Лекция № 5. Работа с элементами

УЧЕБНЫЕ ВОПРСЫ

1. Получение и установка контента элементам в JavaScript
2. Работа с атрибутами и свойствами элементов в JavaScript

Вопрос №1. Получение и установка контента элементам в JavaScript

Свойство textContent

textContent – это свойство, которое предназначено для работы с текстовым контентом элемента. Оно позволяет его как **получить** (включая текстовое содержимое всего его потомков), так и **установить**.

Синтаксис

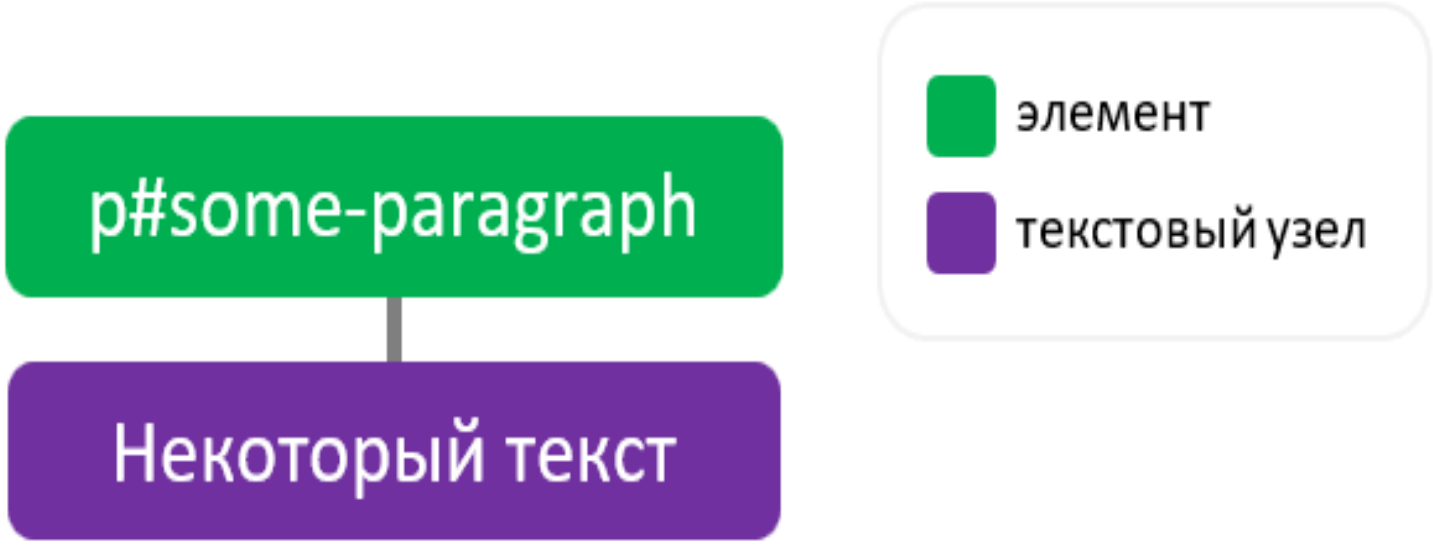
```
// $elem – некоторый DOM-элемент

// получим текстовый контент $elem
const text = $elem.textContent;
// установим текстовый контент $elem:
$elem.textContent = 'Некоторый текст...';
```

Свойство textContent

1. При получении текста элемента, содержащего **один текстовый узел**, textContent возвратит текст, находящийся внутри этого текстового узла.

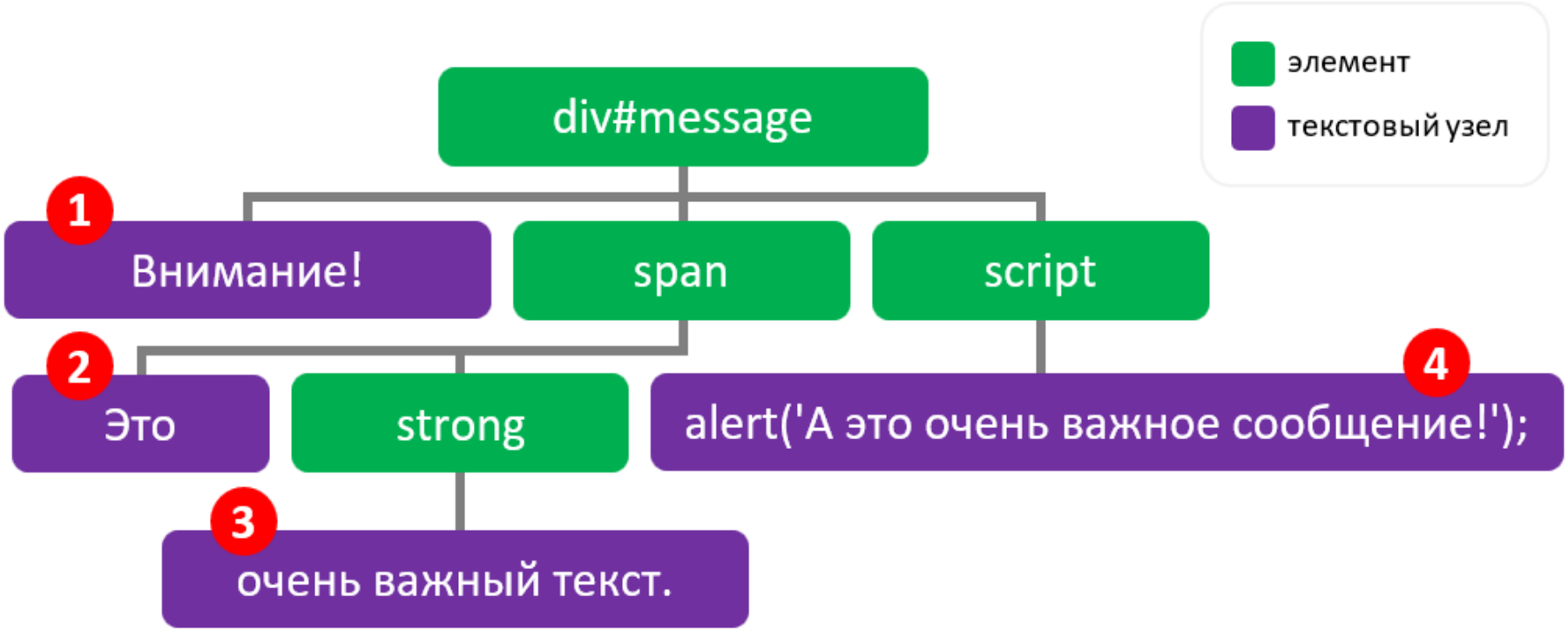
Пример:



Свойство textContent

2. Для элемента, который содержит **множество** других узлов, textContent вернёт **конкатенацию** (сложение) текстов всех его текстовых узлов.

Пример:



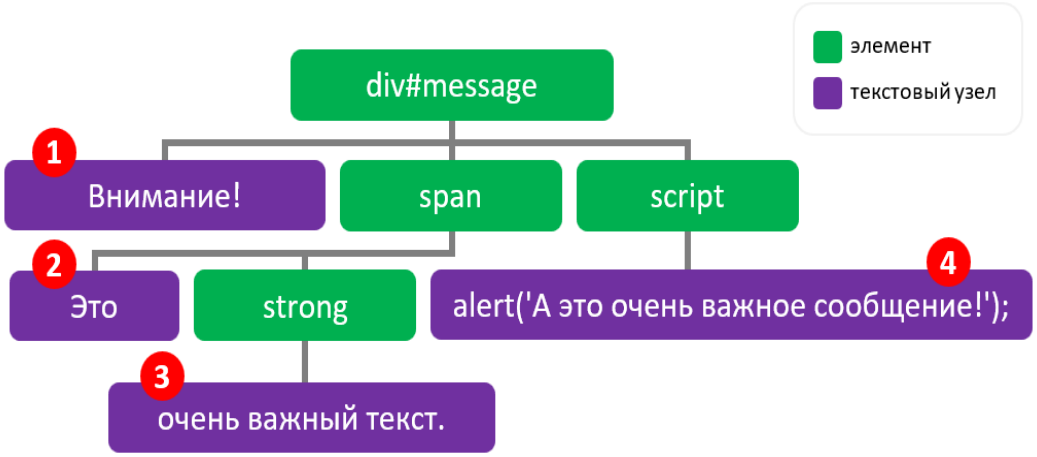
Свойство textContent

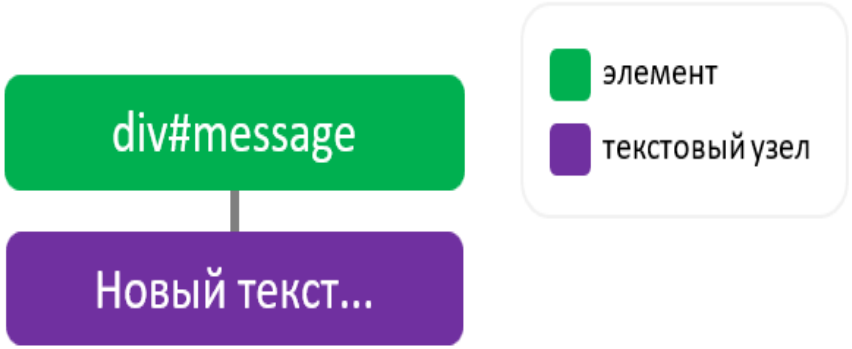
3. При получении textContent у **document**, оно возвратит **null**

Свойство textContent

4. При **установке** элементу текстового содержимого, textContent **удалит** всего его узлы (при их наличии), и добавит в него **один текстовый узел**, содержащий указанный текст.

Пример:





Свойство textContent

5. Если присвоить textContent строку, содержащую HTML код, то символы < и > будут заменены соответственно на < и >

Свойство innerText

innerText также как textContent используется для **извлечения текста** из элементов.

Но innerText в отличие от textContent как бы **копирует текст**, отображаемый этим элементом в браузере. Он **учитывает стили**, применённые к элементу (отображается элемент или нет). Когда элемент скрыт, innerText не включает его текстовый контент в возвращаемые данные.

Свойство innerText

Кроме этого, innerText **не добавляет** в возвращаемый результат содержимое style и script.

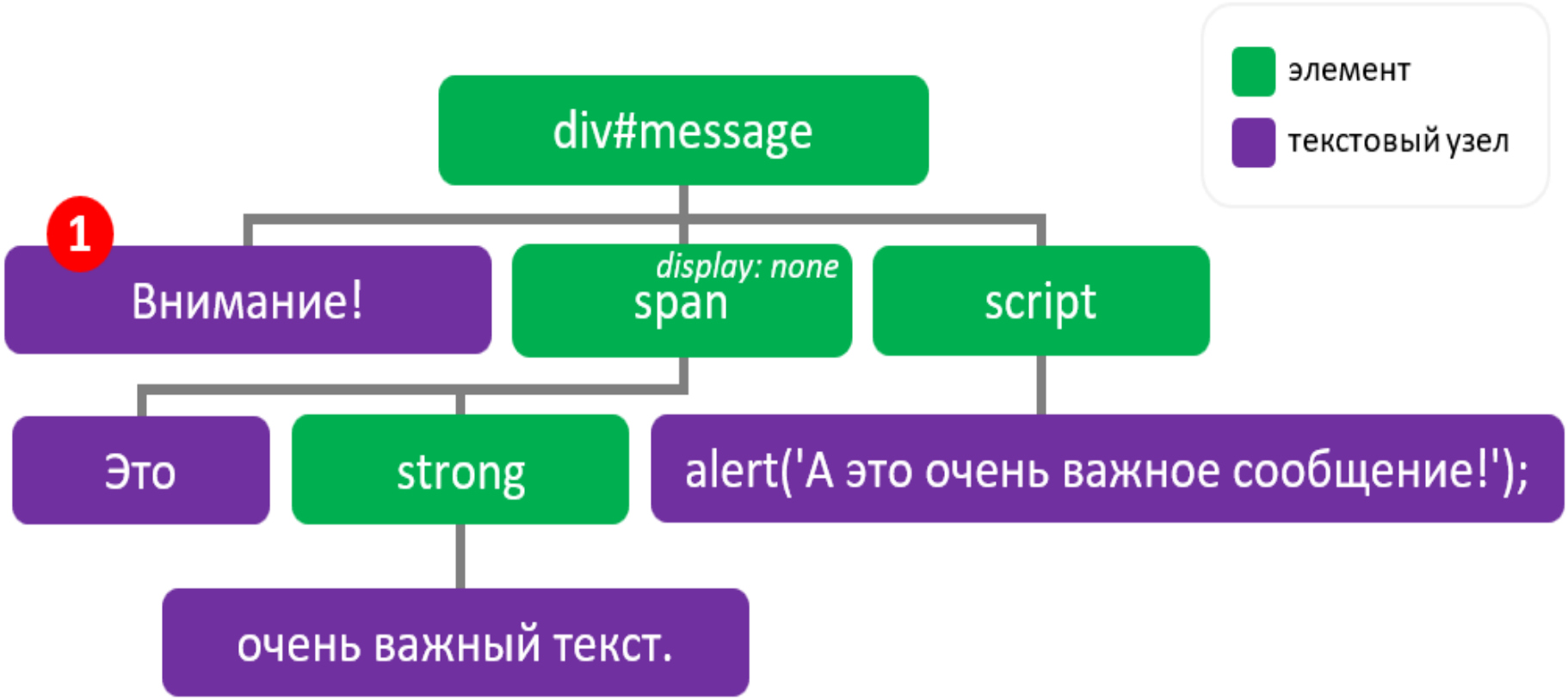
При установке элементу текстового контента, innerText также как textContent **удаляет все** имеющиеся в нём узлы и создаёт новый текстовый узел с указанным содержимым.

Синтаксис свойства **innerText**:

```
// получим текстовый контент $elem
const text = $elem.innerText;
// зададим $elem текстовый контент
$elem.innerText = 'Новый текст...';
```

Свойство innerText

Пример:



Свойство outerText

Свойство возвращает текстовое содержимое элемента аналогично свойству innerText.

Его **отличие** от innerText только в том, что outerText при установки элементу текстового контента **удаляет** не только всё его содержимое, но и **сам этот элемент** и помещает на этом месте **новый текстовый узел** с заданным текстом.

Синтаксис свойства outerText:

```
// получим текстовый контент $elem
const text = $elem.outerText;
// установим $elem текстовый контент
$elem.outerText = 'Текстовый контент...';
```

Свойство innerHTML

innerHTML предназначен для **установки** или **получения** HTML разметки элемента.

Синтаксис:

```
// получим HTML содержимое $elem
const html = $elem.innerHTML;
// установим $elem новый HTML
$elem.innerHTML = '<div>...<div>';
```

Свойство innerHTML

Задание HTML содержимого элементу с помощью innerHTML всегда сопровождается **удалением его контента** и установкой ему **новой HTML разметки**, но основе указанной строки, которая была разобрана внутренним парсером браузера как HTML. Выполнение такого кода обычно сопровождается «миганием».

Поэтому, в ситуациях, когда вам нужно **просто добавить** некоторый фрагмент HTML разметки в некоторый элемент лучше воспользоваться, например, методом **insertAdjacentHTML**.

Свойство innerHTML

Пример использования innerHTML для очистки содержимого элемента

```
$elem.innerHTML = '';
```

Свойство outerHTML

Свойство outerHTML устанавливает или возвращает HTML контент, представляющий **сам элемент** и его **дочерние элементы**

Вопрос №2. Работа с атрибутами и свойствами элементов в JavaScript

Атрибуты — это HTML-сущности, с помощью которых мы можем добавить определённые данные к элементам в HTML-коде.

Когда браузер запрашивает некоторую страницу, он получает её исходный HTML-код. После этого он парсит этот код и строит на его основании DOM. Во время этого процесса HTML-атрибуты элементов **переводятся** в соответствующие DOM-свойства.

Например, браузер, при чтении следующей HTML-строчки кода, создаст для этого элемента следующие DOM-свойства: id, className, src и alt.

<img id="brand" class="brand" src="logo.png" alt="логотип сайта">

Обращение к этим свойствам в коде JavaScript выполняется как к свойствам объекта. Объектом здесь выступает узел (элемент) DOM.

Некоторые названия DOM-свойств **не соответствуют** именам атрибутов. Одним из таких является **атрибут class**. Данному атрибуту соответствует **DOM-свойство className**.

Данное отличие связано с тем, что class является ключевым словом в JavaScript, оно зарезервировано и не может использоваться. Из-за этого разработчики стандарта решили использовать для соответствия какое-то другое название, в качестве которого было выбрано className.

Кроме того перевод HTML-атрибутов, заданных в исходном коде документа, в DOM-свойства не всегда осуществляется **один к одному**.

Если элемент **имеет нестандартный** HTML-атрибут, то свойство, соответствующее ему в DOM, **не создаётся**.

Другое отличие связано с тем, что **значения** определённых **HTML-атрибутов** и соответствующих им **DOM-свойств** могут быть **различными**. Т.е. атрибут может иметь *одно значение*, а DOM-свойство, созданное на его основе – *другое*.

Одним из таких атрибутов является **checked**.

Основные отличия между DOM-свойствами и атрибутами:

- значение атрибута – это всегда строка, а значение DOM-свойства – определённый тип данных (не обязательно строка);
- имя атрибута – регистронезависимо, а DOM-свойства - регистрозависимо.

Работа с DOM-свойствами элемента

Работа со свойствами элементов в JavaScript как уже было отмечено выше осуществляется как со **свойствами объектов**.

В качестве **примера** рассмотрим следующий HTML-элемент:

```
<div id="alert" class="alert alert-info" title="Текст подсказки...">
  Текст информационного сообщения...
</div>
```

Работа атрибутами элемента

Атрибуты изначально задаются в HTML-коде. Они хоть и связаны, некоторым образом, со свойствами, но это не одно и тоже. В большинстве случаев следует работать именно со свойствами, а к атрибутам обращаться только тогда, когда это действительно нужно.

Значения атрибутов, в отличие от DOM-свойств, как это уже было отмечено выше **всегда является строкой**.

Работа атрибутами элемента

В JavaScript для выполнения операций, связанных с атрибутами, имеется четыре метода:

- **.hasAttribute**('имя_атрибута') – проверяет наличие указанного атрибута у элемента. Если проверяемый атрибут есть у элемента, то данный метод возвращает true, в противном случае - false.
- **.getAttribute**('имя_атрибута') – получает значение атрибута. Если указанного атрибута нет у элемента, то данный метод возвращает пустую строку ("") или null.

Работа атрибутами элемента

В JavaScript для выполнения операций, связанных с атрибутами, имеется четыре метода:

- **.setAttribute**('имя_атрибута', 'значение_атрибута') – устанавливает указанный атрибут с указанным значением элементу. Если указанный атрибут есть у элемента, то данный метод тогда просто изменит ему значение.
- **.removeAttribute**('имя_атрибута') - удаляет указанный атрибут у элемента.

Особенности работы с некоторыми атрибутами

– Изменение DOM-свойства **value** не приводит к соответствующему изменению атрибута **value;**

– В атрибуте **href** содержится то, что мы установили в коде HTML, а в DOM-свойстве - полный URL.

– Наличие атрибута **checked** у HTML-элемента **input** с типом **type="checkbox"** означает, что **свойство checked** будет равно **true**, а значение атрибута – пустая строка.