p);
41 // // beforeP.after(p);
42 // var box1 = document.getElementById('box1');
43 // beforeP.replaceWith(p);
44 var pCollection = [],
45     box1 = document.getElementById('box1');
46 for (var i = 0; i < 5; i++) {
47     pCollection.push(document.createElement('p'));
48     pCollection[i].textContent = 'Я новый параграф под номером ' + i;
49     pCollection[i].id = 'p' + i;
50 }
51 console.log(pCollection);
52 box1.prepend(...pCollection);
53
54 </script>
55 </body>
56 </html>
```

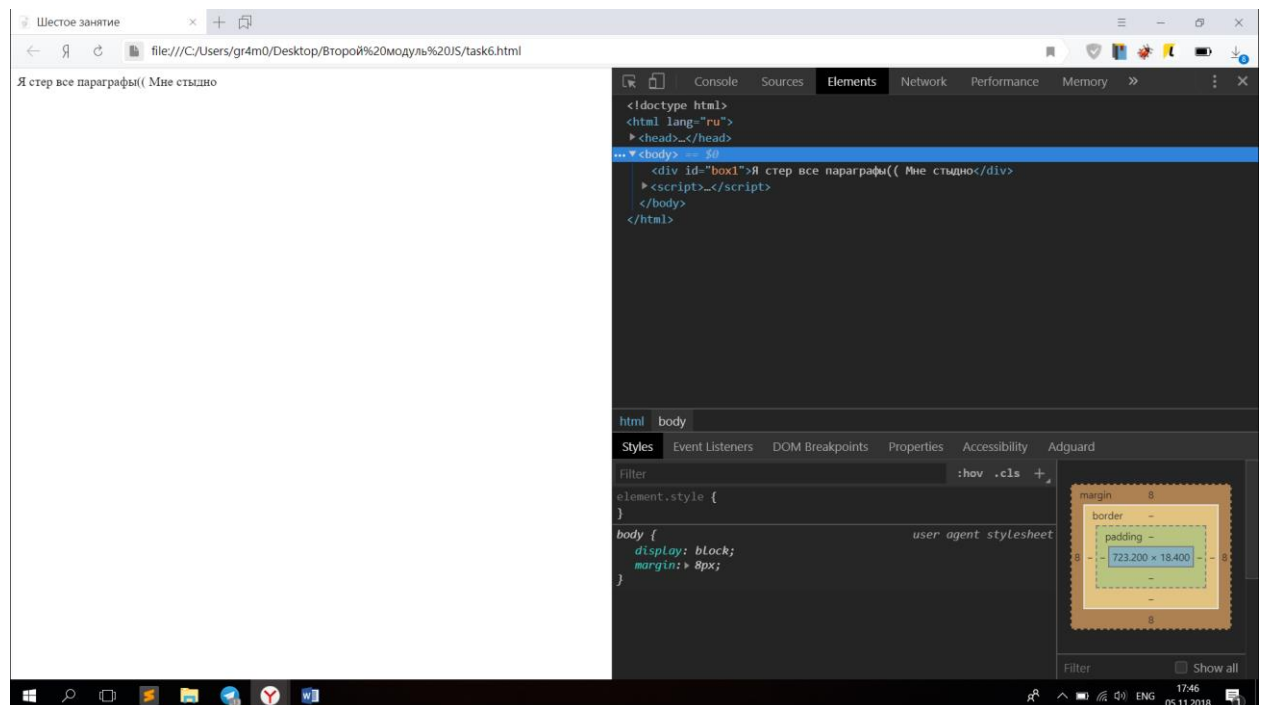


Я использовал новый оператор `...`, этот оператор называется `spread` или оператор деструктуризации. *Деструктуризация* (destructuring assignment) – это особый синтаксис присваивания, при котором можно присвоить массив или объект сразу нескольким переменным, разбив его на части. Поэкспериментируйте добавление массива узлов с всеми методами.

Еще один метод записи элемент innerHTML. Это свойство есть у каждого элемента, с помощью него мы можем полностью переписать всё содержимое тега.

```
C:\Users\gr4m0\Desktop\Второй модуль JS\task6.html (python) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

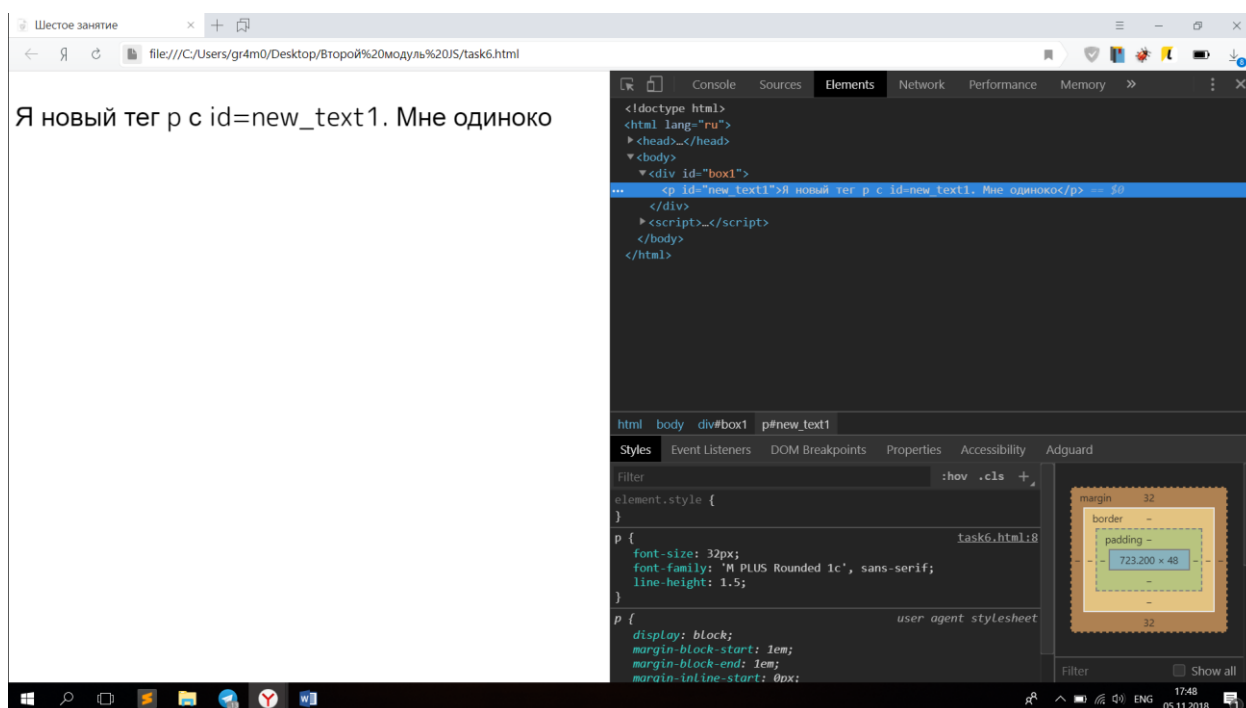
task6.html
32 // cloned.id = 00xx;
33 // document.body.insertBefore(cloned, document.getElementById('box1'))
34
35 // var p = document.createElement('p');
36 // beforeP = document.getElementById('text1');
37 // p.append('Я добавленный текст с помощью метода append!');
38 // p.id = 'before';
39 // p.style.color = 'red';
40 // // beforeP.before(p);
41 // // beforeP.after(p);
42 // var box1 = document.getElementById('box1');
43 // beforeP.replaceWith(p);
44
45 // var pCollection = [],
46 // box1 = document.getElementById('box1');
47 // for (var i = 0; i < 5; i++) {
48 //   pCollection.push(document.createElement('p'));
49 //   pCollection[i].textContent = 'Я новый параграф под номером ' + i;
50 //   pCollection[i].id = 'p' + i;
51 // }
52 // console.log(pCollection);
53 // box1.prepend(...pCollection);
54
55 var box1 = document.getElementById('box1');
56 box1.innerHTML = 'Я стер все параграфы(( Мне стыдно';
57
58 </script>
59 </body>
60 </html>
```



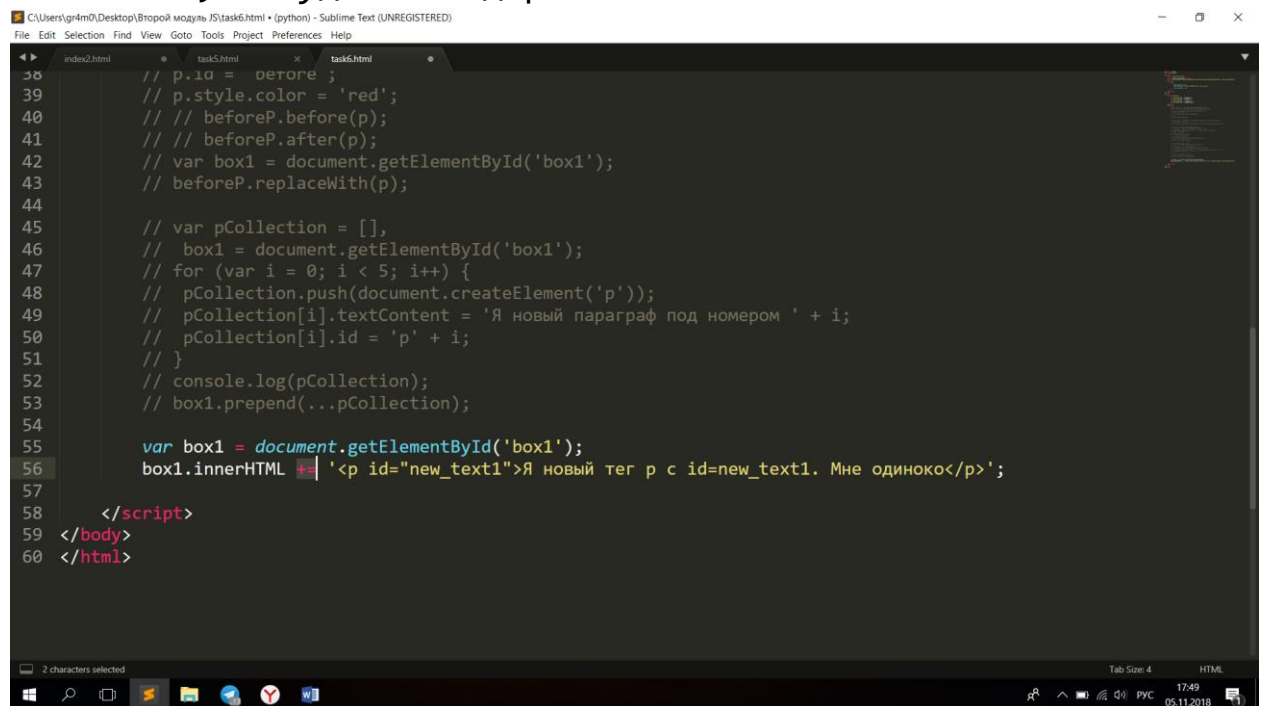
```
C:\Users\gr4m0\Desktop\Второй модуль JS\task6.html (python) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index2.html task5.html task6.html
32 // cloned, u = uoxk;
33 // document.body.insertBefore(cloned, document.getElementById('box1'))
34
35 // var p = document.createElement('p');
36 // beforeP = document.getElementById('text1');
37 // p.append('Я добавленный текст с помощью метода append!');
38 // p.id = 'before';
39 // p.style.color = 'red';
40 // // beforeP.before(p);
41 // // beforeP.after(p);
42 // var box1 = document.getElementById('box1');
43 // beforeP.replaceWith(p);
44
45 // var pCollection = [],
46 // box1 = document.getElementById('box1');
47 // for (var i = 0; i < 5; i++) {
48 //   pCollection.push(document.createElement('p'));
49 //   pCollection[i].textContent = 'Я новый параграф под номером ' + i;
50 //   pCollection[i].id = 'p' + i;
51 // }
52 // console.log(pCollection);
53 // box1.prepend(...pCollection);
54
55 var box1 = document.getElementById('box1');
56 box1.innerHTML = '<p id="new_text1">Я новый тер p с id=new_text1. Мне одиноко</p>';
57
58 </script>
59 </body>
60 </html>

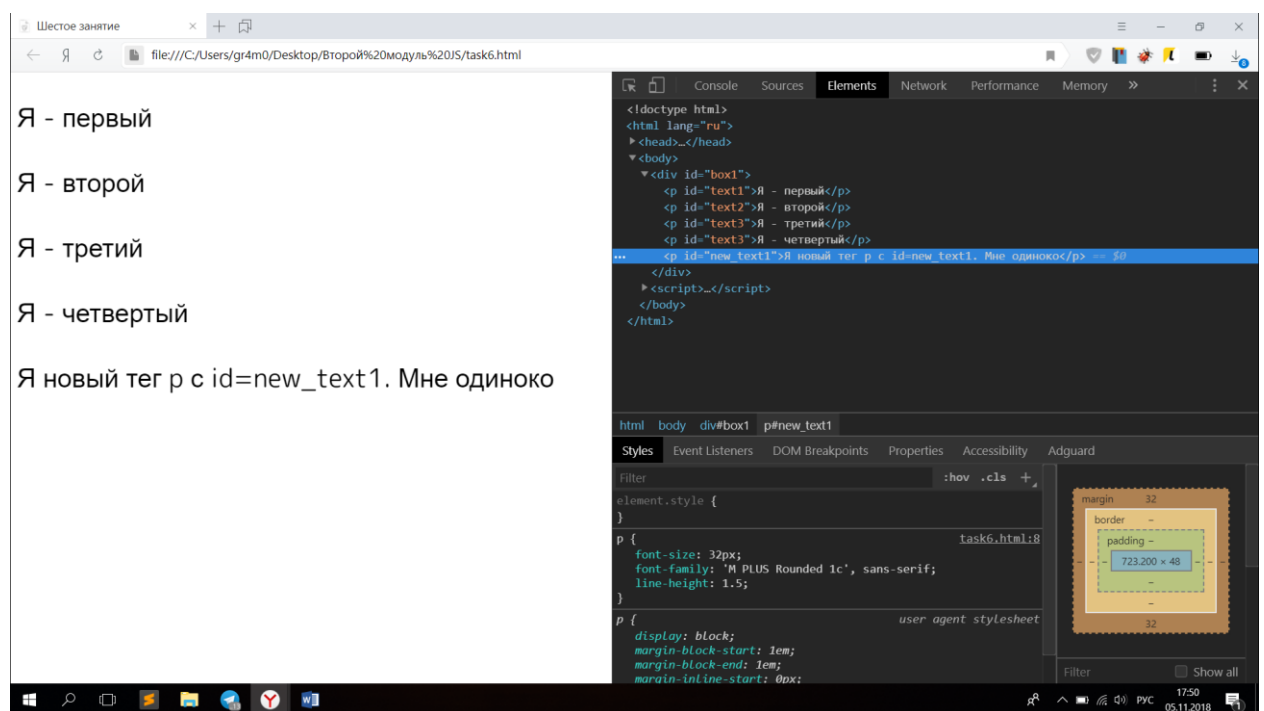
Line 57, Column 1: Saved C:\Users\gr4m0\Desktop\Второй модуль JS\task6.html (UTF-8)
Tab Size: 4 HTML
17:47 05.11.2018
```



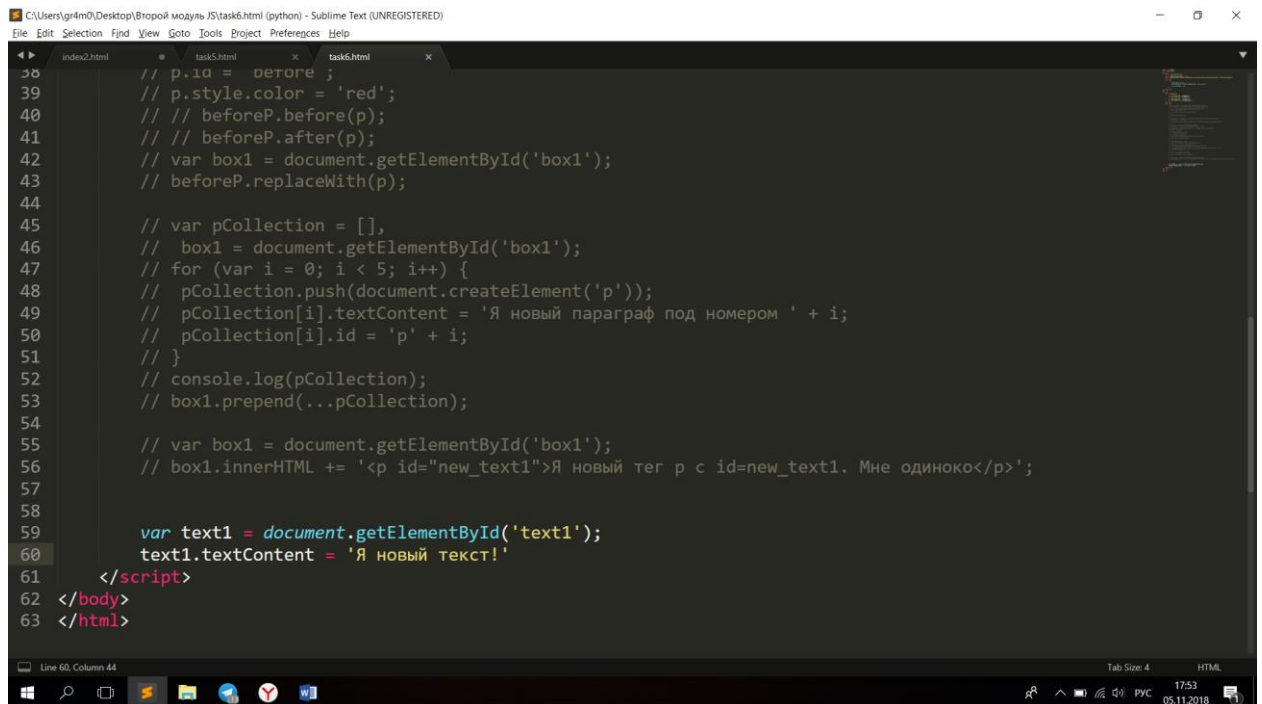
Я использовал конкатенацию строк и добавил в поле innerHTML новый тег, не удаляя содержимого.



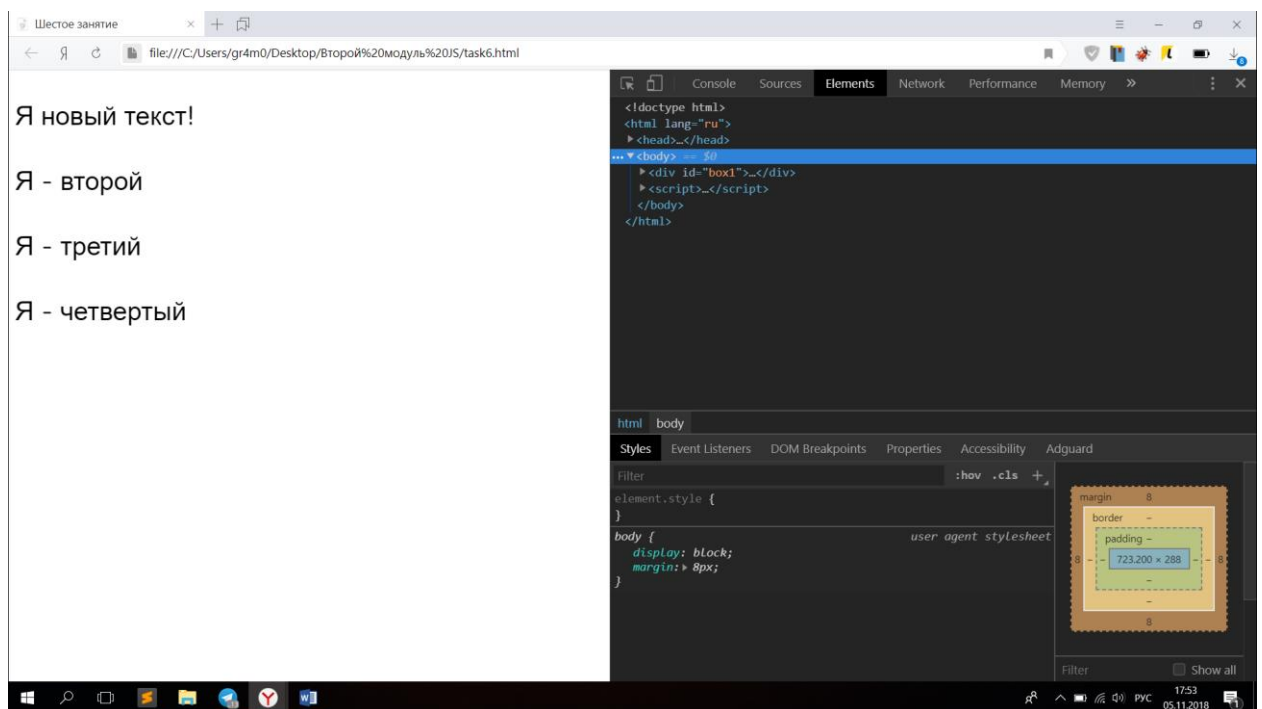
```
38 // p.id = before ;
39 // p.style.color = 'red';
40 // // beforeP.before(p);
41 // // beforeP.after(p);
42 // var box1 = document.getElementById('box1');
43 // beforeP.replaceWith(p);
44
45 // var pCollection = [],
46 // box1 = document.getElementById('box1');
47 // for (var i = 0; i < 5; i++) {
48 //   pCollection.push(document.createElement('p'));
49 //   pCollection[i].textContent = 'Я новый параграф под номером ' + i;
50 //   pCollection[i].id = 'p' + i;
51 // }
52 // console.log(pCollection);
53 // box1.prepend(...pCollection);
54
55 var box1 = document.getElementById('box1');
56 box1.innerHTML += '<p id="new_text1">Я новый тер п с id=new_text1. Мне одиноко</p>';
57
58 </script>
59 </body>
60 </html>
```



У текстовых элементов с помощью поля `textContent` мы можем менять текстовый контент.



```
38 // p.id = before;
39 // p.style.color = 'red';
40 // // beforeP.before(p);
41 // // beforeP.after(p);
42 // var box1 = document.getElementById('box1');
43 // beforeP.replaceWith(p);
44
45 // var pCollection = [],
46 // box1 = document.getElementById('box1');
47 // for (var i = 0; i < 5; i++) {
48 //   pCollection.push(document.createElement('p'));
49 //   pCollection[i].textContent = 'Я новый параграф под номером ' + i;
50 //   pCollection[i].id = 'p' + i;
51 // }
52 // console.log(pCollection);
53 // box1.prepend(...pCollection);
54
55 // var box1 = document.getElementById('box1');
56 // box1.innerHTML += '<p id="new_text1">Я новый тер п с id=new_text1. Мне одиноко</p>';
57
58
59 var text1 = document.getElementById('text1');
60 text1.textContent = 'Я новый текст!';
61 </script>
62 </body>
63 </html>
```

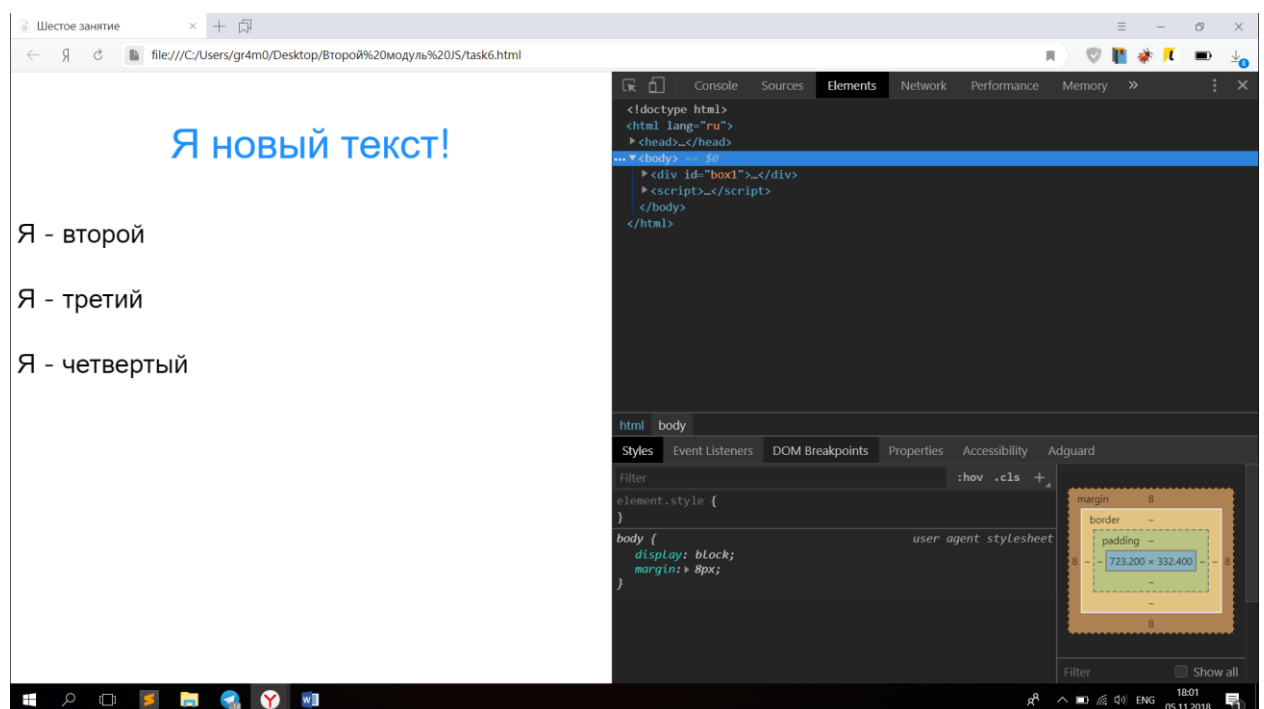


`textContent` меняет только текст(!).

В предыдущем занятии мы изменяли CSS-свойства с помощью `element.style.property`. Но если мы изменяем несколько свойств наш код начинает линейно возрастать. Это не очень хорошо. У этой проблемы есть решение `element.style.cssText`

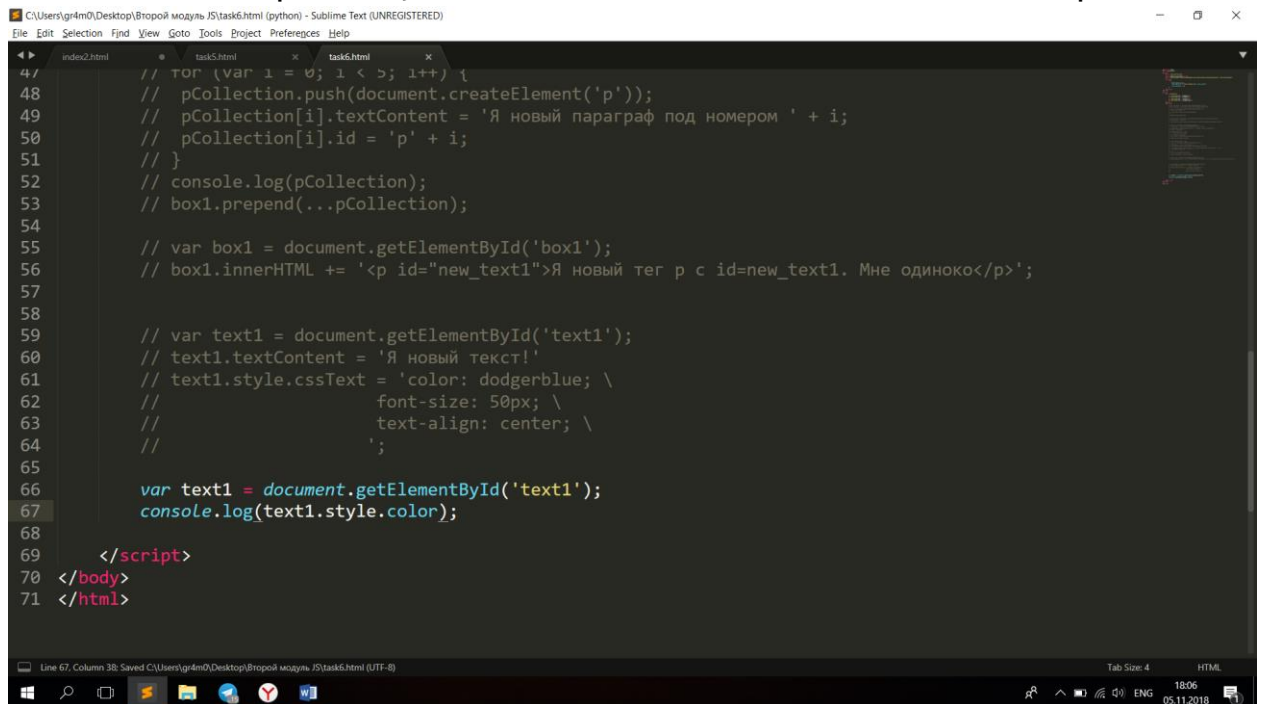
```
C:\Users\gr4m0\Desktop\Второй модуль JS\task6.html (python) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

41 // // beforeP.replaceWith(p);
42 // var box1 = document.getElementById('box1');
43 // beforeP.replaceWith(p);
44
45 // var pCollection = [],
46 // box1 = document.getElementById('box1');
47 // for (var i = 0; i < 5; i++) {
48 //   pCollection.push(document.createElement('p'));
49 //   pCollection[i].textContent = 'Я новый параграф под номером ' + i;
50 //   pCollection[i].id = 'p' + i;
51 // }
52 // console.log(pCollection);
53 // box1.prepend(...pCollection);
54
55 // var box1 = document.getElementById('box1');
56 // box1.innerHTML += '<p id="new_text1">Я новый тер п с id=new_text1. Мне одиноко</p>';
57
58
59
60 var text1 = document.getElementById('text1');
61 text1.textContent = 'Я новый текст!';
62 text1.style.cssText = 'color: dodgerblue; \
63                       font-size: 50px; \
64                       text-align: center; \
65                       ';
66
67 </script>
68 </body>
```



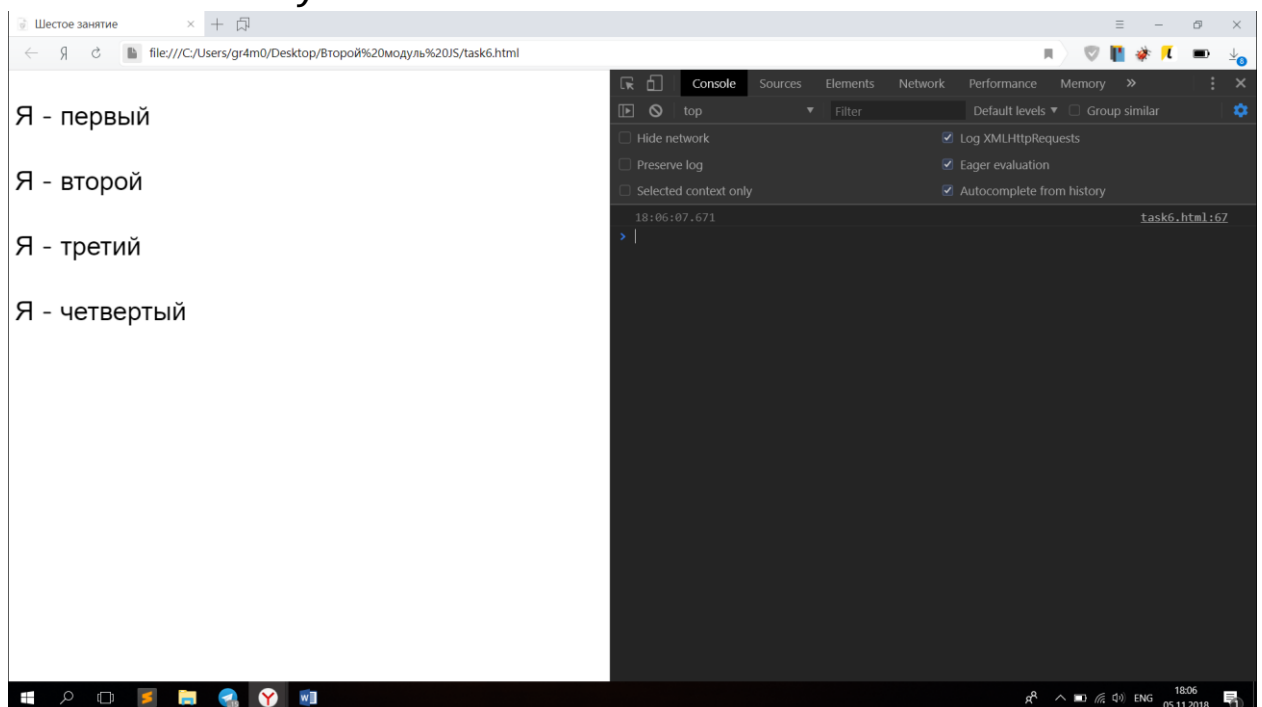
Обратите внимание на синтаксис написания стилей. После каждой строки мы ставим обратный слеш, а в конце на новой(!) строке ставим закрывающую кавычку.

В работе с документом нам может понадобиться чтение стилей из тега `style`, расположенного в теге `head`. Например, если цвет текста красный, то поменяй его на синий и наоборот.



```
47 // for (var i = 0; i < 5; i++) {
48 //   pCollection.push(document.createElement('p'));
49 //   pCollection[i].textContent = 'Я новый параграф под номером ' + i;
50 //   pCollection[i].id = 'p' + i;
51 // }
52 // console.log(pCollection);
53 // box1.prepend(...pCollection);
54
55 // var box1 = document.getElementById('box1');
56 // box1.innerHTML += '<p id="new_text1">Я новый тег p с id=new_text1. Мне одиноко</p>';
57
58
59 // var text1 = document.getElementById('text1');
60 // text1.textContent = 'Я новый текст!';
61 // text1.style.cssText = 'color: dodgerblue; \
62 //                       font-size: 50px; \
63 //                       text-align: center; \
64 //                       ';
65
66 var text1 = document.getElementById('text1');
67 console.log(text1.style.color);
68
69 </script>
70 </body>
71 </html>
```

Но в консоли пусто!

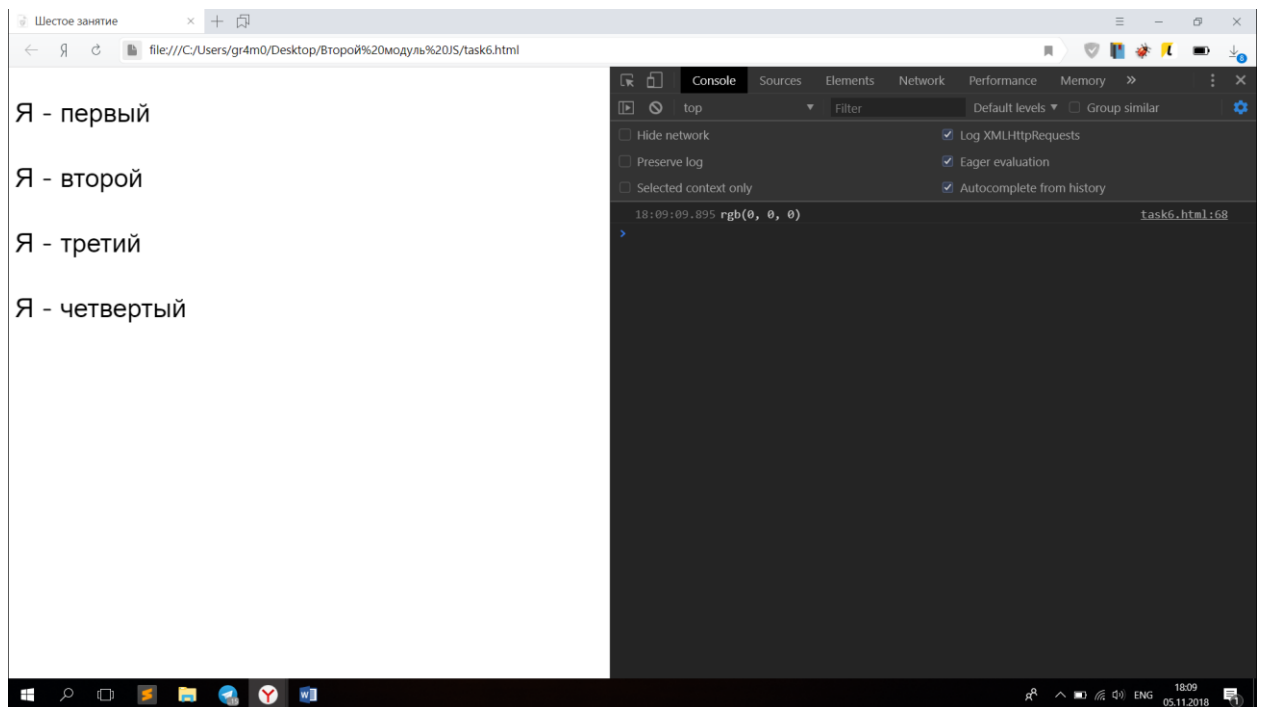


Для этого существует метод `getComputedStyle`

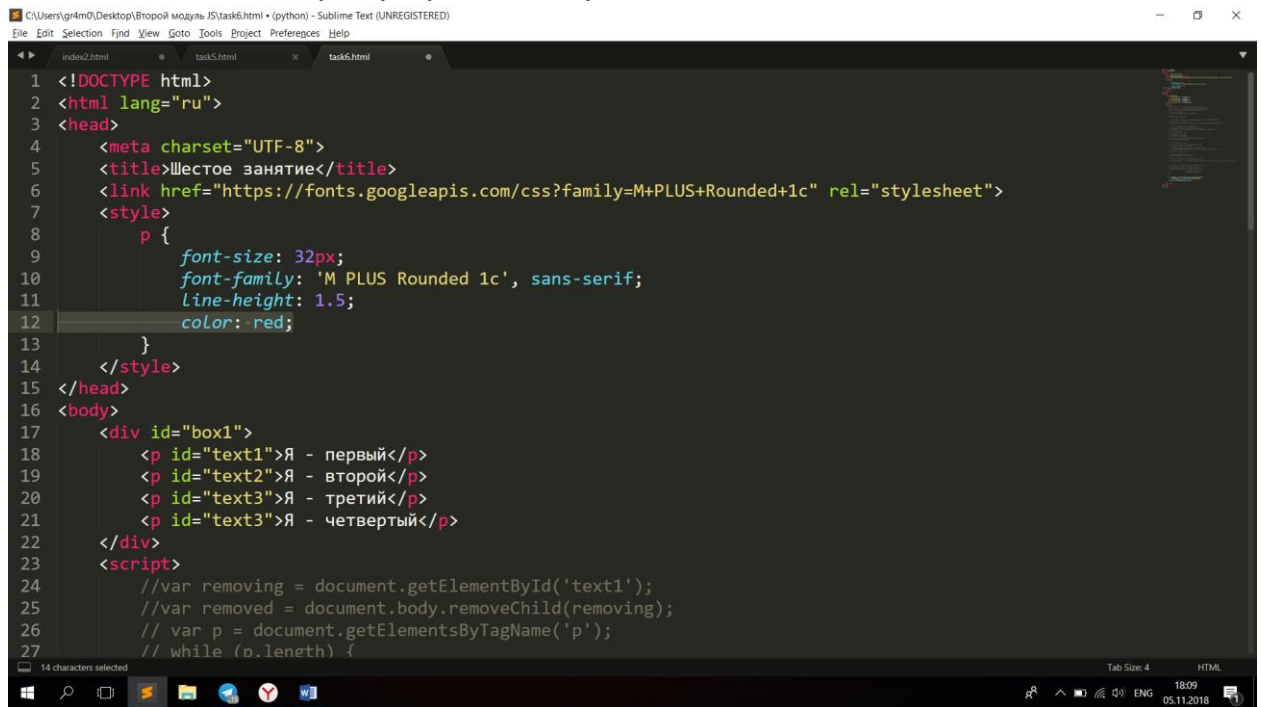
```
C:\Users\gr4m0\Desktop\Второй модуль JS\task6.html (python) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index2.html task5.html task6.html
47 // top (var i = 0; i < 5; i++) {
48 //   pCollection.push(document.createElement('p'));
49 //   pCollection[i].textContent = 'Я новый параграф под номером ' + i;
50 //   pCollection[i].id = 'p' + i;
51 // }
52 // console.log(pCollection);
53 // box1.prepend(...pCollection);
54
55 // var box1 = document.getElementById('box1');
56 // box1.innerHTML += '<p id="new_text1">Я новый тег p с id=new_text1. Мне одиноко</p>';
57
58
59 // var text1 = document.getElementById('text1');
60 // text1.textContent = 'Я новый текст!';
61 // text1.style.cssText = 'color: dodgerblue; \
62 //                       font-size: 50px; \
63 //                       text-align: center; \
64 //                       ';
65
66 var text1 = document.getElementById('text1'),
67   getPStyles = getComputedStyle(text1);
68 console.log(getPStyles.color);
69
70 </script>
71 </body>
72 </html>

Line 68, Column 39: Saved C:\Users\gr4m0\Desktop\Второй модуль JS\task6.html (UTF-8)
Tab Size: 4 HTML
1809 05.11.2018
```

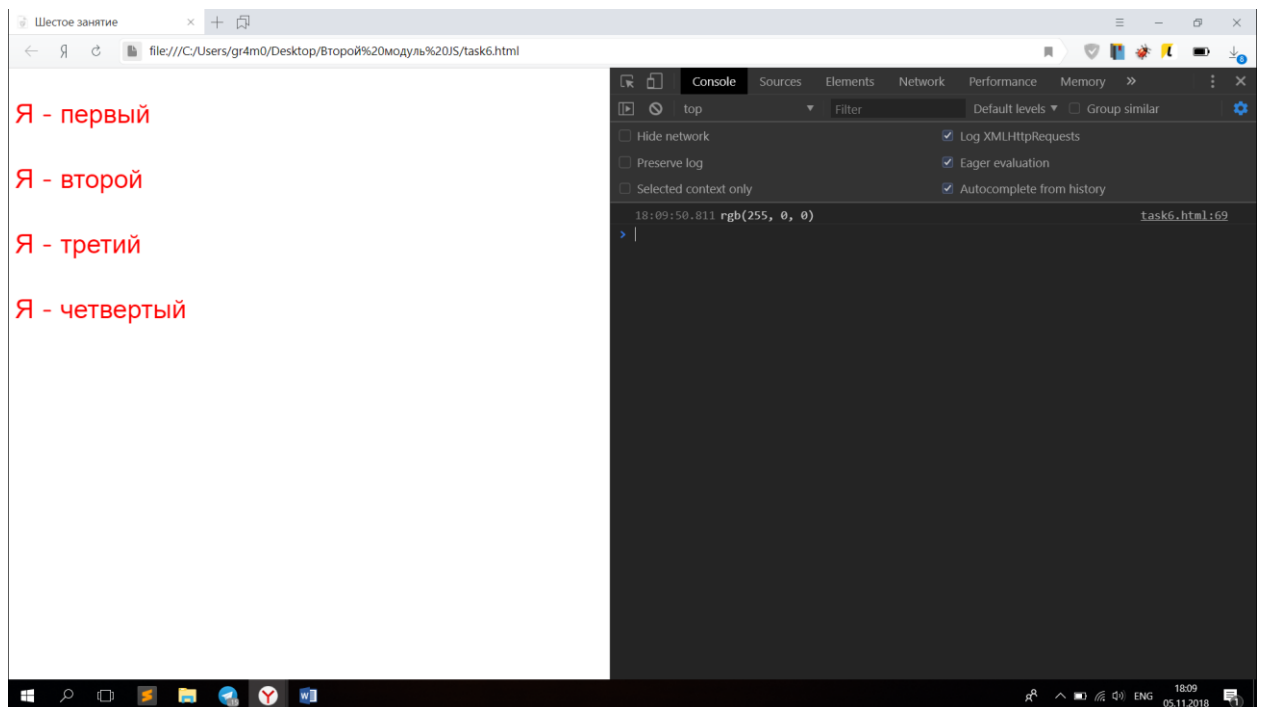


Изменим цвет параграфов на красный в теге head

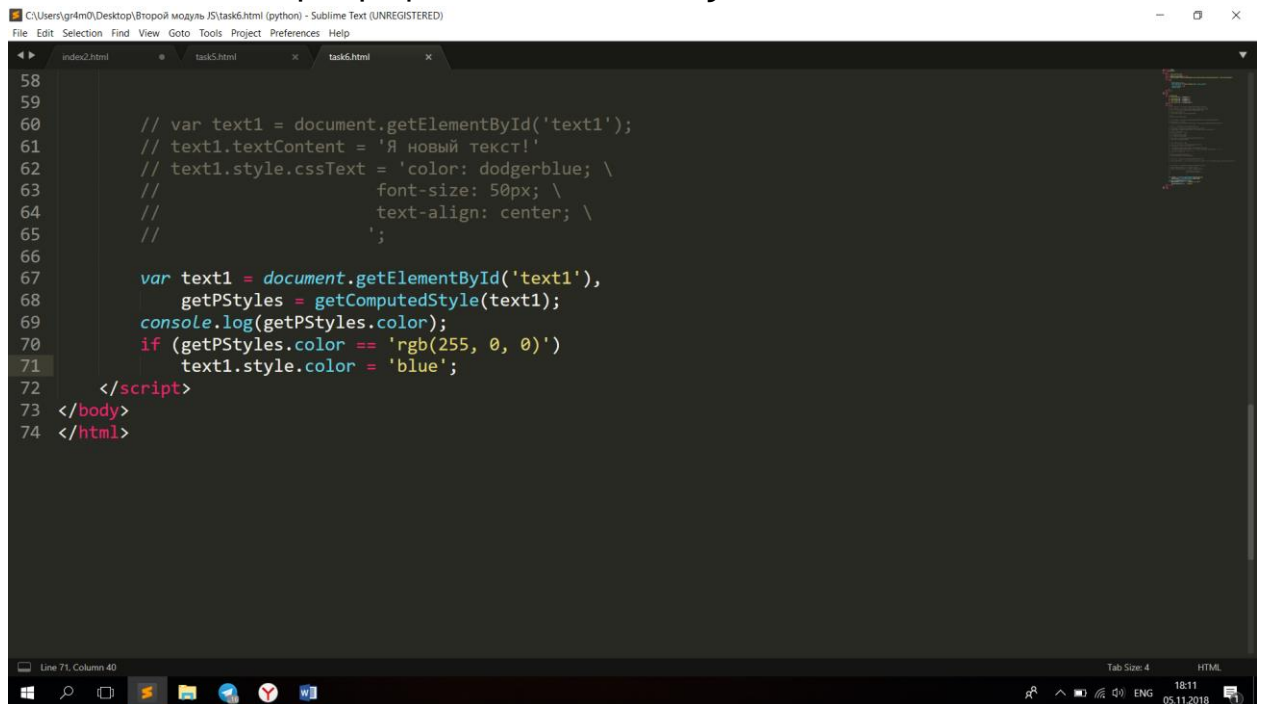


```
1 <!DOCTYPE html>
2 <html lang="ru">
3 <head>
4   <meta charset="UTF-8">
5   <title>Шестое занятие</title>
6   <link href="https://fonts.googleapis.com/css?family=M+PLUS+Rounded+1c" rel="stylesheet">
7   <style>
8     p {
9       font-size: 32px;
10      font-family: 'M PLUS Rounded 1c', sans-serif;
11      line-height: 1.5;
12      color: red;
13    }
14  </style>
15 </head>
16 <body>
17   <div id="box1">
18     <p id="text1">Я - первый</p>
19     <p id="text2">Я - второй</p>
20     <p id="text3">Я - третий</p>
21     <p id="text3">Я - четвертый</p>
22   </div>
23   <script>
24     //var removing = document.getElementById('text1');
25     //var removed = document.body.removeChild(removing);
26     // var p = document.getElementsByTagName('p');
27     // while (p.length) {
```

Теперь в консоли мы увидим красный в палитре rgb

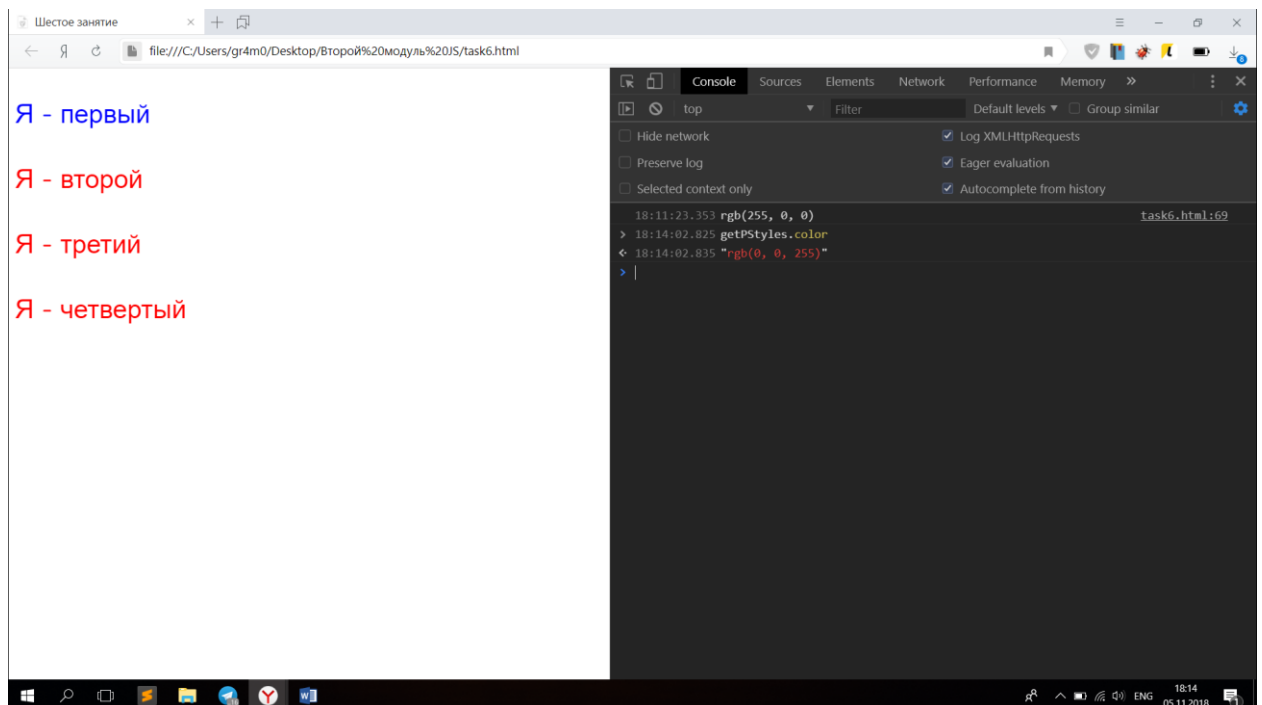


Если цвет параграфа красный, то изменим его на синий. Так как `getComputedStyle` является неизменяемым свойством, то меняем цвет параграфа с помощью `style`.



The screenshot shows the Sublime Text editor with a file named `task6.html`. The code is as follows:

```
58
59
60 // var text1 = document.getElementById('text1');
61 // text1.textContent = 'Я новый текст!';
62 // text1.style.cssText = 'color: dodgerblue; \
63 //                               font-size: 50px; \
64 //                               text-align: center; \
65 //                               ';
66
67 var text1 = document.getElementById('text1'),
68     getPStyles = getComputedStyle(text1);
69 console.log(getPStyles.color);
70 if (getPStyles.color == 'rgb(255, 0, 0)')
71     text1.style.color = 'blue';
72 </script>
73 </body>
74 </html>
```



Последнее, что мы рассмотрим в рамках темы «работа с элементами».

Атрибуты тегов

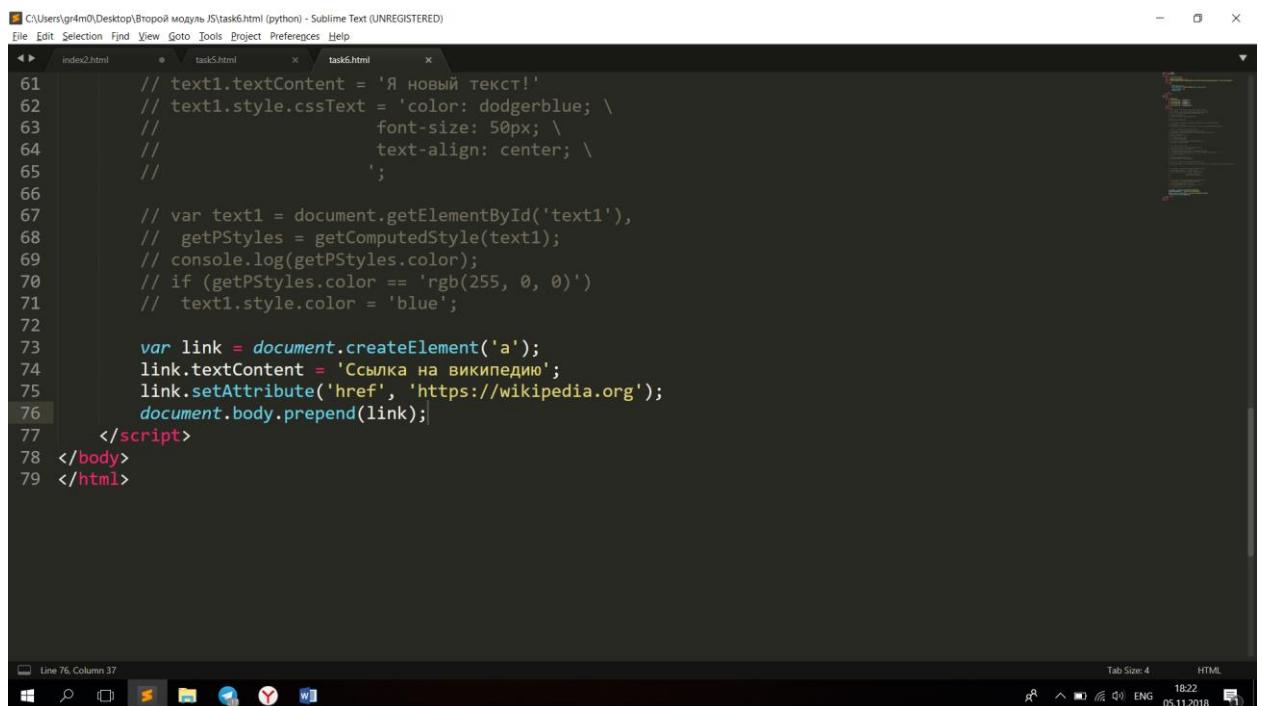
Элементам DOM, с другой стороны, соответствуют HTML-теги, у которых есть текстовые атрибуты.

Конечно, здесь речь именно об узлах-элементах, не о текстовых узлах или комментариях.

Доступ к атрибутам осуществляется при помощи стандартных методов:

- `elem.hasAttribute(name)` – проверяет наличие атрибута
- `elem.getAttribute(name)` – получает значение атрибута
- `elem.setAttribute(name, value)` – устанавливает атрибут
- `elem.removeAttribute(name)` – удаляет атрибут

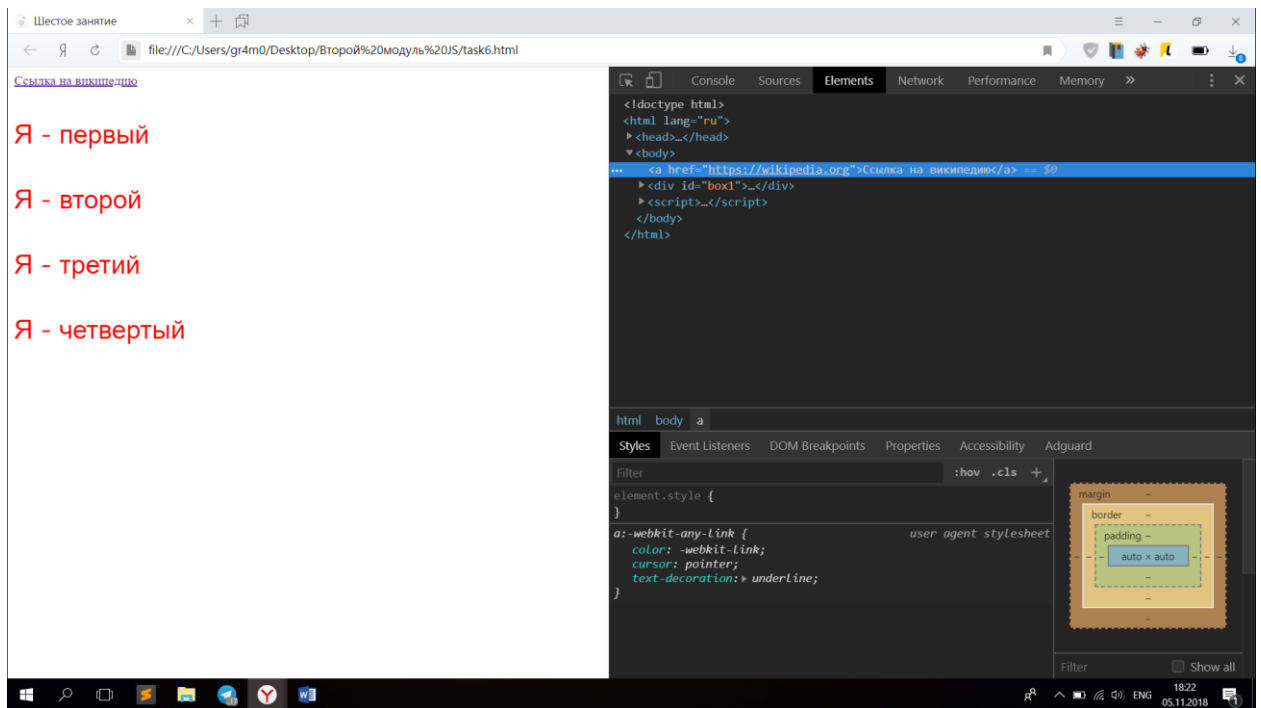
Посмотрим на эти методы в полевых условиях



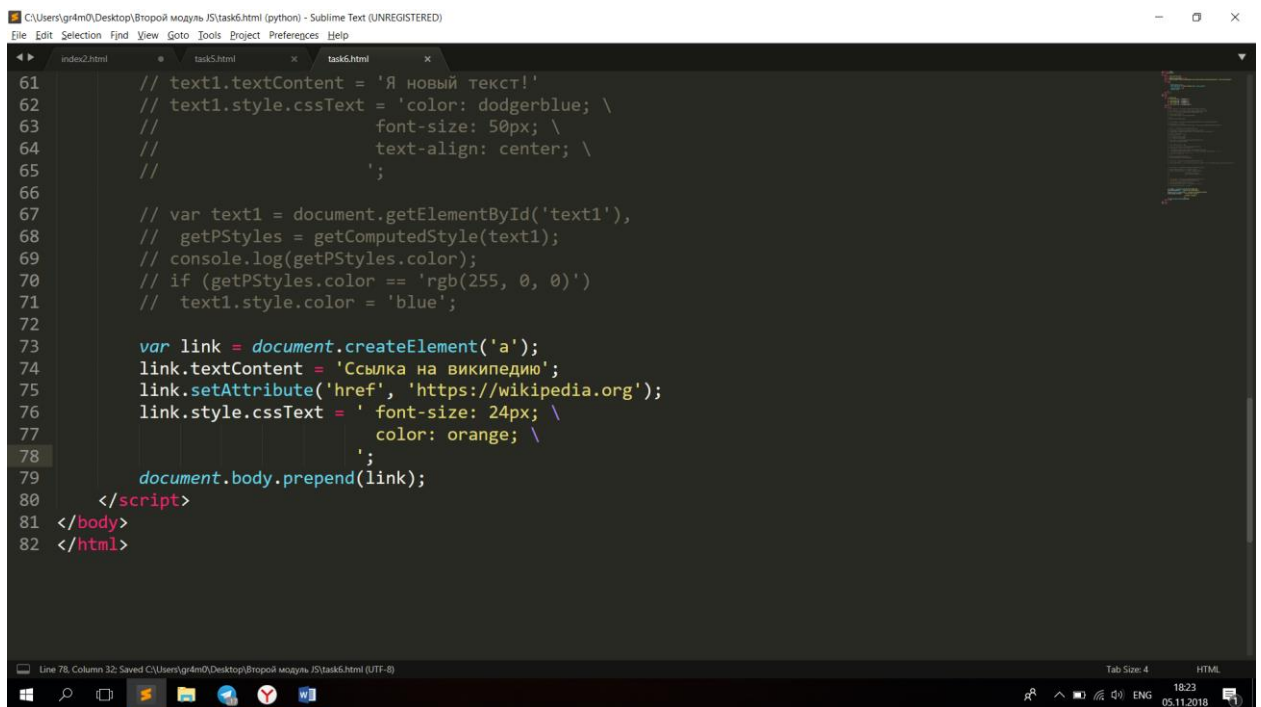
The screenshot shows a Sublime Text editor window with the following code:

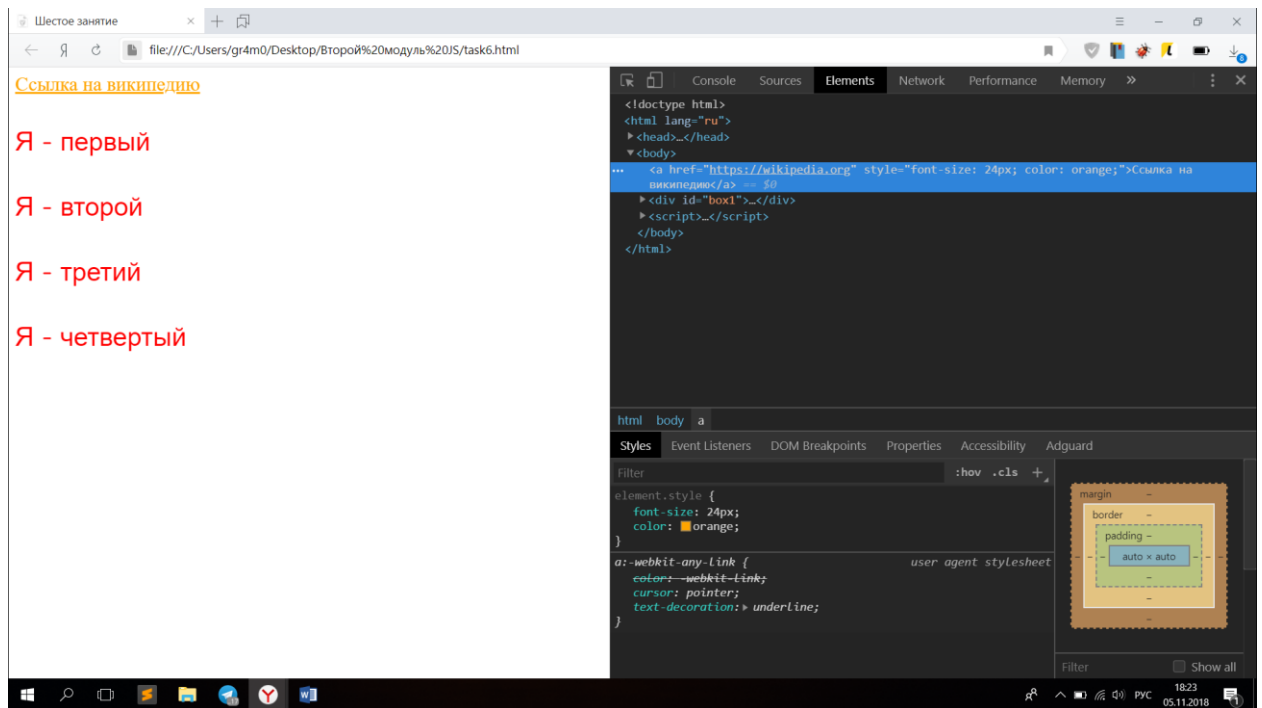
```
61 // text1.textContent = 'Я новый текст!';
62 // text1.style.cssText = 'color: dodgerblue; \
63 //                               font-size: 50px; \
64 //                               text-align: center; \
65 //                               ';
66
67 // var text1 = document.getElementById('text1'),
68 //     getPStyles = getComputedStyle(text1);
69 // console.log(getPStyles.color);
70 // if (getPStyles.color == 'rgb(255, 0, 0)')
71 //     text1.style.color = 'blue';
72
73 var link = document.createElement('a');
74 link.textContent = 'Ссылка на википедию';
75 link.setAttribute('href', 'https://wikipedia.org');
76 document.body.prepend(link);
77 </script>
78 </body>
79 </html>
```

The status bar at the bottom indicates 'Line 76, Column 37', 'Tab Size: 4', and 'HTML'.

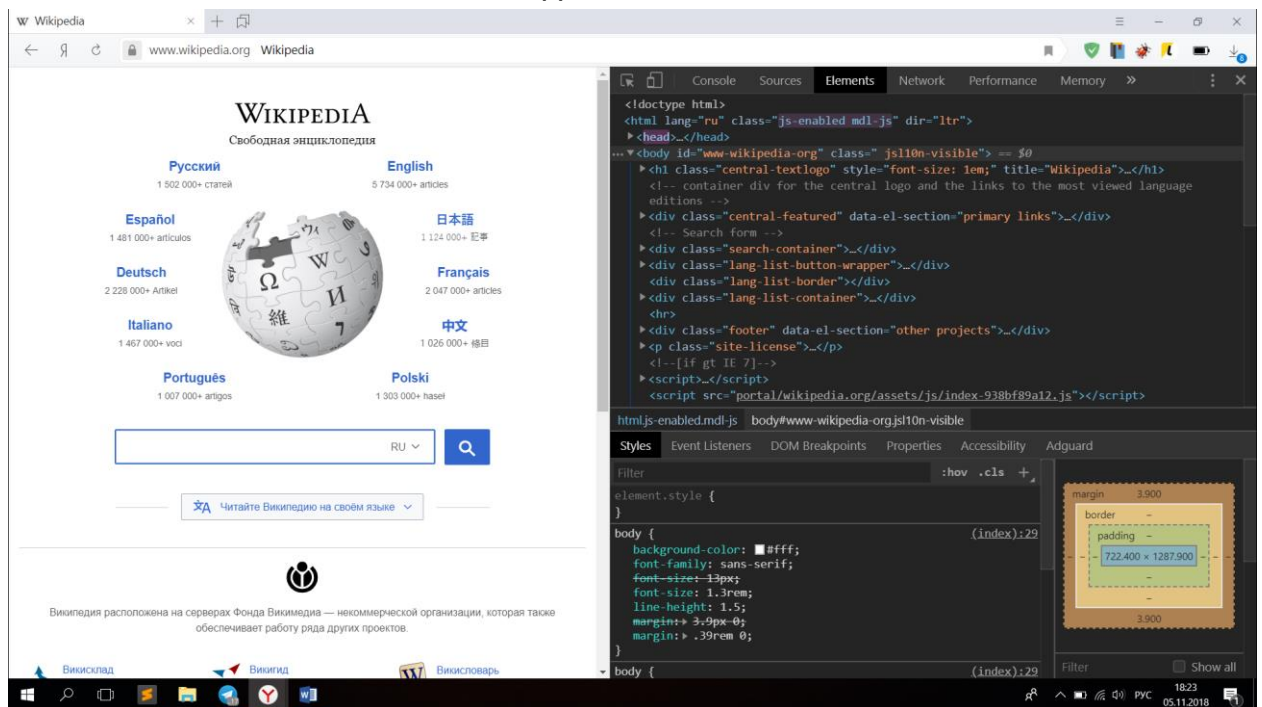


Изменим стили нашей ссылки



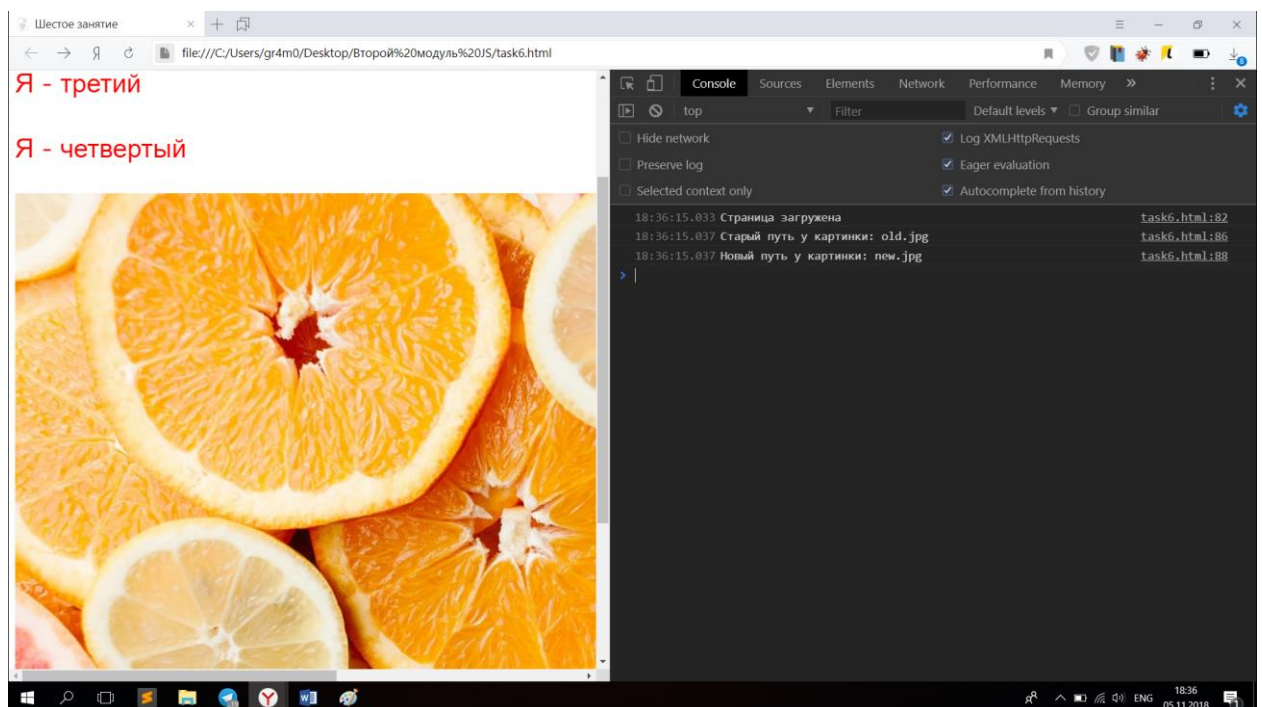


Наша ссылка полностью валидна



Более живой пример: ленивая загрузка изображений. Сначала загружаем маленькую размытую картинку, а после загрузки страницы подгружаем «хорошую».

```
74 // var link = document.createElement('a');
75 // link.textContent = 'Ссылка на википедию';
76 // link.setAttribute('href', 'https://wikipedia.org');
77 // link.style.cssText = ' font-size: 24px; \
78 // color: orange; \
79 // ';
80 // document.body.prepend(link);
81 document.addEventListener("DOMContentLoaded", function(event) {
82     console.log('Страница загружена');
83     var img = document.getElementById('image1');
84     if (img.hasAttribute('data-src')) {
85         var data = img.getAttribute('data-src');
86         console.log('Старый путь у картинки: ' + img.getAttribute('src'));
87         img.setAttribute('src', data);
88         console.log('Новый путь у картинки: ' + img.getAttribute('src'));
89         img.removeAttribute('data-src');
90     }
91 });
92
93 </script>
94 </body>
95 </html>
```



Что такое `addEventListener` мы разберем позднее. Сейчас нас интересует механизм замены атрибута:

- Проверили есть ли атрибут
- Получили его
- Изменили атрибут `src`
- Удалили атрибут `data-src`

Домашнее задание:

- Повторить весь изученный материал (на следующем занятии будет блиц-опрос)
- Промоделировать работу методов `append`, `prepend`, `before`, `after` и `replaceWith` на своих примерах
- Создать несколько ссылок и изменить у них `href` с помощью `setAttribute`