Лабораторная работа №6.

Тема: Создание интерактивного контента с JavaScript.

Цель работы: изучить основы работы с JavaScript для добавления интерактивности на веб-страницу.

Основные термины и понятия:

JavaScript – это язык программирования, который можно использовать для управления и изменения содержимого веб-страницы в реальном времени. Он интерпретируется и исполняется javascript-движком, который встроен в веб-браузер пользователя.

Чтобы использовать JavaScript, его можно вставить прямо в HTML с использованием тега <script>. Также можно разместить JavaScript-код в отдельном файле и подключить его к HTML-документу:

```
<!DOCTYPE html>
<html>
<head>
  <title>Мой интерактивный сайт</title>
</head>
<body>
  <h1 id="example">Пример</h1>
  <script src="script.js"></script>
</body>
</html>
```

DOM (Document object model) – объектная модель документа (имеются в виду HTML-документы), представляющая собой древовидную структуру, узлами которой являются HTML-теги, а также API для управления этой структурой. **API (Application Program Interface)** – программный интерфейс приложения, предназначен для того, чтобы иметь возможность управлять приложением с помощью сторонних программ. В данном случае приложением является веб-браузер, а сторонней программой – программа на языке JavaScript.

В DOM API входят следующие **типы данных** (интерфейсы):

– Document – представляет корневой объект документа.

– Node – узел. Документ состоит из узлов, они в свою очередь подразделяются на узлы элементов, текстовые узлы и узлы атрибутов.

– Element – узел элемента (тега). Расширяет функционал Node. Также существуют более специфичные расширения самого Element: HTMLElement – элемент HTML-документа, HTMLTableElement – элемент для тега <table>.

– NodeList – массив узлов.

– Attr – узел атрибута.

– NamedNodeMap – неупорядоченная коллекция узлов-атрибутов. Позволяет получать доступ к Attr по индексу и по имени, также позволяет добавлять, изменять, удалять узлы-атрибуты.

– Text – текстовый узел.

Для работы с DOM определён глобальный объект document. Он предоставляет ряд свойств и методов для управления элементами веб-страницы.

Свойства объекта document:

– documentElement – предоставляет доступ к корневому элементу <html>;

– body – предоставляет доступ к элементу <body>;

– images – содержит коллекцию всех объектов изображений – элементов <img>;

– links – содержит коллекцию ссылок – элементов <a> и <area>, у которых определён атрибут href;

– anchors – предоставляет доступ к коллекции элементов <a>, у которых определён атрибут name;

– forms – содержит коллекцию всех форм.

Методы объекта document:

– getElementById(value) – предоставляет элемент, у которого атрибут id имеет значение value;

– getElementByTagName(value) – предоставляет элементы, у которых имя тега совпадает с value;

– getElementByClassName(value) – предоставляет элементы, которые имеют класс value (какое-либо значение из атрибута class совпадает с value);

– querySelector(value) – предоставляет первый элемент, который соответствует CSS-селектору value;

– querySelectorAll(value) – предоставляет все элементы, которые соответствуют CSS-селектору value;

– createElement(tagName) – создаёт и возвращает элемент (HTML-тег), имя тега передаётся в параметрах;

– createTextNode(text) – создаёт и возвращает текстовый узел, текст узла передаётся в параметрах.

Для взаимодействия с узлами, можно использовать следующие **свойства объекта типа Node**:

– childNodes – коллекция дочерних узлов;

– firstChild – первый дочерний узел текущего узла;

– lastChild – последний дочерний узел текущего узла;

– previousSibling – предыдущий элемент, который находится на одном уровне с текущим;

– nextSibling – следующий элемент, который находится на одном уровне с текущим;

– ownerDocument – корневой узел документа;

– parentNode – родительский узел, который непосредственно содержит текущий;

– nodeName – имя узла;

– nodeType – тип узла (1 – тег, 2 – атрибут, 3 – текст);

– nodeValue – значение узла (в виде текста).

Методы интерфейса Node:

– appendChild(newNode) – добавляет новый узел newNode в конец коллекции дочерних узлов;

– insertBefore(newNode, referenceNode) – добавляет новый узел newNode перед узлом referenceNode;

– cloneNode(true/false) – клонирует узел: если в параметр передан true, то узел клонируется со всеми дочерними узлами, если false, то нет;

– removeChild(childNode) – удаляет дочерний узел childNode;

– replaceChild(newNode, oldNode) – заменяет старый дочерний узел oldNode на новый newNode.

Для управления тегами можно использовать следующие **свойства объекта типа Element**:

– tagName – имя тега;

– innerText – можно использовать напрямую, без вызова методов, для получения и установки текстового содержимого элемента;

– textContent – аналогично innerText;

– innerHTML – можно использовать напрямую, без вызова методов, для получения или установки HTML-разметки;

– offsetWidth, offsetHeight – ширина и высота элемента в px, соответственно, с учётом границы;

– clientWidth, clientHeight – аналогично offsetWidth, offsetHeight, но без учёта границы;

– style – напрямую сопоставляется с атрибутом style (CSS-свойства указываются в camel case, а не в kebab case);

– className – напрямую сопоставляется с атрибутом class (старое значение атрибута удаляется, если присвоено новое);

– classList – свойство управляет атрибутом class с помощью методов: add(className) – добавляет класс в список классов, remove(className) – удаляет класс из списка классов, toggle(className) – добавляет класс в список классов, если его там не было, и удаляет, если он там был.

Методы интерфейса Element:

– getAttribute(attr) – возвращает значение атрибута attr;

– setAttribute(attr, value) – устанавливает атрибуту attr значение value, если атрибута нет, то он добавляется;

– removeAttribute(attr) – удаляет атрибут attr;

– getBoundingClientRect() – возвращает объект со свойствами top, bottom, left, right, которые указывают на позицию элемента относительно верхнего левого угла браузера.

Для взаимодействия с пользователем, в JavaScript определён механизм событий. **Событие** – это сигнал от браузера о том, что произошло какое-либо действие, например, пользователь нажал на кнопку. Все DOM-узлы могут обрабатывать события.

Далее представлен список некоторых DOM-событий:

События мыши:

– click – происходит, когда пользователь кликнул на элемент левой кнопкой мыши (на устройствах с сенсорными экранами оно происходит при касании).

– contextmenu – происходит, когда пользователь кликнул на элемент правой кнопкой мыши.

– mouseover / mouseout – когда мышь наводится на / покидает элемент.

– mousedown / mouseup – когда пользователь нажал / отжал кнопку мыши.

– mousemove – когда пользователь двигает мышь.

События на элементах управления:

– submit – пользователь отправил форму <form>.

– focus – пользователь фокусируется на элементе, например, нажимает на <input>.

Клавиатурные события:

– keydown и keyup – когда пользователь нажимает / отпускает клавишу.

События документа:

– DOMContentLoaded – когда HTML загружен и обработан, DOM документа полностью построен и доступен.

CSS events:

– transitionend – когда CSS-анимация завершена.

Существует множество других событий.

Событию можно назначить **обработчик**, то есть функцию, которая вызовется, как только событие произошло. Именно благодаря обработчикам JavaScript может реагировать на действия пользователя.

Есть три **способа назначения обработчиков событий**:

1. Атрибут HTML: onclick="...".

2. Свойство элемента: elem.onclick= function.

3. Методы элемента: elem.addEventListener(event, handler[,phase]) для добавления, removeEventListener для удаления.

HTML-атрибуты используются редко потому, что принято разделять HTML-разметку от JavaScript-кода.

DOM-свойства вполне можно использовать, но нельзя назначить больше одного обработчика на один тип события. Во многих случаях с этим ограничением можно мириться.

Последний способ самый гибкий. К тому же, есть несколько типов событий, которые можно обработать только с помощью этого типа. К примеру, transitionend и DOMContentLoaded. Также addEventListener() поддерживает объекты типа Event. Объект event может быть передан в первый параметр функции, которая обрабатывает событие.

Объект интерфейса Event содержит следующие свойства:

– bubbles – имеет значение true, если событие является восходящим, т.е. может возникать на вложенном элементе, а обрабатываться на родительском;

– cancelable – имеет значение true, если можно отменить стандартную обработку события;

– currentTarget – определяет элемент, к которому прикреплён обработчик события;

– defaultPrevented – содержит true, если у этого объекта был вызван метод preventDefault(), который останавливает дальнейшую обработку события;

– enevtPhase – определяет стадию обработки события;

– target – указывает на элемент, на котором было вызвано событие;

– timeStamp – хранит время возникновения события;

– type – хранит имя события.

В JavaScript имеются три встроенных элемента для отображения **модальных окон**: alert, prompt, confirm.

Модальное окно alert

JavaScript-функция alert("Message") отображает модальное окно с сообщением «Message» и кнопкой «Закрыть». Модальными называются окна, которые блокируют работу

всей страницы в целом (и скриптов в частности) до тех пор, пока пользователь не выполнит требуемое окном действие. Пока модальное окно запущено, пользователь не может прокручивать страницу, нажимать на ней какие-либо кнопки или переходить по ссылкам.

Задача alert – донести до сведения пользователя какую-либо информацию, будь то справочная информация, информация об ошибке и так далее. Функция alert – единственная, которая ничего не возвращает.

Модальное окно prompt

Функция prompt() также, как и alert(), создает модальное окно, которое имеет поле для ввода и две кнопки: «Продолжить» и «Отмена». Вызов prompt() выглядит следующим образом:

let result = prompt(title, default)

Функция принимает два аргумента: title – название поля для ввода, default – строка, которая отображается в поле для ввода по умолчанию. Функция возвращает то, что ввел пользователь в поле ввода, если нажал «Продолжить». Если пользователь нажал «Отмена», функция возвратит null.

Задача prompt – запросить у пользователя ввод каких-либо данных, при этом пользователь может отказаться от ввода.

Модальное окно confirm

Это модальное окно служит для получения от пользователя подтверждения на заданный вопрос – ДА или НЕТ. Функция confirm(question) выводит окно с вопросом question и двумя кнопками: Да, Нет. Функция возвращает true при нажатии Да и false при нажатии Нет.

Для более сложных случаев можно эмитировать модальные окна с любым содержанием и поведением с помощью применения CSS к определённому блоку HTML, который скрыт по умолчанию. JavaScript позволяет динамически подменять CSS-классы этого HTML-блока, например по достижению какого либо события. Набор свойств этих CSS-классов должен отвечать за отрисовку модального окна: его размер, содержимое, задний фон, а с помощью DOM-API можно обрабатывать действия пользователя, взаимодействующего с модальным окном.

Задание:

Напишите JavaScript-скрипт для выполнения следующих задач:

– Создание интерактивной галереи изображений с кнопками "Следующее" и "Предыдущее".

– Реализация всплывающего окна с сообщением (modal window).

Динамическое изменение содержимого страницы при клике на элементы.

Дополнительно: используйте обработчики событий для управления взаимодействием пользователя с DOM.

Ход выполнения лабораторной работы должен быть отражен в *отчете*. Отчет должен содержать титульный лист, цель работы, задание, листинг исходного кода, описание проделанной работы, скриншоты результата, вывод, ответы на контрольные вопросы.

Контрольные вопросы:

1. Определение JavaScript. Как подключить JavaScript для HTML-документа?
2. Определение DOM и API.
3. Перечислите интерфейсы в DOM API.
4. Перечислите свойства объекта document.
5. Перечислите методы объекта document.
6. Перечислите свойства объекта типа Node.
7. Перечислите методы интерфейса Node.
8. Перечислите свойства объекта типа Element.
9. Перечислите методы интерфейса Element.
10. Как устроен механизм обработки событий. Перечислите основные DOM-события.
11. Перечислите события мыши и клавиатурные события.
12. Перечислите события на элементах управления и события на документах.
13. Что такое обработчик события? Какие способы назначения обработчиков событий?
14. Дайте определение модального окна. Перечислите встроенные элементы для отображения модальных окон в JavaScript.
15. Назначение модального окна alert.
16. Назначение модального окна prompt.
17. Назначение модального окна confirm.