Модифікація DOM-дерев. CSS-стилі

№ уроку:	3	**Курс:**	JavaScript Базовий

Засоби навчання:	Visual Studio Code Web Browser

Огляд, мета та призначення уроку

Навчитися створювати та додавати елементи в DOM-дерево. Вивчити способи клонування і видалення вузлів. Навчитися використовувати CSS-класи та редагувати окремі стилі елементів через JavaScript.

Вивчивши матеріал цього заняття, учень зможе:

- Створювати елементи.
- Використовувати різні методи для вставлення елементів у DOM.
- Видаляти елементи.
- Використовувати глибоке та поверхневе клонування.
- Додавати та видаляти CSS-класи для елементів.
- Працювати зі стилями через властивість style.
- Використовувати обчислені стилі.

Зміст уроку

1. Способи створення вузлів у DOM-дереві.
2. Вставлення та видалення вузлів.
3. Способи клонування вузлів DOM-дерева.
4. Властивості й атрибути елементів.
5. Методи роботи з атрибутами.
6. Робота з користувацькими атрибутами data-*.
7. Використання CSS-стилів, властивості className та classList.
8. Обчислені стилі та робота з ними.

Резюме

- **document.createElement(тег)** – створення нового елемента, який вказаний у параметрах. Як параметр передається рядок, який визначає створюваний елемент. Наприклад, **document.createElement(–div||)** створює в оперативній пам'яті об'єкт, який є еквівалентом використання .
- **documetn.createTextNode(текст)** – створення текстового вузла. Є еквівалентом використання літерала (звичайного тексту) у тілі документа. Значення в елементі, який динамічно створений, можна створити за допомогою цього методу або використання властивостей textContent або innerHTML.

- **node.append(новий_вузол)** – додати елемент новий_вузол як останній дочірній елемент для node.

CyberBionic Systematics ®
19 Evgenia Sverstiuka Str., 5 floor
Kyiv, Ukraine

Tel.:0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- **node.prepend(новий_вузол)** – додати елемент новий_вузол як перший дочірній елемент для node.
- **node.before(новий_вузол)** – додати елемент новий_вузол як вузол перед елементом node.
- **node.after(новий_вузол)** – додати елемент новий_вузол як вузол після елемента node.
- **node.replaceWith(новий_вузол)** – замінює node на вузол, який вказаний у параметрі новий_вузол.
- Щоб видалити вузли зі сторінки, на вузлі необхідно викликати метод **remove().**
- У DOM-дереві може бути лише один екземпляр вузла. Якщо вузол, який вже розміщений у дереві, перенести в іншу частину дерева за допомогою методів вставлення, то в старій позиції вузол буде видалено, а потім додано в нову.
- За допомогою методу **cloneNode** можна створити копію вузла. Можна створити поверхневу (передання значення false у параметр) або глибоку копію (передання значення true як параметр).
- **Поверхнева копія** – копія, що містить стан самого елемента, але не додає дочірні елементи та їхній стан (атрибути за поверхневого клонування також копіюються).
- **Глибока копія** – те саме, що й поверхнева копія, але клонування відбувається всього графа об'єктів – зазначеного вузла та всіх дочірніх вузлів.
- На початку цього документа перераховані сучасні методи роботи з DOM. Браузери підтримують й інші методи, які є ранніми версіями методів, що підтримуються браузерами. Нові методи відрізняються додатковою гнучкістю.
- **Приклади альтернативних методів роботи з DOM:**
  parent.appendChild(node);
  parent.insertBefore(node, nextSibling);
  parent.removeChild(node);
  parent.replaceChild(newElem, node).
- **Вузол DOM** – звичайний об'єкт JavaScript. Для DOM-вузлів можна задавати нові властивості та методи.

- Під час парсингу HTML-розмітки на основі тегів створюються об'єкти, а значення **стандартних** атрибутів присвоюються відповідним властивостям створених об'єктів.

- Якщо атрибут не є стандартним, його не можна отримати за допомогою властивості. Наприклад, атрибут type не є стандартним для div, його можна задати в розмітці, але отримати його значення через властивість type не вийде (буде отримано значення undefined). Такі атрибути можуть бути отримані за допомогою спеціальних методів, які описані нижче.

- **Методи роботи з атрибутами:**
- element.hasAttribute(<ім'я_атрибута>); – перевірка наявності атрибута.
- element.getAttribute(<ім'я_атрибута>); – отримання значення атрибута.
- element.setAttribute(<ім'я_атрибута>, <значення_атрибута>); – встановлення значення атрибута.
- element.removeAttribute(<ім'я_атрибута>); – видалення атрибута.

- Ці методи працюють із вмістом HTML-розмітки, вони не читають значення властивостей, а саме той контент, який розташований на сторінці як атрибут елемента.

- **Атрибути HTML мають дві особливості:**
  1. Їхній вміст – рядкове значення.
  2. Імена атрибутів незалежні від реєстру, наприклад, ID та id – це один і той самий атрибут.

CyberBionic systematics
CyberBionic Systematics ®
19 Evgenia Sverstiuka Str., 5 floor
Kyiv, Ukraine
Tel.:0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Між атрибутами та властивостями відбувається **автоматична синхронізація**. Якщо змінити атрибут, зміниться значення властивості. Якщо змінити властивість, зміниться значення атрибута.

- Винятком є властивість та атрибут value для елемента input. Якщо змінити атрибут, значення зміниться. Але якщо змінити властивість, атрибут не зміниться. Це може бути зручним способом одержання вихідного значення елемента. Дії користувача змінюють значення властивості value, але значення атрибута залишається вихідним.

- Властивості DOM *типізовані*, наприклад властивість checked елемента input типу Boolean. Властивість style – це об'єкт, який дає змогу працювати з оформленням елемента.

- Не бажано використовувати свої атрибути з довільним ім'ям, оскільки це ім'я у майбутніх версіях специфікації може стати стандартним атрибутом елемента. Для цілей створення користувацьких атрибутів зарезервований спеціальний префікс **data-**. Атрибути, які створені з допомогою цього префікса, розглядаються як користувацькі, створені розробником. Наприклад, data-status, data-custom-message – користувацькі атрибути.

- **Атрибути data-*** доступні через властивість елемента **dataset**. Через dataset можна як прочитати значення атрибута, так і змінити.

- Якщо атрибут складається з кількох властивостей, у dataset він потрапляє перейменованим з використанням camelCasin, наприклад, data-report-month dataset буде властивістю *reportMonth*.

- Зазвичай нестандартні атрибути використовуються для передання даних у JavaScript-код.

- **CSS-стилі** – найкращий варіант оформлення елементів на сторінці. Намагайтеся уникати використання атрибута style та властивості style у коді JavaScript, через які можна змінювати значення окремих CSS-властивостей.
  Використання класів замість окремих стилів дає перевагу у продуктивності та простоті подальшого використання. Оформлення зосереджено у CSS-файлі, тому редагування зовнішнього вигляду зачіпає лише редагування цього файлу, а не розбір логіки JavaScript-застосунку.

- Через JavaScript можна змінити як окремий стиль, так і клас, який застосовується для елемента.

- За допомогою властивості **className** можна прочитати або змінити класи, які використовуються елементом. Ця властивість дає доступ до атрибута елемента class. Якщо надати властивості значення, то наявні класи, які вказані в атрибуті class, будуть перезаписані.

- Властивість **classList** дає гнучкіший інтерфейс для роботи з CSS-класами елемента. Ця властивість є об'єктом, який надає метод для додавання, видалення, перевірки наявності класу в елементі.

    - add – додати клас до вже наявних;
    - remove – видалити клас;
    - toggle – додати, якщо класу немає, видалити, якщо клас є; contains – перевірка наявності вказаного класу у списку класів.

CyberBionic Systematics ®
19 Evgenia Sverstiuka Str., 5 floor
Kyiv, Ukraine
Tel.:0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Також classList можна перебрати за допомогою циклу for … of.

- Щоб працювати з окремими CSS-властивостями елемента, використовується **властивість style**. Властивість представляє об'єкт, який відповідає значенням, що вказані в атрибуті style HTML-елементам.
  Імена властивостей в об'єкті style відповідають іменам CSS-властивостей. Якщо в CSS властивість містить дефіс, то в JavaScript для цієї властивості використовується **camel Casing**. Якщо властивість використовує префікс, наприклад, -moz або -webkit, назва такої властивості пишеться з великої літери.

CSS-властивості	*Властивість у JavaScript*
background-color	backgroundColorfont-family fontFamily
-moz-radial-gradient	MozRadialGradient

- **Обчислені стилі** – стилі, які отримані елементом після застосування всіх CSS-правил і дозволу всіх геометричних властивостей – висоти, ширини тощо.
  Для отримання обчислених стилів використовується метод **getComputedStyles** (елемент, псевдоелемент).
  Наприклад, якщо потрібно дізнатися ширину елемента, для якого явно не було вказано значення властивості width, читання властивості style.width не дасть результату. Оскільки ширину елемента обчислював браузер, щоб дізнатися про поточний розмір, необхідно використовувати метод **getComputedStyles**.

Закріплення матеріалу

- Як динамічно додати об'єкт на сторінку?
- Які є методи для розміщення створеного вузла в DOM-дереві?
- Як видалити вузол зі сторінки?
- Що буде, якщо параграф, який розміщений на початку сторінки, знайти та додати наприкінці документа?
- Що таке поверхнева копія? Що таке глибока копія?
- Як зіставляються елементи та атрибути в HTML з об'єктами?
- Перерахуйте методи роботи з атрибутами.
- Який елемент та який атрибут не синхронізуються, коли відбувається зміна значення властивості, відповідний атрибут не змінюється?
- Для чого використовуються атрибути з префіксом data-*?
- Як можна отримати доступ до атрибута data-my-attribte?
- Чому краще використовувати CSS-класи, а не окремі стилі CSS?
- У чому різниця між властивостями className та classList?
- Як імена CSS-властивостей, де є дефіси, використовуються разом із властивістю style?
- Як написати код, який додасть CSS-клас, якщо в елементі такого класу немає, або видалить CSS-клас, якщо в елементі такий клас вже є?
- Що таке обчислені стилі та як їх отримати?

Самостійна діяльність учня

Виконайте завдання у директоріях:

Tasks\003 DOM Tree Modification
Tasks\004 Props and Attributes
Tasks\005 Styles

Текст завдань розташований у

CyberBionic Systematics ®
19 Evgenia Sverstiuka Str., 5 floor
Kyiv, Ukraine
Tel.:0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

коментарях, у тегах script.

Рекомендовані ресурси

Створення елементів
https://developer.mozilla.org/en-US/docs/Web/API/Document/createElement

Додавання елементів у DOM
https://developer.mozilla.org/ru/docs/Web/API/ParentNode/append

Підтримка методів append та appendChild
https://caniuse.com/?search=append

setAttribute
https://developer.mozilla.org/ru/docs/Web/API/Element/setAttribute

getAttribute
https://developer.mozilla.org/ru/docs/Web/API/Element/getAttribute

Властивість dataset
https://developer.mozilla.org/ru/docs/Web/API/HTMLElement/dataset

Style, робота з інлайновими стилями
https://developer.mozilla.org/ru/docs/Web/API/HTMLElement/style

className
https://developer.mozilla.org/ru/docs/Web/API/Element/className

classList
https://developer.mozilla.org/ru/docs/Web/API/Element/classList

getComputedStyle
https://developer.mozilla.org/ru/docs/Web/API/Window/getComputedStyle

CyberBionic Systematics ®
19 Evgenia Sverstiuka Str., 5 floor
Kyiv, Ukraine

Tel.:0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com