Лекция 6

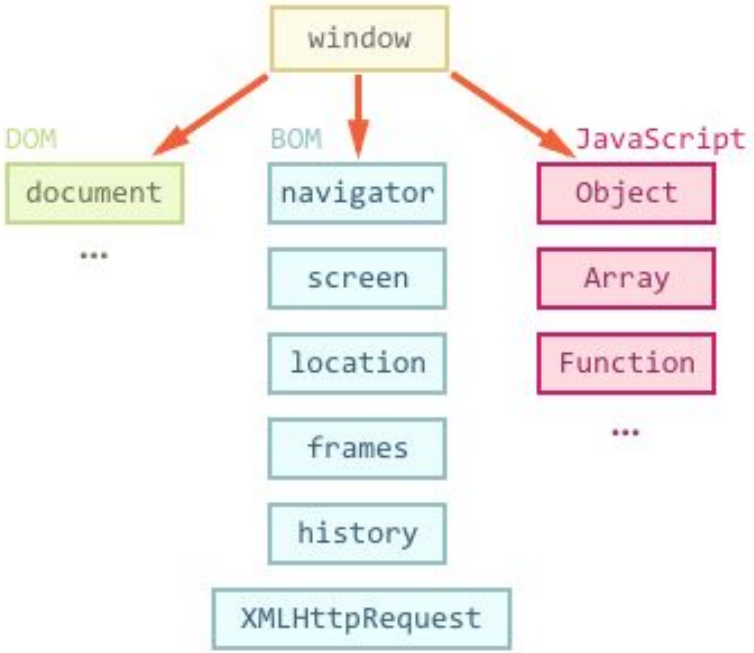
DOM

Что мы научимся делать?

- искать элементы;
- обращаться к элементам;
- менять содержимое HTML-документа;
- изменять стили;
- добавлять и удалять HTML-элементы;
- поговорим о шаблонах.

DOM (Document Object Model) --

это представление HTML-документа в виде дерева объектов, доступное для изменения через JavaScript. Из DOM легко можно управлять содержимым, стилями, значениями атрибутов.



Сам по себе язык JavaScript не предусматривает работы с браузером.

Он вообще не знает про HTML. Но позволяет легко расширять себя новыми функциями и объектами.

Мы будем работать с двумя типами узлов (нод):

- Теги образуют **узлы-элементы** (element node). Естественным образом одни узлы вложены в другие. Структура дерева образована исключительно за счет них.
- Текст внутри элементов образует **текстовые узлы** (text node). Текстовый узел содержит исключительно строку текста и не может иметь потомков, то есть он всегда на самом нижнем уровне.

Как найти элемент на странице?

HTML коллекции

- document.all;
- document.anchors;
- document.body;
- document.**documentElement**;
- document.embeds;
- document.forms;
- document.head;
- document.images;
- document.links;
- document.scripts;
- document.title.

Поиск по классу, тегу или идентификатору

`document.getElementById(`*`elementId: DOMString`*`)` --

принимает в качестве аргумента идентификатор и возвращает элемент.

`document.getElement``s`**`ByClassName(`*`classNames: DOMString`*`)`** --

принимает в качестве аргумента любой класс и возвращает коллекцию.

`document.getElementsByTagName(localName: DOMString)` --

принимает в качестве аргумента любой тег и возвращает коллекцию.

Коллекция -- не массив

Они похожи на массивы, поскольку имеют индексы и к ним можно обратиться, как к обычному массиву (`element[1]`, `form[2]`). Однако, у них **отсутствуют методы массивов** и для их перебора используют цикл ***for***.

Для перебора также нельзя использовать ***for … in*** , который кроме индексов будет выводить еще и лишнюю для нас информацию.

Крутые методы querySelectorAll и querySelector

`document.querySelectorAll(`*`selectors: DOMString`*`)` -- принимает в качестве аргумента любой CSS-селектор и возвращает коллекцию элементов.

`document.querySelector(`*`selectors: DOMString`*`)` -- принимает в качестве аргумента любой CSS-селектор и возвращает **первый найденный** на странице элемент. Аналог *`document.querySelectorAll('.someclass')[0]`*, однако работает быстрее.

matches и matchMedia

`window.matchMedia(`*`query: string`*`)` --
принимает в качестве аргумента любой CSS-селектор и возвращает колекцию элементов.

`document.matches(`*`selectors: DOMString`*`)` --
проверяет, удовлетворяет ли заданный элемент указанному селектору. Возвращает true или false и бывает очень полезен при переборе элементов и в связке с **matchMedia( )**.

Как влиять на HTML?

Как писать в документе?

`document.write(text...: DOMString)` --

дописывает текст в документ, пока он еще не загружен. В качестве аргумента можно передать любой текст, на страницу он вставляется, как есть. Является небезопасным и используется очень редко.

`document.getElementById(id).innerHTML` --

позволяет получить содержимое элемента в виде строки. Этим же методом строку можно присвоить указанному элементу. Метод является безопасным, поскольку браузер автоматически исправляет ошибки.

Как получать и изменять значения атрибута?

```
document.getElementById(id).attribute  --
```

позволяет получить значение атрибута.

```
document.getElementById(id).attribute  = new value; --
```

изменяет значение атрибута.

Как получать и изменять значения атрибута?

```
document.getElementById(id).hasAttribute(name: DOMString);

document.getElementById(id).removeAttribute(name:
DOMString)

document.getElementById(id).getAttribute(name: DOMString)

document.getElementById(id).setAttribute(name: DOMString,
value: DOMString)
```

Как влиять на CSS?

Как получать и изменять значения стилей?

`document.getElementById(id).style.property` --

возвращает строку со значением указанного свойства. Таким образом можно и получать и задавать стили.

Если свойство состоит из нескольких слов, записанных через дефис, то к нему обращаются, используя *camelCase (backgroundColor, borderRadius, webkitBoxSizing).*

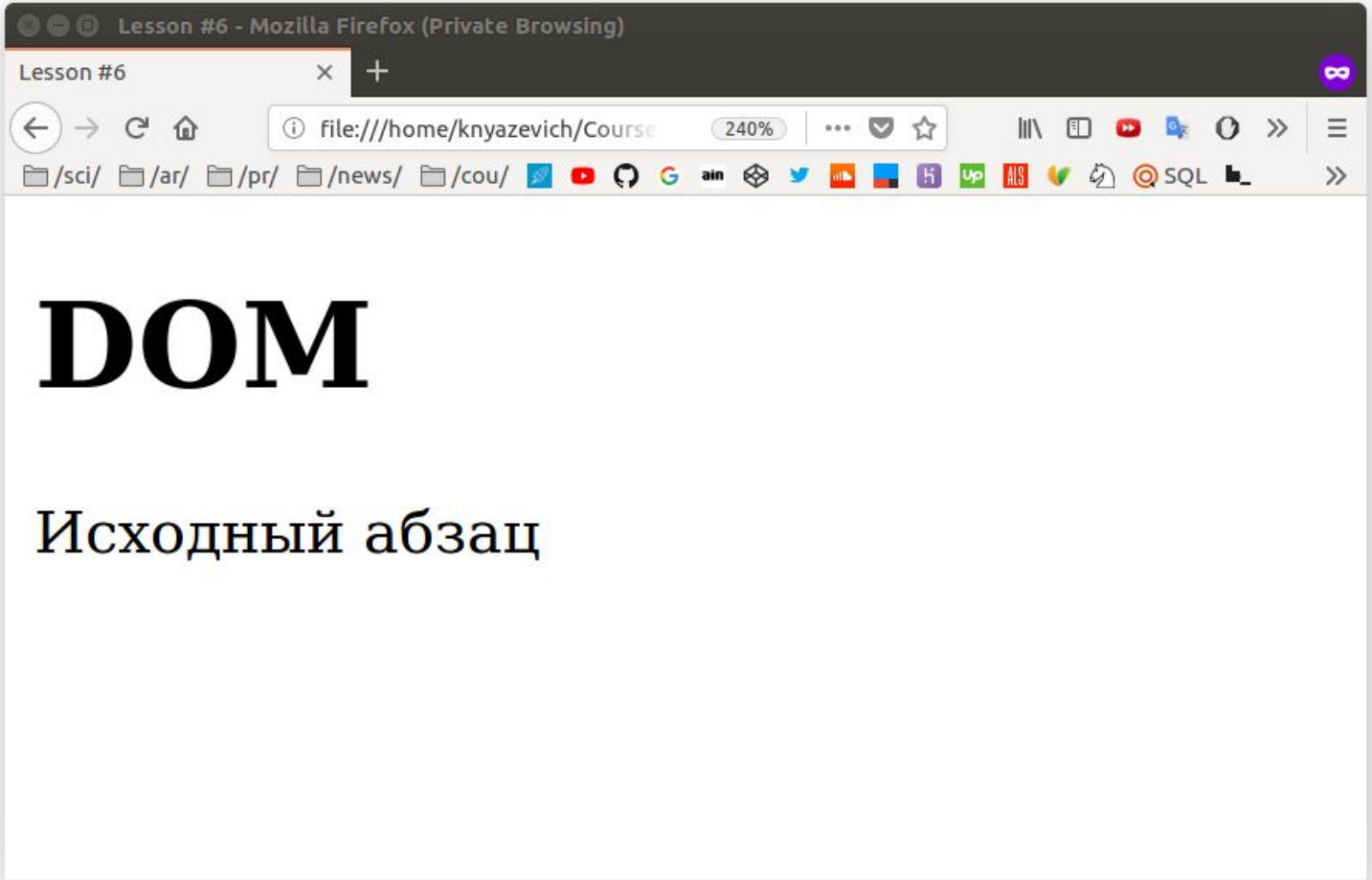
Как добавлять и удалять элементы?

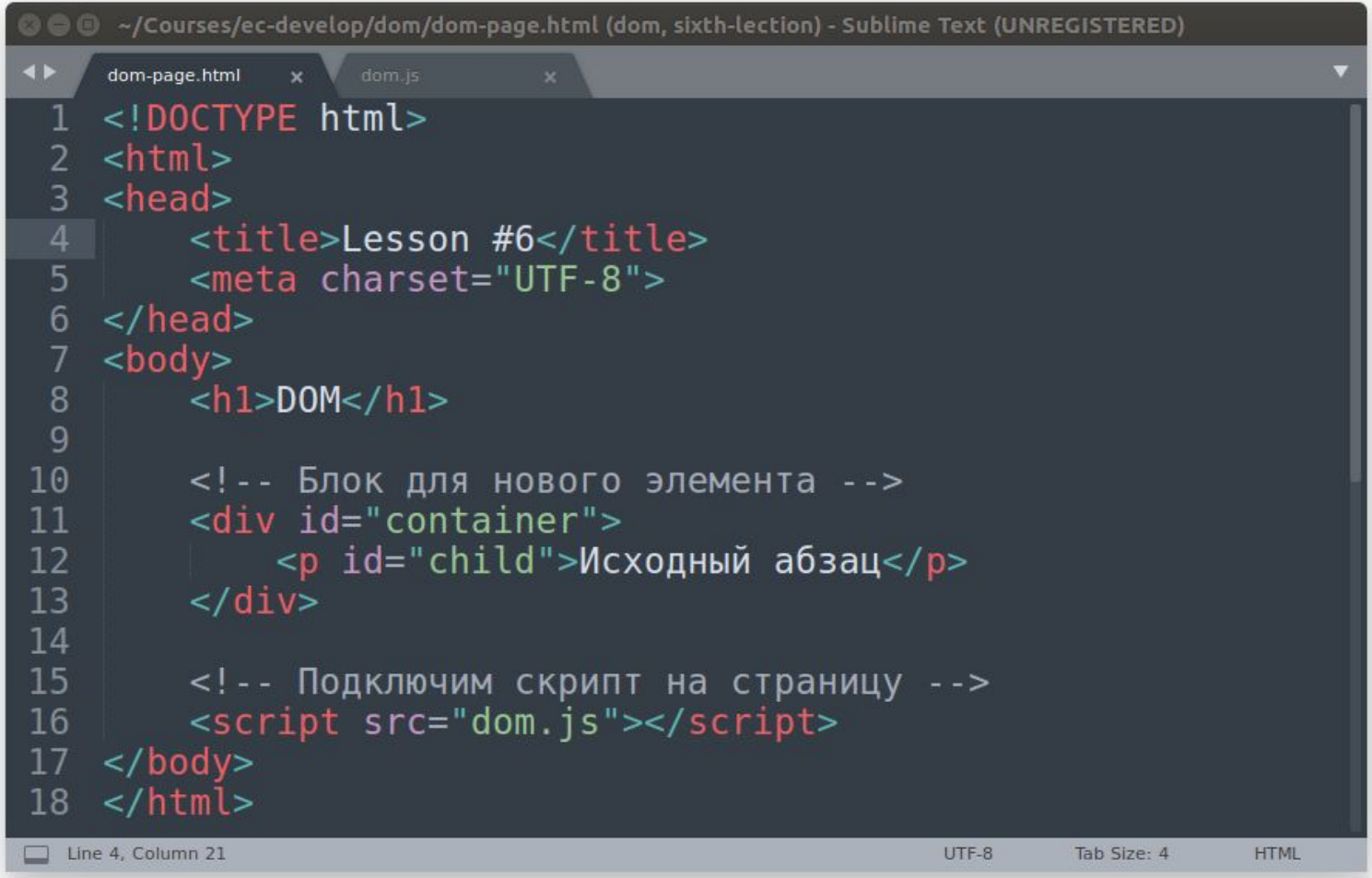
Крутые методы querySelectorAll и querySelector

`document.querySelectorAll(`*`selectors: DOMString`*`)` -- принимает в качестве аргумента любой CSS-селектор и возвращает коллекцию элементов.

`document.querySelector(`*`selectors: DOMString`*`)` -- принимает в качестве аргумента любой CSS-селектор и возвращает **первый найденный** на странице элемент. Аналог *`document.querySelectorAll('.someclass')[0]`*, однако работает быстрее.

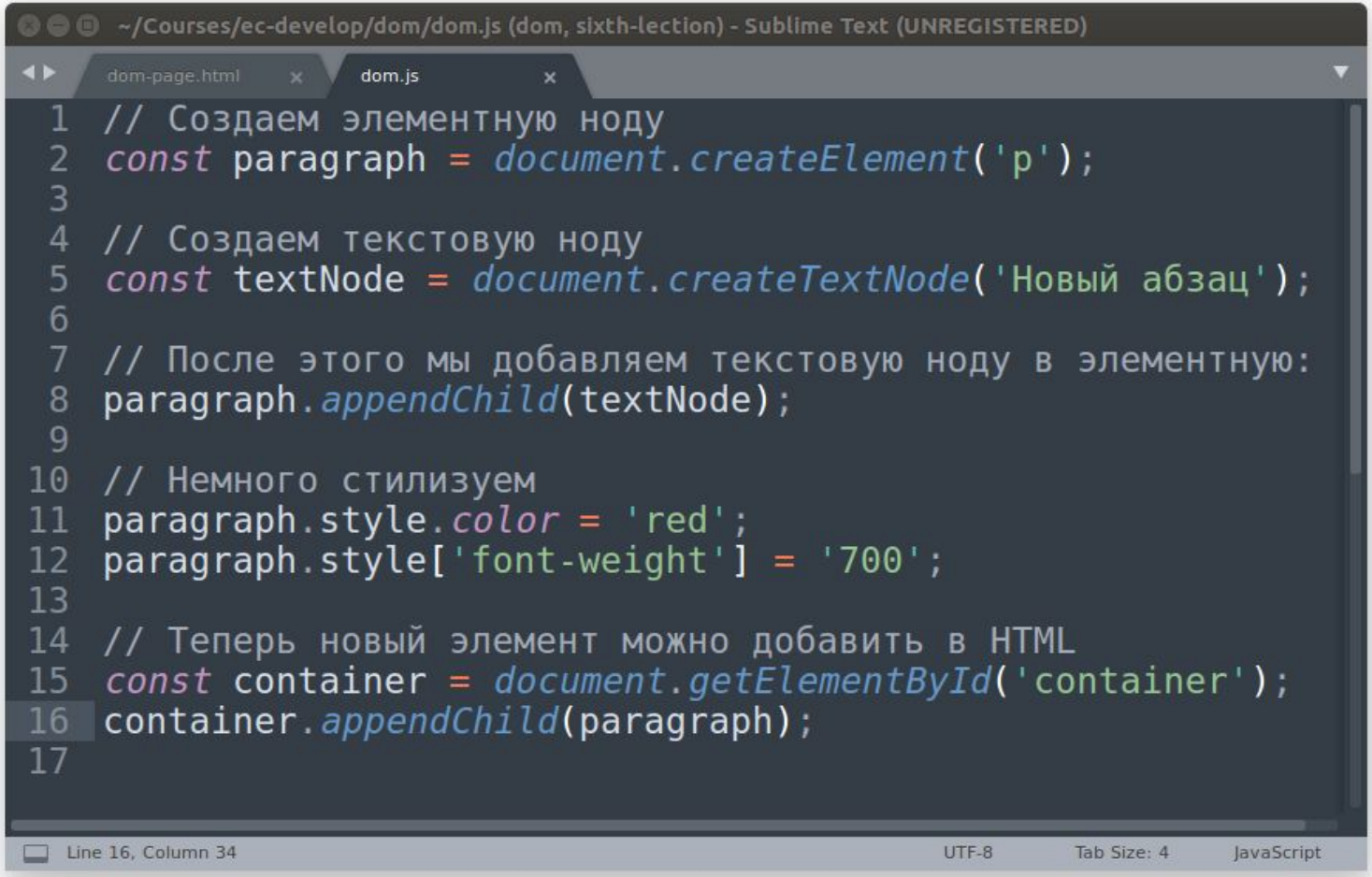
Добавляем и удяляем ноды

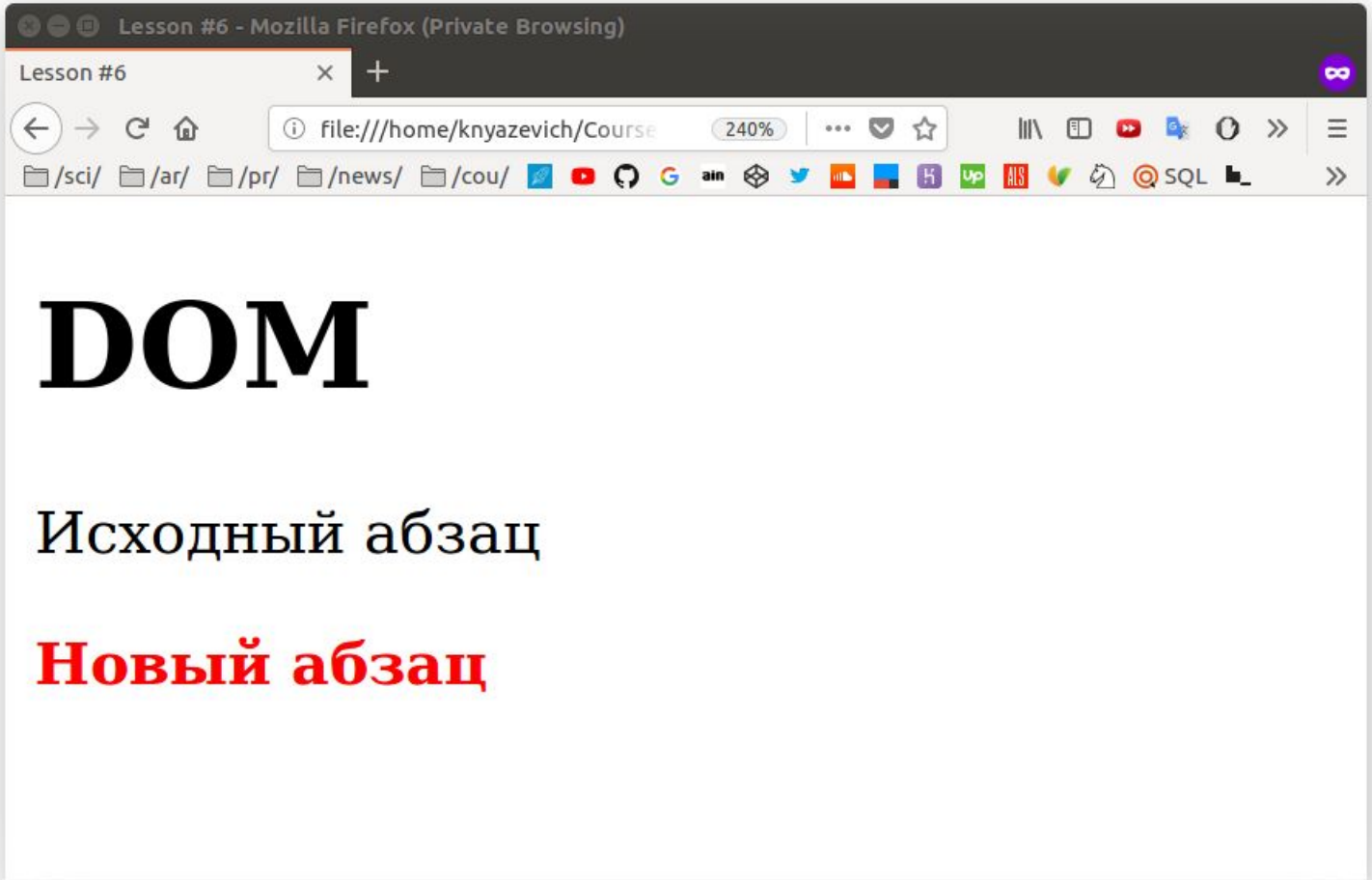

Lesson #6 - Mozilla Firefox (Private Browsing)
Lesson #6
file:///home/knyazevich/Course
240%
/sci/
/ar/
/pr/
/news/
/cou/
SQL
DOM
Исходный абзац


~/Courses/ec-develop/dom/dom-page.html (dom, sixth-lection) - Sublime Text (UNREGISTERED)
dom-page.html
dom.js
<!DOCTYPE html>
<html>
<head>
<title>Lesson #6</title>
<meta charset="UTF-8">
</head>
<body>
<h1>DOM</h1>
<!-- Блок для нового элемента -->

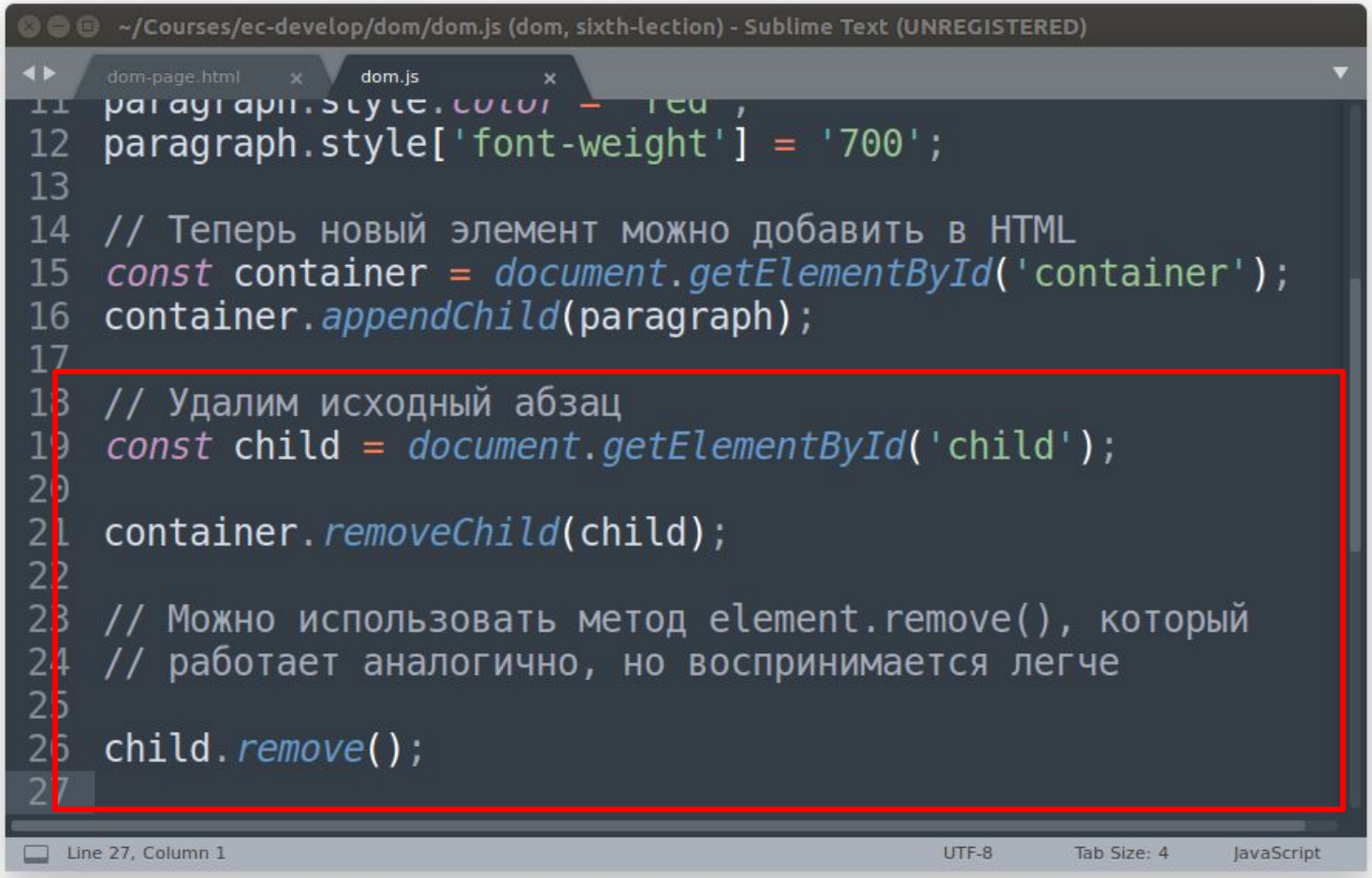
Исходный абзац

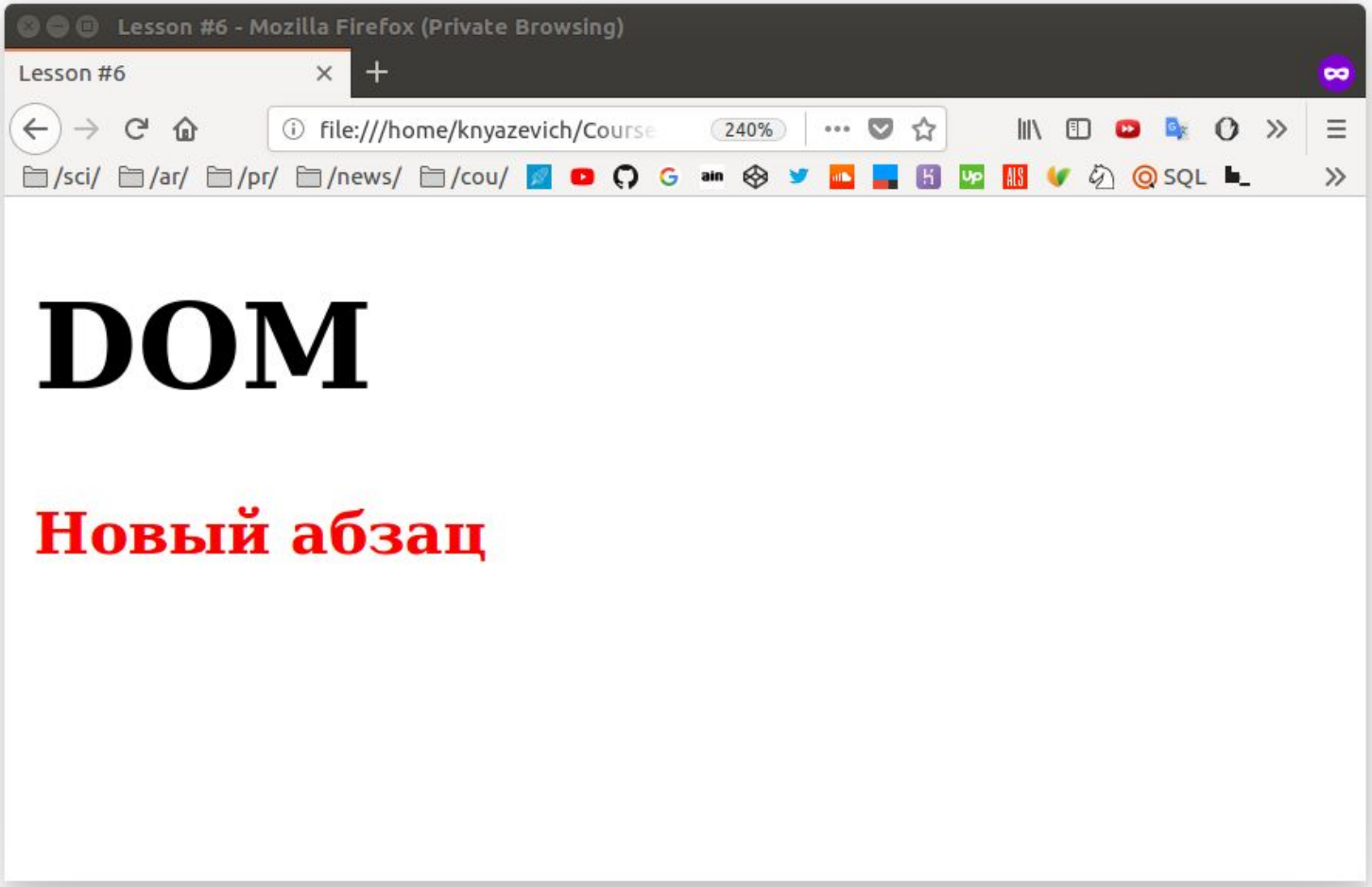
<!-- Подключим скрипт на страницу -->
<script src="dom.js"></script>
</body>
</html>
Line 4, Column 21
UTF-8
Tab Size: 4
HTML


~/Courses/ec-develop/dom/dom.js (dom, sixth-lection) - Sublime Text (UNREGISTERED)
dom-page.html
dom.js
// Создаем элементную ноду
const paragraph = document.createElement('p');
// Создаем текстовую ноду
const textNode = document.createTextNode('Новый абзац');
// После этого мы добавляем текстовую ноду в элементную:
paragraph.appendChild(textNode);
// Немного стилизуем
paragraph.style.color = 'red';
paragraph.style['font-weight'] = '700';
// Теперь новый элемент можно добавить в HTML
const container = document.getElementById('container');
container.appendChild(paragraph);
Line 16, Column 34
UTF-8
Tab Size: 4
JavaScript


Lesson #6 - Mozilla Firefox (Private Browsing)
Lesson #6
file:///home/knyazevich/Course
240%
/sci/
/ar/
/pr/
/news/
/cou/
SQL
DOM
Исходный абзац
Новый абзац

Удаляем элемент


~/Courses/ec-develop/dom/dom.js (dom, sixth-lection) - Sublime Text (UNREGISTERED)
dom-page.html
dom.js
paragraph.style.color = 'red';
paragraph.style['font-weight'] = '700';
// Теперь новый элемент можно добавить в HTML
const container = document.getElementById('container');
container.appendChild(paragraph);
// Удалим исходный абзац
const child = document.getElementById('child');
container.removeChild(child);
// Можно использовать метод element.remove(), который
// работает аналогично, но воспринимается легче
child.remove();
Line 27, Column 1
UTF-8
Tab Size: 4
JavaScript


Lesson #6 - Mozilla Firefox (Private Browsing)
Lesson #6
file:///home/knyazevich/Course
240%
/sci/
/ar/
/pr/
/news/
/cou/
SQL
DOM
Новый абзац

Шаблоны

Шаблон --

это строка в специальном формате, которая путём подстановки значений (текст сообщения, цена, товар и т.п.) и выполнения встроенных фрагментов кода превращается в DOM/HTML.