Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебная дисциплина «Программные средства создания Internet-приложений»

Инструкция

по выполнению лабораторной работы
«Создание HTML-страниц с использованием сценариев JavaScript. Использование управляющих конструкций в JavaScript»

Минск
2020

Лабораторная работа № 17

Тема работы: Создание HTML-страниц с использованием сценариев JavaScript. Использование управляющих конструкций в JavaScript

1. Цель работы

Формирование умений использования управляющих конструкций при написании сценариев JavaScript.

2. Задание

Выполнить задания в соответствии с порядком выполнения лабораторной работы.

3. Оснащение работы

ПК, редактор исходного кода, браузер.

4. Основные теоретические сведения

Инструкции JavaScript состоят из: значений, операторов, выражений, ключевых слов и комментариев.

Переменные в JS **не являются типизированными**. Это означает, что при создании переменной не указывается, какого типа информация будет в ней находиться. Для того, чтобы использовать переменную, ее необходимо **объявить** и **инициализировать**.

Для объявления переменных используются ключевые слова **var** и **let**.

JavaScript поддерживает следующие виды операторов:

- математические;
- присваивания;
- сравнения;
- строковые;
- условные;
- логические;
- побитовые.

Таблица, содержащая операторы и их приоритет.

Операторы	Порядок	Приоритет
() (группировка)	не определено	1
. (доступ к свойствам и методам объекта) [] (доступ к элементам)	слева направо	2
new (создание объекта со списком аргументов)	не определено	
() (вызов функции, метода)	слева направо	3
new (создание объекта без списка аргументов)	справа налево	
++ (постфиксный инкремент), -- (постфиксный декремент)	не определено	4
! (логическое отрицание), **~** (побитовое отрицание), + (унарный плюс), - (унарный минус), ++ (префиксный инкремент), -- (префиксный декремент), **typeof**, **void**, **delete**	справа налево	5
****** (возведение в степень)	справа налево	6
***** (умножение), / (деление), % (остаток от деления)	слева направо	7
+ (сложение), - (вычитание)	слева направо	8
<< (сдвиг влево), >> (сдвиг вправо), >>> (сдвиг вправо с заполнением нулями)	слева направо	9
< (меньше), <= (меньше или равно), > (больше), >= (больше или равно), in, instanceof	слева направо	10

Операторы	Порядок	Приоритет
== (равенство), != (не равенство), === (строгое равенство), !== (строгое не равенство)	слева направо	11
& (побитовое И)	слева направо	12
^ (побитовое исключающее ИЛИ)	слева направо	13
| (побитовое ИЛИ)	слева направо	14
&& (логическое И)	слева направо	15
|| (логическое ИЛИ)	слева направо	16
?: (условный оператор)	справа налево	17
= (присваивание), +=, -=, *=, /=, %=, **=, <<=, >>=, >>>=, &=, ^=, |=	справа налево	18
yield, yield*	справа налево	19
... (расширение)	неопределено	20
, (запятая)	слева направо	21

Оператор, имеющий меньшее значение приоритета, выполняется раньше, чем оператор, имеющий более высокое значение приоритета. Если операторы имеют одинаковое значение приоритета, то они выполняются слева направо.

Условные операторы – это операторы языка JavaScript (ECMAScript), которые в зависимости от некоторого условия позволяют выполнить одно или несколько определённых инструкций.

Формы условных операторов в JavaScript:

- условный оператор if (с одной ветвью);
- условный оператор if...else (с двумя ветвями);
- условный оператор else if... (с несколькими ветвями);
- тернарный оператор (?:);
- оператор выбора switch.

Выражение (expression) – это комбинация значений, переменных и операторов, которые либо присваивают переменной значение, либо возвращают какое-то значение без его присваивания.

Язык JavaScript поддерживает стандартный набор управляющих конструкций:

- последовательная структура;
- условные операторы: if (структура с единичным выбором); if/else (с двойным выбором), else if... (с несколькими ветвями), тернарный оператор (?:), switch (с множественным выбором). **Условные операторы** - это операторы языка JavaScript (ECMAScript), которые в зависимости от некоторого условия позволяют выполнить одно или несколько определённых инструкций;
- пять типов структур повторения: while, do/while, for, for/in, for...of.

Любая программа может быть составлена на основе выбора нужных для реализации алгоритма программы типов управляющих структур из имеющихся, соединяемых друг с другом только двумя способами: сложение управляющих структур друг за другом и вложение управляющих структур друг в друга.

Типы данных в JavaScript можно разделить на две категории: простые типы и составные (объекты).

В JavaScript имеется восемь основных типов данных:

1) **number** для любых чисел: целочисленных или чисел с плавающей точкой, целочисленные значения ограничены диапазоном ±253.

2) **bigint** для целых чисел произвольной длины.

3) **string** для строк. Строка может содержать один или больше символов, нет отдельного символьного типа.

4) **boolean** для true/false.

5) **null** для неизвестных значений – отдельный тип, имеющий одно значение null.

6) **undefined** для неприсвоенных значений – отдельный тип, имеющий одно значение undefined.

7) **object** для более сложных структур данных.

8) **symbol** для уникальных идентификаторов.

JavaScript позволяет нам работать с примитивными типами данных – строками, числами и т.д., как будто они являются объектами. У них есть и методы. Каждый примитив имеет свой собственный «объект-обёртку», которые называются: String, Number, Boolean и Symbol. Таким образом, они имеют разный набор методов.

Для ввода и вывода данных можно воспользоваться методами браузера и загруженного в него документа.

Объект, представляющий свойства браузера, называется window (окно), а три его метода, предназначенных для ввода и вывода данных посредством диалоговых окон:

- **alert();**
- **prompt()**;
- **confirm()**.

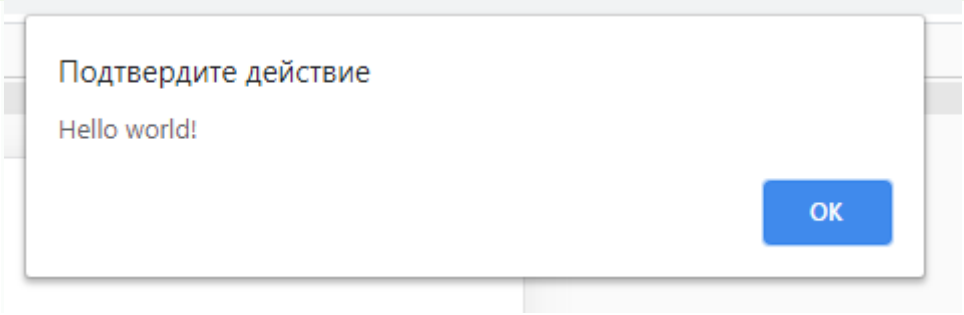
Метод **alert()** выводит на экран модальное окно с сообщением. Модальное окно означает, что выполнение сценария и дальнейшее взаимодействие со страницей приостанавливается до тех пор, пока не закроется данное окно, в данном случае, пока не будет нажата кнопка **OK** для продолжения работы.

Синтаксис метода:

alert(сообщение)

Пример:

```
<script>
  alert("Hello world!");
</script>
```



Метод **prompt()** выводит на экран модальное окно приглашения на ввод данных пользователем.

Синтаксис метода:

1 **var имя_переменной = prompt(msg, defaultText)**

где:

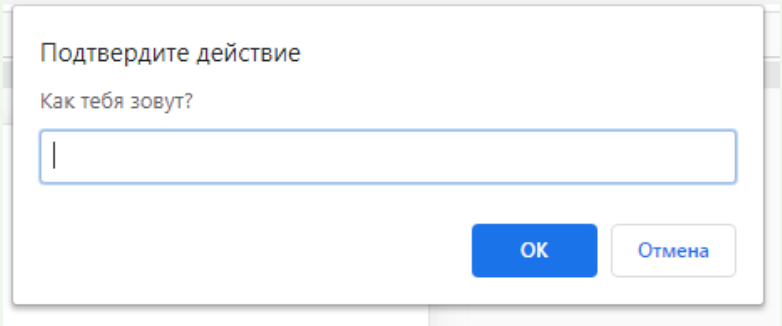
имя_переменной – имя используемой переменной, которой будет присвоено значение возвращаемое методом prompt(),

msg – сообщение, которое будет показано пользователю (обычно это вопрос),

defaultText – строка, которая отображается по умолчанию в поле ввода, обычно второй аргумент оставляют пустым и записывают так - "":

Пример:

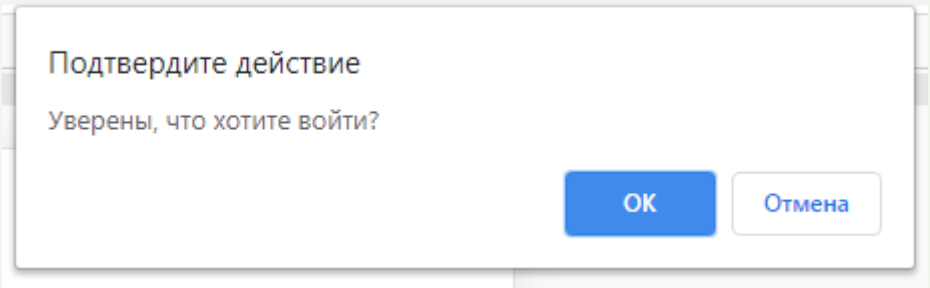
```
<script>
  let myName = prompt("Как тебя зовут?","");
  alert(`Привет ${ myName }!`);
</script>
```



Пользователь должен, что-нибудь ввести и нажать ***OK****, или отменить ввод нажав на* ***CANCEL****. Метод* ***prompt()*** *возвращает, то, что ввел пользователь - строку или специальное значение* ***null****, если ввод был отменен.*

Метод **confirm()** действует очень похоже на **alert()**, выводит окно с вопросом **question** с двумя кнопками: **OK** и **CANCEL**.

```
<SCRIPT >
confirm("Уверены, что хотите войти?")
</SCRIPT>
```



Одна команда сама по себе многого не дает. Нет никакой разницы, какой ответ будет выбран - "ОК" или "ОТМЕНА". Но стоит добавить функции IF (если) и ELSE (иначе), и готовы отличные эффекты.

Пример:

```
<SCRIPT >
if (confirm("Уверены, что хотите на прошлый урок?")) {
        parent.location='http://www.jsp.newmail.ru/les12.htm';
        alert("Счастливого пути");
        }
else {
        alert("Тогда оставайтесь");
        }
</SCRIPT>
```

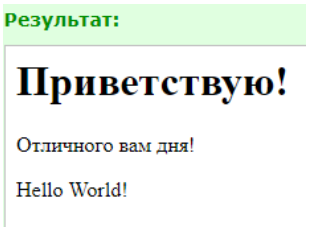
Метод **document.write()** выводит на страницу переданные ему аргументы.

Синтаксис метода:

document.write(arg1,arg2,arg3,...);

Аргументов может быть любое количество, и они могут быть любых типов, при выводе они преобразуются в строки:

```
<script>
  document.write("<h1>Приветствую!</h1><p>Отличного вам дня!</p>");
  document.write("Hello World!");
</script>
```



Метод **document.write()** работает только на этапе загрузки страницы. Если document.write() вызвать после того, как страница загрузилась, результатом будет - перезаписанная страница, с текстом, который был добавлен с помощью **document.write()**.

Также существует метод **document.writeln(str)**, который добавляет после str символ перевода строки "\n".

5. Порядок выполнения работы

1. Напишите сценарий, который с помощью диалогового окна запрашивает у пользователя его имя, и выводит приветственное сообщение для введенного имени. В случае, если пользователь нажал отмену или не ввел имя, должно выводиться сообщение, что такого имени не бывает.

2. Напишите сценарий, который запрашивает у пользователя возраст, и в зависимости от введенного возраста: от 2 до 5 – запрашивает номер сада, и вывод сообщение с указанием ранее введенного имени, какой сад посещает пользователь, от 6 до 16 – с указанием школы, от 17 до 21 – учебного заведения, от 22-до 65 – места работы, свыше 66 – информацию о том, что пользователь на заслуженном отдыхе. Реализовать указанный алгоритм двумя способами.

3. Напишите сценарий, который запрашивает у пользователя число и выводит сообщение в каком диапазоне (0-9, 10-19, 20-29 и т.д. до 100) находится введенное число. Добавить в модальное окно для ввода числа подсказку с указанием интервала, в пределах которого можно вводить число. Реализовать проверку, если введено не число, либо вообще ничего не введено, должны выводиться соответствующие сообщения.

4. Разместить на HTML-странице 3 кнопки. По нажатию на первую кнопку должно появляться модальное окно с предложением ввести число. Таким образом реализовать ввод 2-ух чисел. По нажатию на вторую кнопку должно выводиться модальное окно, содержащее сумму введенных чисел. По нажатию на третью кнопку должно выводиться модальное окно, содержащее произведение введенных чисел.

5. Разместить на веб-странице надпись «Я, (ФИО), учусь работать с JavaScript». Реализовать обработку событий в соответствии с вариантом.

Вариант	Задание
1	При наведении курсора, надпись должна увеличиваться, при выходе за пределы элемента надпись должна изменить цвет.
2	При клике на надпись должно выводиться сообщение с аналогичным содержанием. При двойном клике должно выводиться сообщение «хватит кликать!».
3	При каждом наведении на надпись, она должна увеличиваться в 1,5 раза. При клике на надпись, она должна вернуться к первоначальным разменам.
4	При наведении на надпись, она должна становиться невидимой, при выходе за пределы надпись – возвращаться в исходное состояние.
5	При клике по надписи, она должна изменить шрифт и начертание. При двойном клике – изменить цвет.

6*. Реализовать на HTML-странице простейший калькулятор.

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____
Фамилия, инициалы учащегося _____
Дата выполнения работы _____
Тема работы: _____
Цель работы: _____
Оснащение работы: _____
Результат выполнения работы: _____
_____
_____

7. Контрольные вопросы и задания

1. Какие управляющие конструкции поддерживает JavaScript?
2. Перечислите условные операторы, используемые в JavaScript.
3. Приведите пример использования структур повторения, используемые в JavaScript.
4. Охарактеризуйте типы данных JavaScript.

8. Рекомендуемая литература

1. **JAVASCRIPT.RU** [Электронный ресурс] / Современный учебник JavaScript – 2007—2020 Илья Кантор. – Режим доступа: https://learn.javascript.ru. – Дата доступа: 04.03.2020.
2. **Никсон, Р.** Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. 4-е изд. – СПб.: Питер, 2018.
3. **Симпсон, К.** ES6 и не только / К. Симпсон. – СПб.: Питер, 2017.
4. **Хавербеке, М.** Выразительный JavaScript. Современное веб-программирование / М. Хавербеке – СПб.: Питер, 2019.