Частное учреждение образования

«Колледж бизнеса и права»

| УТВЕРЖДАЮ  Заведующий  методическим кабинетом  \_\_\_\_\_\_\_\_\_\_Е.В.Паскал  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ |
| --- |

| Специальность:  2-40 01 01 «Программное обеспечение информационных технологий» | Дисциплина: «Скриптовые языки программирования» |
| --- | --- |

**Лабораторная работа № 1**

**Инструкционно-технологическая карта**

Тема: Подключение скриптов к html. Переменные. Имена переменных

Цель: Научиться подключать скрипты JavaScript к html страницам, создавать переменные с корректными именами.

Время выполнения: 2 часа

# Порядок выполнения работы

1. Изучить теоретические сведения к лабораторной работе.
2. Реализовать алгоритм решения задачи.
3. Отлаженную, работающую программу сдать преподавателю. Работу программы показать с помощью самостоятельно разработанных тестов.
4. Ответить на контрольные вопросы.

# Теоретические сведения

## Подключение скриптов к html.

**JavaScript в элементе script**

Самый простой способ внедрения JavaScript в HTML-документ – использование тега **<script>**. Теги **<script>** часто помещают в элемент **<head>**, и ранее этот способ считался чуть ли не обязательным. Однако в наши дни теги **<script>** используются как в элементе **<head>**, так и в теле веб-страниц.

Таким образом, на одной веб-странице могут располагаться сразу несколько сценариев. Как правило, выполнение сценариев браузерами происходит по мере их загрузки. Браузер читает HTML-документ сверху вниз и, когда он встречает тег **<script>**, рассматривает текст программы как сценарий и выполняет его. Остальной контент страницы не загружается и не отображается, пока не будет выполнен весь код в элементе **<script>**.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

</head>

<body>

<p>Это обычный HTML документ</p>

<script language="JavaScript">

    alert("Привет, мир!");

</script>

<p>Выходим обратно в HTML</p>

</body>

</html>

**JavaScript в атрибутах событий HTML-элементов**

Вышеприведенный сценарий был выполнен при открытии страницы и вывел строку: «Привет, мир!». Однако не всегда нужно, чтобы выполнение сценария начиналось сразу при открытии страницы. Чаще всего требуется, чтобы программа запускалась при определенном событии, например при нажатии какой-то кнопки.

В следующем примере функция JavaScript помещается в раздел **<head>** HTML-документа. Вот пример HTML-элемента **<button>** с атрибутом события, обеспечивающим реакцию на щелчки мышью. При нажатии кнопки генерируется событие **onclick**.

<!DOCTYPE html>

<html>

<head>

<script>

    function myFunction() {

    document.getElementById("demo").innerHTML = "Привет, javascript!";

    }

</script>

</head>

<body>

<p id="demo">Привет, мир!</p>

<button type="button" onclick="myFunction()">Кликни меня</button>

</body>

</html>

**Внешний JavaScript**

Если JavaScript-кода много – его выносят в отдельный файл, который, как правило, имеет расширение .js.

Чтобы включить в HTML-документ JavaScript-кoд из внешнего файла, нужно использовать атрибут **src (source)** тега **<script>**. Его значением должен быть URL-aдpec файла, в котором содержится JS-код:

<script src="/scripts/script.js"></script>

В этом примере указан абсолютный путь к файлу с именем script.js, содержащему скрипт (из корня сайта). Сам файл должен содержать только JavaScript-кoд, который иначе располагался бы между тегами **<script>** и **</script>.**

По аналогии с элементом **<img>** атрибуту **src** элемента **<script>** можно назначить полный URL-aдpec, не относящийся к домену текущей НТМL-страницы:

<script src=" http://www.somesite.com/script.js"></script>

## Переменные

Переменная – это «именованное хранилище» для данных. Переменные можно использовать для хранения товаров, посетителей и других данных.

Для создания переменной в JavaScript используется ключевое слово **let**.

Приведённая ниже инструкция создаёт (другими словами: объявляет или определяет) переменную с именем «message»:

let message;

Теперь можно поместить в неё данные, используя оператор присваивания =:

let message;

message = 'Hello'; // сохранить строку

Строка сохраняется в области памяти, связанной с переменной. Можно получить к ней доступ, используя имя переменной:

let message;

message = 'Hello!';

alert(message); // показывает содержимое переменной

Для краткости можно совместить объявление переменной и запись данных в одну строку:

let message = 'Hello!'; // определяем переменную и присваиваем ей значение

alert(message); // Hello!

Также можно объявить несколько переменных в одной строке:

let user = 'John', age = 25, message = 'Hello';

Такой способ может показаться короче, но он не рекомендуем. Для лучшей читаемости каждую переменную следует объявлять на новой строке.

Многострочный вариант немного длиннее, но легче для чтения:

let user = 'John';

let age = 25;

let message = 'Hello';

Некоторые люди также определяют несколько переменных в таком многострочном стиле:

let user = 'John',

  age = 25,

  message = 'Hello';

…Или даже с запятой в начале строки:

let user = 'John'

  , age = 25

  , message = 'Hello';

В принципе, все эти варианты работают одинаково. Так что это вопрос личного вкуса и эстетики.

## Аналогия из жизни

Концепцию «переменной» легко понять, если представить её в виде «коробки» для данных с уникальным названием на ней.

Например, переменную message можно представить как коробку с названием "message" и значением "Hello!" внутри:

Можно положить любое значение в коробку.

Также можно изменить его сколько угодно раз:

let message;

message = 'Hello!';

message = 'World!'; // значение изменено

alert(message);

При изменении значения старые данные удаляются из переменной:

Также можно объявить две переменные и скопировать данные из одной в другую.

let hello = 'Hello world!';

let message;

// копируем значение 'Hello world' из переменной hello в переменную message

message = hello;

// теперь две переменные содержат одинаковые данные

alert(hello); // Hello world!

alert(message); // Hello world!

#### Повторное объявление вызывает ошибку.

Переменная может быть объявлена только один раз.

Повторное объявление той же переменной является ошибкой:

let message = "Это";

// повторение ключевого слова 'let' приводит к ошибке

let message = "Другое"; // SyntaxError: 'message' has already been declared

Поэтому следует объявлять переменную только один раз и затем использовать её уже без **let**.

## Имена переменных

В JavaScript есть два ограничения, касающиеся имён переменных: имя переменной должно содержать только буквы, цифры или символы $ и \_.; первый символ не должен быть цифрой.

Примеры допустимых имён:

let userName;

let test123;

Если имя содержит несколько слов, обычно используется верблюжья нотация, то есть, слова следуют одно за другим, где каждое следующее слово начинается с заглавной буквы: **myVeryLongName**.

Знак доллара '$' и подчёркивание '\_' также можно использовать в названиях. Это обычные символы, как и буквы, без какого-либо особого значения.

Эти имена являются допустимыми:

let $ = 1; // объявили переменную с именем "$"

let \_ = 2; // а теперь переменную с именем "\_"

alert($ + \_); // 3

Примеры неправильных имён переменных:

let 1a; // не может начинаться с цифры

let my-name; // дефис '-' не разрешён в имени

**Регистр имеет значение**

Переменные с именами apple и AppLE – это две разные переменные.

#### Зарезервированные имена

Существует список зарезервированных слов, которые нельзя использовать в качестве имён переменных, потому что они используются самим языком.

Например: **let**, **class**, **return** и **function** зарезервированы.

Приведённый ниже код даёт синтаксическую ошибку:

let let = 5; // нельзя назвать переменную "let", ошибка!

let return = 5; // также нельзя назвать переменную "return", ошибка!

#### Создание переменной без использования use strict

Обычно нужно определить переменную перед её использованием. Но в старые времена было технически возможно создать переменную простым присвоением значения без использования **let**. Это все ещё работает, если не включить use strict в файлах, чтобы обеспечить совместимость со старыми скриптами.

// заметка: "use strict" в этом примере не используется

num = 5; // если переменная "num" раньше не существовала, она создаётся

alert(num); // 5

Это плохая практика, которая приводит к ошибке в строгом режиме:

"use strict";

num = 5; // ошибка: num is not defined

## Константы

Чтобы объявить константную, то есть, неизменяемую переменную, используйте const вместо let:

const myBirthday = '18.04.1982';

Переменные, объявленные с помощью const, называются «константами». Их нельзя изменить. Попытка сделать это приведёт к ошибке:

const myBirthday = '18.04.1982';

myBirthday = '01.01.2001'; // ошибка, константу нельзя перезаписать!

Если программист уверен, что переменная никогда не будет меняться, он может гарантировать это и наглядно донести до каждого, объявив её через const.

#### Константы в верхнем регистре

Широко распространена практика использования констант в качестве псевдонимов для трудно запоминаемых значений, которые известны до начала исполнения скрипта.

Названия таких констант пишутся с использованием заглавных букв и подчёркивания.

Например, сделаем константы для различных цветов в «шестнадцатеричном формате»:

const COLOR\_RED = "#F00";

const COLOR\_GREEN = "#0F0";

const COLOR\_BLUE = "#00F";

const COLOR\_ORANGE = "#FF7F00";

// ...когда нам нужно выбрать цвет

let color = COLOR\_ORANGE;

alert(color); // #FF7F00

Когда нужно использовать для констант заглавные буквы, а когда называть их нормально?

Название «константа» просто означает, что значение переменной никогда не меняется. Но есть константы, которые известны до выполнения (например, шестнадцатеричное значение для красного цвета), а есть константы, которые вычисляются во время выполнения сценария, но не изменяются после их первоначального назначения.

Например:

const pageLoadTime = /\* время, потраченное на загрузку веб-страницы \*/;

Значение **pageLoadTime** неизвестно до загрузки страницы, поэтому её имя записано обычными, а не прописными буквами. Но это всё ещё константа, потому что она не изменяется после назначения.

Другими словами, константы с именами, записанными **заглавными** буквами, используются только как псевдонимы для **«жёстко закодированных»** значений.

## Придумывайте правильные имена

В разговоре о переменных необходимо упомянуть, что есть ещё одна чрезвычайно важная вещь.

Название переменной должно иметь ясный и понятный смысл, говорить о том, какие данные в ней хранятся.

Именование переменных – это один из самых важных и сложных навыков в программировании. Быстрый взгляд на имена переменных может показать, какой код был написан новичком, а какой – опытным разработчиком.

В реальном проекте большая часть времени тратится на изменение и расширение существующей кодовой базы, а не на написание чего-то совершенно нового с нуля. Когда программист возвращается к коду после какого-то промежутка времени, гораздо легче найти информацию, которая хорошо размечена. Или, другими словами, когда переменные имеют хорошие имена.

#### Повторно использовать или создавать новую переменную?

И последняя заметка. Есть ленивые программисты, которые вместо объявления новых переменных повторно используют существующие.

В результате их переменные похожи на коробки, в которые люди бросают разные предметы, не меняя на них этикетки. Что сейчас находится внутри коробки? Кто знает? Нам необходимо подойти поближе и проверить.

Такие программисты немного экономят на объявлении переменных, но теряют в десять раз больше при отладке.

Дополнительная переменная – это добро, а не зло.

Современные JavaScript-минификаторы и браузеры оптимизируют код достаточно хорошо, поэтому он не создаёт проблем с производительностью. Использование разных переменных для разных значений может даже помочь движку оптимизировать ваш код.

# Контрольные вопросы

1. Опишите способы использования JavaScript в html.
2. Какое ключевое слово используется для создания переменных в JavaScript?
3. С каких символов не может начинаться переменная?
4. Каким ключевым словом объявляется константа?
5. Перечислите ограничения при названии переменных.
6. Приведите пример объявления константы.
7. Когда нужно использовать для констант заглавные буквы, а когда называть их нормально?

# Задания

**Задание 1**

Разместите в теле НТМL-страницы сценарий, выводящий всплывающее окно с надписью: "Привет, javascript!"

**Задание 2.**

Включите в НТМL-страницу сценарий из внешнего файла script.js, который расположен в той же директории, где располагается и сам HTML-документ.

**Задание 3.**

Объявите две переменные: admin и name. Запишите строку "Джон" в переменную name. Скопируйте значение из переменной name в admin. Выведите на экран значение admin, используя функцию alert (должна показать «Джон»).

**Задание 4.**

Создайте переменную для названия нашей планеты. Как бы вы её назвали? Создайте переменную для хранения имени текущего посетителя сайта. Как бы вы назвали такую переменную?

# Литература

**Диков, А.В.** Клиентские технологии веб программирования: JavaScript и DOM: учебное пособие / А.В. Диков. – СПб: Лань, 2020 – 124 с.

**Читанамбри, Кирупа.** Изучаем React / Кирупа Читанамбри – 2-е изд. – М.: Эксмо, 2019. 368 с.

**Хавербеке, Марейн.** Выразительный JavaScript. Современное веб-программирование. / Марейн Хавербеке – 3-е изд. – СПб.: Питер, 2019. – 480 с.:

Преподаватель Рогалевич А.В.

Рассмотрено на заседании цикловой

комиссии программного обеспечения

информационных технологий №10

Протокол № \_\_\_ от \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Председатель ЦК\_\_\_\_\_\_\_\_ В.Ю.Михалевич