ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 10. ИСПОЛЬЗОВАНИЕ ЯЗЫКА СЦЕНАРИЕВ JAVASCRIPT ПРИ СОЗДАНИИ WEB-САЙТА «ТЕМА

МДК. 09.01 Проектирование и разработка веб-приложений

Цель занятия: Познакомить обучающихся с языком клиентского программирования JavaScript

Краткие теоретические сведения:

JavaScript изначально создавался для того, чтобы сделать web-странички «живыми». Программы на этом языке называются скриптами. В браузере они подключаются напрямую к HTML и, как только загружается страничка – тут же выполняются.

JS – это клиентский язык программирования

Современный JavaScript – это «безопасный» язык программирования общего назначения. Он не предоставляет низкоуровневых средств работы с памятью, процессором, так как изначально был ориентирован на браузеры, в которых это не требуется.

Что же касается остальных возможностей – они зависят от окружения, в котором запущен JavaScript. В браузере JavaScript умеет делать всё, что относится к манипуляции со страницей, взаимодействию с посетителем и, в какой-то мере, с сервером:

Создавать новые HTML-теги, удалять существующие, менять стили элементов, прятать, показывать элементы и т.п.

Реагировать на действия посетителя, обрабатывать клики мыши, перемещения курсора, нажатия на клавиатуру и т.п.

Посылать запросы на сервер и загружать данные без перезагрузки страницы (эта технология называется "AJAX").

Получать и устанавливать cookie, запрашивать данные, выводить сообщения…

…и многое, многое другое!

JavaScript не может читать/записывать произвольные файлы на жесткий диск, копировать их или вызывать программы. Он не имеет прямого доступа к операционной системе.

Современные браузеры могут работать с файлами, но эта возможность ограничена специально выделенной директорией – «песочницей». Возможности по доступу к устройствам также прорабатываются в современных стандартах и частично доступны в некоторых браузерах.

JavaScript, работающий в одной вкладке, не может общаться с другими вкладками и окнами, за исключением случая, когда он сам открыл это окно или несколько вкладок из одного источника (одинаковый домен, порт, протокол).

Из JavaScript можно легко посылать запросы на сервер, с которого пришла страница. Запрос на другой домен тоже возможен, но менее удобен, т. к. и здесь есть ограничения безопасности.

Программы на языке JavaScript можно вставить в любое место HTML при помощи тега SCRIPT. Например:

```
<body>

  <p>Начало документа...</p>
  <script>
    alert( 'Привет, Мир!' );
  </script>

  <p>...Конец документа</p>

```

Сделайте страницу, которая выводит «Я – JavaScript!».

Создайте её на диске, откройте в браузере, убедитесь, что всё работает.

Если JavaScript-кода много – его выносят в отдельный файл, который подключается в HTML:

```
<script src="/path/to/script.js"></script>
```

Чтобы подключить несколько скриптов, используйте несколько тегов:

```
<script src="/js/script1.js"></script>
<script src="/js/script2.js"></script>
...
```

Браузер загружает и отображает HTML постепенно. Особенно это заметно при медленном интернет соединении: браузер не ждёт, пока

страница загрузится целиком, а показывает ту часть, которую успел загрузить.

Если браузер видит тег <script>, то он по стандарту обязан сначала выполнить его, а потом показать оставшуюся часть страницы.

Например, в примере ниже – пока все кролики не будут посчитаны – нижний <p> не будет показан:

```
  <p>Начинаем считать:</p>

  <script>
    alert( 'Первый кролик!' );
    alert( 'Второй кролик!' );
    alert( 'Третий кролик!' );
  </script>

  <p>Кролики посчитаны!</p>
```

Такое поведение называют «синхронным». Как правило, оно вполне нормально, но есть важное следствие.

Если скрипт – внешний, то пока браузер не выполнит его, он не покажет часть страницы под ним.

То есть, в таком документе, пока не загрузится и не выполнится big.js, содержимое <body> будет скрыто:

```
<html>
<head>
  <script src="big.js"></script>
</head>
<body>
  Этот текст не будет показан, пока браузер не выполнит big.js.
</body>
</html>
```

И здесь вопрос – действительно ли мы этого хотим? То есть, действительно ли оставшуюся часть страницы нельзя показывать до загрузки скрипта?

Что делать?

Можно поставить все подобные скрипты в конец страницы – это уменьшит проблему, но не избавит от неё полностью, если скриптов несколько. Допустим, в конце страницы 3 скрипта, и первый из них тормозит – получается, другие два его будут ждать – тоже нехорошо.

Кроме того, браузер дойдёт до скриптов, расположенных в конце страницы, они начнут грузиться только тогда, когда вся страница загрузится.

А это не всегда правильно. Например, счётчик посещений наиболее точно сработает, если загрузить его пораньше.

Поэтому «расположить скрипты внизу» – не лучший выход.

Кардинально решить эту проблему помогут атрибуты async или defer:

Поддерживается всеми браузерами, кроме IE9-. Скрипт выполняется полностью асинхронно. То есть, при обнаружении <script async src="..."> браузер не останавливает обработку страницы, а спокойно работает дальше. Когда скрипт будет загружен – он выполнится.

Поддерживается всеми браузерами, включая самые старые IE. Скрипт также выполняется асинхронно, не заставляет ждать страницу, но есть два отличия от async.

Первое – браузер гарантирует, что относительный порядок скриптов с defer будет сохранён.

То есть, в таком коде (с async) первым сработает тот скрипт, который раньше загрузится:

```
<script src="1.js" async></script>
<script src="2.js" async></script>
```

А в таком коде (с defer) первым сработает всегда 1.js, а скрипт 2.js, даже если загрузился раньше, будет его ждать.

Для того, чтобы добавить в код ещё одну команду – можно поставить её после точки с запятой.

Например, вместо одного вызова alert сделаем два:

```
alert('Привет'); alert('Мир');
```

Для объявления или, другими словами, создания переменной используется ключевое слово var:

```
var message;
```

После объявления, можно записать в переменную данные:

```
var message;
message = 'Hello'; // сохраним в переменной строку
```

Эти данные будут сохранены в соответствующей области памяти и в дальнейшем доступны при обращении по имени:

```
var message;
message = 'Hello!';

alert( message ); // выведет содержимое переменной
```

Для краткости можно совместить объявление переменной и запись данных:

```
var message = 'Hello!';
```

На имя переменной в JavaScript наложены всего два ограничения.

Имя может состоять из: букв, цифр, символов $ и _

Первый символ не должен быть цифрой.

А такие переменные были бы неправильными:

```
var 1a; // начало не может быть цифрой

var my-name; // дефис '-' не является разрешенным символом
```

Регистр букв имеет значение

Переменные apple и AppLE – две разные переменные.

Константа – это переменная, которая никогда не меняется. Как правило, их называют большими буквами, через подчёркивание. Например:

```
var COLOR_RED = "#F00";
var COLOR_GREEN = "#0F0";
var COLOR_BLUE = "#00F";
var COLOR_ORANGE = "#FF7F00";

var color = COLOR_ORANGE;
alert( color ); // #FF7F00
```

Единый тип число используется как для целых, так и для дробных чисел.

```
var n = 123;
n = 12.345;
```

Существуют специальные числовые значения Infinity (бесконечность) и NaN (ошибка вычислений).

Например, бесконечность Infinity получается при делении на ноль:

```
alert( 1 / 0 ); // Infinity
```

Ошибка вычислений NaN будет результатом некорректной математической операции, например:

```
alert( "нечисло" * 2 ); // NaN, ошибка
```

```
var str = "Мама мыла раму";
str = 'Одинарные кавычки тоже подойдут';
```

У него всего два значения: true (истина) и false (ложь).

Как правило, такой тип используется для хранения значения типа да/нет, например:

```
var checked = true; // поле формы помечено галочкой
checked = false;    // поле формы не содержит галочки
```

Значение null не относится ни к одному из типов выше, а образует свой отдельный тип, состоящий из единственного значения null:

```
var age = null;
```

В JavaScript null не является «ссылкой на несуществующий объект» или «нулевым указателем», как в некоторых других языках. Это просто специальное значение, которое имеет смысл «ничего» или «значение неизвестно».

Значение undefined, как и null, образует свой собственный тип, состоящий из одного этого значения. Оно имеет смысл «значение не присвоено».

Если переменная объявлена, но в неё ничего не записано, то её значение как раз и есть undefined:

```
var x;
alert( x ); // выведет "undefined"
```

Первые 5 типов называют «примитивными».

Особняком стоит шестой тип: «объекты».

Он используется для коллекций данных и для объявления более сложных сущностей.

Объявляются объекты при помощи фигурных скобок {...}, например:

```
var user = { name: "Вася" };
```

Функция prompt принимает два аргумента:

Она выводит модальное окно с заголовком title, полем для ввода текста, заполненным строкой по умолчанию default и кнопками OK/CANCEL.

```
result = prompt(title, default);
```

Как и в случае с alert, окно prompt модальное.

```
var years = prompt('Сколько вам лет?', 100);

alert('Вам ' + years + ' лет!')
```

Синтаксис:

```
result = confirm(question);
```

confirm выводит окно с вопросом question с двумя кнопками: OK и CANCEL.

Например:

```
var isAdmin = confirm("Вы - администратор?");

alert( isAdmin );
```

Иногда, в зависимости от условия, нужно выполнить различные действия. Для этого используется оператор if.

Например:

```
var year = prompt('В каком году появилась спецификация ECMA-262 5.1?', '');

if (year != 2011) alert( 'А вот и неправильно!' );
```

Оператор if («если») получает условие, в примере выше это year != 2011. Он вычисляет его, и если результат – true, то выполняет команду.

Если нужно выполнить более одной команды – они оформляются блоком кода в фигурных скобках:

```
if (year != 2011) {
  alert( 'А вот..' );
  alert( '..и неправильно!' );
}
```

Необязательный блок else («иначе») выполняется, если условие неверно:

```
var year = prompt('Введите год появления стандарта ECMA-262 5.1', '');

if (year == 2011) {
  alert( 'Да вы знаток!' );
} else {
  alert( 'А вот и неправильно!' ); // любое значение, кроме 2011
}
```

Бывает нужно проверить несколько вариантов условия. Для этого используется блок else if .... Например:

```
var year = prompt('В каком году появилась спецификация ECMA-262 5.1?', '');

if (year < 2011) {
  alert( 'Это слишком рано..' );
} else if (year > 2011) {
  alert( 'Это поздновато..' );
} else {
  alert( 'Да, точно в этом году!' );
}
```

Иногда нужно в зависимости от условия присвоить переменную. Например:

```
var access;
var age = prompt('Сколько вам лет?', '');

if (age > 14) {
  access = true;
} else {
  access = false;
}

alert(access);
```

Оператор вопросительный знак '?' позволяет делать это короче и проще.

Он состоит из трех частей:

условие ? значение1 : значение2

Проверяется условие, затем если оно верно – возвращается значение1, если неверно – значение2, например:

```
access = (age > 14) ? true : false;
```

Последовательность операторов '?' позволяет вернуть значение в зависимости не от одного условия, а от нескольких.

Например:

```
var age = prompt('возраст?', 18);

var message = (age < 3) ? 'Здравствуй, малыш!'
  (age < 18) ? 'Привет!' :
  (age < 100) ? 'Здравствуйте!' :
  'Какой необычный возраст!';

alert( message );
```

Поначалу может быть сложно понять, что происходит. Однако, внимательно приглядевшись, мы замечаем, что это обычная последовательная проверка!

Вопросительный знак проверяет сначала age < 3, если верно – возвращает 'Здравствуй, малыш!', если нет – идет за двоеточие и проверяет age < 18. Если это верно – возвращает 'Привет!', иначе проверка age < 100 и 'Здравствуйте!'… И наконец, если ничего из этого не верно, то 'Какой необычный возраст!'.

То же самое через if..else:

```
if (age < 3) {
  message = 'Здравствуй, малыш!';
} else if (age < 18) {
  message = 'Привет!';
} else if (age < 100) {
  message = 'Здравствуйте!';
} else {
  message = 'Какой необычный возраст!';
}
```

Иногда оператор вопросительный знак '?' используют как замену if:

```
var company = prompt('Какая компания создала JavaScript?', '');

(company == 'Netscape') ?
   alert('Да, верно') : alert('Неправильно');
```

Рекомендуется не использовать вопросительный знак таким образом.

Циклы while, for

При написании скриптов зачастую встает задача сделать однотипное действие много раз.

Например, вывести товары из списка один за другим. Или просто перебрать все числа от 1 до 10 и для каждого выполнить одинаковый код.

Для многократного повторения одного участка кода – предусмотрены циклы.

Цикл while имеет вид:

Пока условие верно – выполняется код из тела цикла.

```
while (условие) {
  // код, тело цикла
}
```

Например, цикл ниже выводит i пока i < 3:

```
var i = 0;
while (i < 3) {
  alert( i );
  i++;
}
```

Если бы i++ в коде выше не было, то цикл выполнялся бы (в теории) вечно. На практике, браузер выведет сообщение о «зависшем» скрипте и посетитель его остановит.

Проверку условия можно поставить под телом цикла, используя специальный синтаксис do..while:

```
do {
  // тело цикла
} while (условие);
```

Цикл, описанный, таким образом, сначала выполняет тело, а затем проверяет условие.

```
var i = 0;
do {
  alert( i );
  i++;
} while (i < 3);
```

Синтаксис do..while редко используется, т.к. обычный while нагляднее – в нём не приходится искать глазами условие и ломать голову, почему оно проверяется именно в конце.

Чаще всего применяется цикл for. Выглядит он так:

```
for (начало; условие; шаг) {
  // ... тело цикла ...
}
```

Пример цикла, который выполняет alert(i) для i от 0 до 2 включительно (до 3):

```
var i;

for (i = 0; i < 3; i++) {
  alert( i );
}
```

Выйти из цикла можно не только при проверке условия но и, вообще, в любой момент. Эту возможность обеспечивает директива break.

Например, следующий код подсчитывает сумму вводимых чисел до тех пор, пока посетитель их вводит, а затем – выдаёт:

```
var sum = 0;

while (true) {

  var value = +prompt("Введите число", '');

  if (!value) break; // (*)

  sum += value;

}
alert( 'Сумма: ' + sum );
```

Директива break в строке (*), если посетитель ничего не ввёл, полностью прекращает выполнение цикла и передаёт управление на строку за его телом, то есть на alert.

Конструкция switch заменяет собой сразу несколько if.

Она представляет собой более наглядный способ сравнить выражение сразу с несколькими вариантами.

Массив – это переменная в которой может храниться множество значений.

Они обычно используются для хранения упорядоченных коллекций данных, например – списка товаров на странице, студентов в группе и т.п.

Пустой массив:

```
var arr = [];
```

Массив fruits с тремя элементами:

```
var fruits = ["Яблоко", "Апельсин", "Слива"];
```

Элементы нумеруются, начиная с нуля.

Чтобы получить нужный элемент из массива – указывается его номер в квадратных скобках:

```
var fruits = ["Яблоко", "Апельсин", "Слива"];

alert( fruits[0] ); // Яблоко
alert( fruits[1] ); // Апельсин
alert( fruits[2] ); // Слива
```

Элемент можно всегда заменить:

```
fruits[2] = 'Груша'; // теперь ["Яблоко", "Апельсин", "Груша"]
```

…Или добавить:

```
fruits[3] = 'Лимон'; // теперь ["Яблоко", "Апельсин", "Груша", "Лимон"]
```

Общее число элементов, хранимых в массиве, содержится в его свойстве length:

```
var fruits = ["Яблоко", "Апельсин", "Груша"];

alert( fruits.length ); // 3
```

Через alert можно вывести и массив целиком.

При этом его элементы будут перечислены через запятую:

```
var fruits = ["Яблоко", "Апельсин", "Груша"];

alert( fruits ); // Яблоко,Апельсин,Груша
```

Конец массива

pop

Удаляет последний элемент из массива и возвращает его:

```
var fruits = ["Яблоко", "Апельсин", "Груша"];

alert( fruits.pop() ); // удалили "Груша"

alert( fruits ); // Яблоко, Апельсин
```

push

Добавляет элемент в конец массива:

```
var fruits = ["Яблоко", "Апельсин"];

fruits.push("Груша");

alert( fruits ); // Яблоко, Апельсин, Груша
```

Вызов fruits.push(...) равнозначен fruits[fruits.length] = ...

Начало массива

shift

Удаляет из массива первый элемент и возвращает его:

```
var fruits = ["Яблоко", "Апельсин", "Груша"];

alert( fruits.shift() ); // удалили Яблоко

alert( fruits ); // Апельсин, Груша
```

unshift

Добавляет элемент в начало массива:

```
var fruits = ["Апельсин", "Груша"];

fruits.unshift('Яблоко');

alert( fruits ); // Яблоко, Апельсин, Груша
```

Методы push и unshift могут добавлять сразу по несколько элементов:

```
var fruits = ["Яблоко"];

fruits.push("Апельсин", "Персик");
fruits.unshift("Ананас", "Лимон");

// результат: ["Ананас", "Лимон", "Яблоко", "Апельсин", "Персик"]
alert( fruits );
```

Внутреннее устройство массива

Массив – это объект, где в качестве ключей выбраны цифры, с дополнительными методами и свойством length.

Так как это объект, то в функцию он передаётся по ссылке:

```
function eat(arr) {
  arr.pop();
}

var arr = ["нам", "не", "страшен", "серый", "волк"]

alert( arr.length ); // 5
eat(arr);
eat(arr);
alert( arr.length ); // 3, в функцию массив не скопирован, а передана ссылка
```

Ещё одно следствие – можно присваивать в массив любые свойства.

Например:

```
var fruits = []; // создать массив

fruits[99999] = 5; // присвоить свойство с любым номером

fruits.age = 25; // назначить свойство со строковым именем
```

… Но массивы для того и придуманы в JavaScript, чтобы удобно работать именно с упорядоченными, нумерованными данными. Для этого в них существуют специальные методы и свойство length.

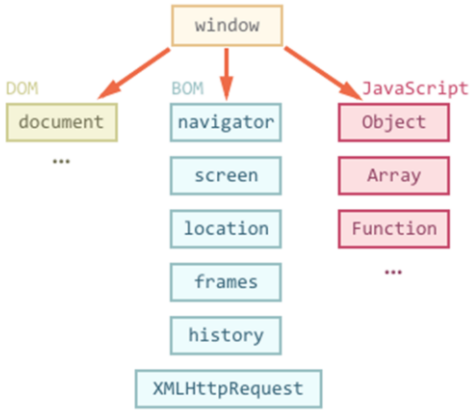
Как правило, нет причин использовать массив как обычный объект, хотя технически это и возможно.

Окружение: DOM, BOM и JS

Сам по себе язык JavaScript не предусматривает работы с браузером.

Он вообще не знает про HTML. Но позволяет легко расширять себя новыми функциями и объектами.

На рисунке ниже схематически отображена структура, которая получается если посмотреть на совокупность браузерных объектов с «высоты птичьего полёта».



Как видно из рисунка, на вершине стоит window.

У этого объекта двоякая позиция – он с одной стороны является глобальным объектом в JavaScript, с другой – содержит свойства и методы для управления окном браузера, открытия новых окон, например:

```
// открыть новое окно/вкладку с URL http://ya.ru
window.open('http://ya.ru');
```

Глобальный объект document даёт возможность взаимодействовать с содержимым страницы.

Пример использования:

```
document.body.style.background = 'red';
alert( 'Элемент BODY стал красным, а сейчас обратно вернётся' );
document.body.style.background = '';
```

BOM – это объекты для работы с чем угодно, кроме документа.

Например:

Объект navigator содержит общую информацию о браузере и операционной системе. Особенно примечательны два свойства: navigator.userAgent – содержит информацию о браузере и navigator.platform – содержит информацию о платформе, позволяет различать Windows/Linux/Mac и т.п.

Объект location содержит информацию о текущем URL страницы и позволяет перенаправить посетителя на новый URL.

Функции alert/confirm/prompt – тоже входят в BOM.

Пример использования:

```
alert( location.href ); // выведет текущий адрес
```

Основным инструментом работы и динамических изменений на странице является DOM (Document Object Model) – объектная модель, используемая для XML/HTML-документов.

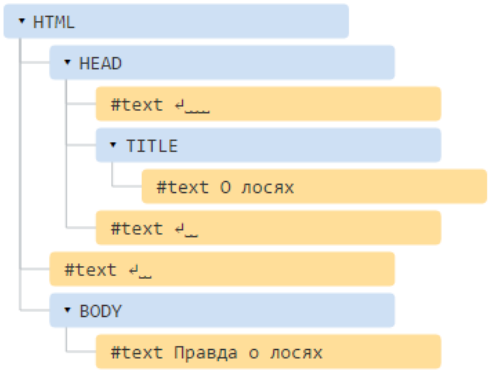
Согласно DOM-модели, документ является иерархией, деревом. Каждый HTML-тег образует узел дерева с типом «элемент». Вложенные в него теги становятся дочерними узлами. Для представления текста создаются узлы с типом «текст».

DOM – это представление документа в виде дерева объектов, доступное для изменения через JavaScript.

Построим, для начала, дерево DOM для следующего документа.

```
<!DOCTYPE HTML>
<html>
<head>
  <title>О лосях</title>
</head>
<body>
  Правда о лосях
</body>
</html>
```

Его вид:



В этом дереве выделено два типа узлов.

Теги образуют узлы-элементы (element node). Естественным образом одни узлы вложены в другие. Структура дерева образована исключительно за счет них.

Текст внутри элементов образует текстовые узлы (text node), обозначенные как #text. Текстовый узел содержит исключительно строку текста и не может иметь потомков, то есть он всегда на самом нижнем уровне.

На рисунке выше синие узлы-элементы можно кликать, при этом их дети будут скрываться-раскрываться.

Обратите внимание на специальные символы в текстовых узлах:

перевод строки: ↵

пробел: ␣

Всё честно – если пробелы есть в документе, то они есть и в DOM, а если их убрать, то и в DOM их не будет, получится так:

```
<!DOCTYPE HTML>
<html><head><title>О лосях</title></head><body>Правда о лосях</body></html>
```

Например, можно поменять цвет BODY и вернуть обратно:

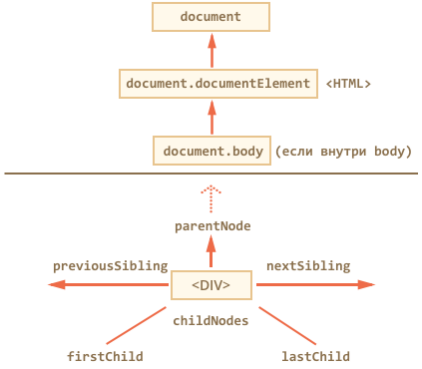
```
document.body.style.backgroundColor = 'red';
alert( 'Поменяли цвет BODY' );

document.body.style.backgroundColor = '';
alert( 'Сбросили цвет BODY' );
```

DOM позволяет делать что угодно с HTML-элементом и его содержимым, но для этого нужно сначала нужный элемент получить.

Доступ к DOM начинается с объекта document. Из него можно добраться до любых узлов.

Так выглядят основные ссылки, по которым можно переходить между узлами DOM:



Самые верхние элементы дерева доступны напрямую из document.

<HTML> = document.documentElement

Первая точка входа – document.documentElement. Это свойство ссылается на DOM-объект для тега <html>.

<BODY> = document.body

Вторая точка входа – document.body, который соответствует тегу <body>.

В современных браузерах (кроме IE8-) также есть document.head – прямая ссылка на <head>

Есть два принципиально разных термина.

Дочерние элементы (или дети) – элементы, которые лежат непосредственно внутри данного. Например, внутри <HTML> обычно лежат <HEAD> и <BODY>.

Потомки – все элементы, которые лежат внутри данного, вместе с их детьми, детьми их детей и так далее. То есть, всё поддерево DOM.

Псевдо-массив childNodes хранит все дочерние элементы, включая текстовые.

Пример ниже последовательно выведет дочерние элементы document.body:

```
<!DOCTYPE HTML>
<html>

<body>
  <div>Начало</div>

  <ul>
    <li>Информация</li>
  </ul>

  <div>Конец</div>

  <script>
    for (var i = 0; i < document.body.childNodes.length; i++) {
      alert( document.body.childNodes[i] ); // Text, DIV, Text, UL, ..., SCRIPT
    }
  </script>
  ...
</body>

</html>
```

Свойства firstChild и lastChild обеспечивают быстрый доступ к первому и последнему элементу.

Контрольные вопросы:

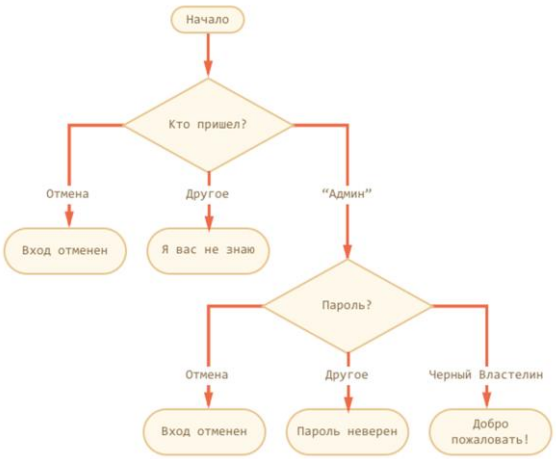
1. Что такое язык JavaScript
2. Какие функции JS вы знаете?
3. Что умеет делать язык JS?
4. Что не умеет делать язык JS?
5. Как вывести значение в тег?
6. Что такое массив в JS?
7. Что такое объект в JS?
8. Что делает функция getElementById ?

Задание для практического занятия:

Задание № 1. Напишите код, который будет спрашивать логин (prompt).

Если посетитель вводит «Админ», то спрашивать пароль, если нажал отмена (escape) – выводить «Вход отменён», если вводит что-то другое – «Я вас не знаю».

Пароль проверять так. Если введён пароль «Чёрный Властелин», то выводить «Добро пожаловать!», иначе – «Пароль неверен», при отмене – «Вход отменён».



Задание № 2. Создайте переменную str и присвойте ей значение 'abcde'. Обращаясь к отдельным символам этой строки выведите на экран символ 'a', символ 'b', символ 'e'.

Задание № 3. Задача. Напишите скрипт, который считает количество секунд в часе.

Задание № 4. Задача. Переделайте приведенный код так, чтобы в нем использовались операции +=, -=, *=, /=, ++, --. Количество строк кода при этом не должно измениться. Код для переделки:

```
var num = 1;
num = num + 12;
num = num - 14;
num = num * 5;
num = num / 7;
num = num + 1;
num = num - 1;
alert(num);
```

Задание № 5. Работа с переменными

Создайте переменную num и присвойте ей значение 3. Выведите значение этой переменной на экран с помощью метода alert.

Создайте переменные a=10 и b=2. Выведите на экран их сумму, разность, произведение и частное (результат деления).

Создайте переменные c=15 и d=2. Просуммируйте их, а результат присвойте переменной result. Выведите на экран значение переменной result.

Создайте переменные a=10, b=2 и c=5. Выведите на экран их сумму.

Создайте переменные a=17 и b=10. Отнимите от a переменную b и результат присвойте переменной c. Затем создайте переменную d, присвойте ей значение 7. Сложите переменные c и d, а результат запишите в переменную result. Выведите на экран значение переменной result.

Задание № 6. Работа со строками

Создайте переменную str и присвойте ей значение 'Привет, Мир!'. Выведите значение этой переменной на экран.

Создайте переменные str1='Привет, ' и str2='Мир!'. С помощью этих переменных и операции сложения строк выведите на экран фразу 'Привет, Мир!'.

Создайте переменную name и присвойте ей ваше имя. Выведите на экран фразу 'Привет, %Имя%!'.

Создайте переменную age и присвойте ей ваш возраст. Выведите на экран 'Мне %Возраст% лет!'.

Задание № 7. Функция prompt

Спросите имя пользователя с помощью методы prompt. Выведите с помощью alert сообщение 'Ваше имя %имя%'.

Спросите у пользователя число. Выведите с помощью alert квадрат этого числа.

Задание № 8. Обращение к символам строки

Создайте переменную str и присвойте ей значение 'abcde'. Обращаясь к отдельным символам этой строки выведите на экран символ 'a', символ 'c', символ 'e'.

Создайте переменную num и присвойте ей значение '12345'. Найдите произведение (умножение) цифр этого числа.

Задание № 9. Практическая часть

Напишите скрипт, который считает количество секунд в часе, в сутках, в месяце.

Создайте три переменные - час, минута, секунда. С их помощью выведите текущее время в формате 'час:минута:секунда'.

Создайте переменную, присвойте ей число. Возведите это число в квадрат. Выведите его на экран.

Задание № 10. Работа с массивами и объектами

Задача. Дан массив с элементами 'Привет, ', 'мир' и '!'. Необходимо вывести на экран фразу 'Привет, мир!'.

Задание № 11. Дан массив ['Привет, ', 'мир', '!']. Необходимо записать в нулевой элемент этого массива слово 'Пока, ' (то есть вместо слова 'Привет, ' будет 'Пока, ').

Задание № 12. Объекты (ассоциативные массивы)

Задача. Создайте ассоциативный массив (объект) заработных плат obj. Выведите на экран зарплату Пети и Коли.

```
//Этот объект дан:
var obj = {'Коля':'1000', 'Вася':'500', 'Петя':'200'};
```

Задача № 13. Создайте массив arr с элементами 1, 2, 3, 4, 5 двумя различными способами.

Задача № 14. Многомерный массив

Дан многомерный массив arr:

```
var arr = {
    'ru':['голубой', 'красный', 'зеленый'],
    'en':['blue', 'red', 'green'],
};
```

Выведите с его помощью слово 'голубой'.

Здание № 15. Работа с массивами

Создайте массив arr = ['a', 'b', 'c']. Выведите его на экран с помощью функции alert.

С помощью массива arr из предыдущего номера выведите на экран содержимое первого, второго и третьего элементов.

Создайте массив arr = ['a', 'b', 'c', 'd'] и с его помощью выведите на экран строку 'a+b, c+d'.

Создайте массив arr с элементами 2, 5, 3, 9. Умножьте первый элемент массива на второй, а третий элемент на четвертый. Результаты сложите, присвойте переменной result. Выведите на экран значение этой переменной.

Здание № 15. Работа с ассоциативными массивами (Объектами)

Создайте объект obj. Выведите на экран элемент с ключом 'c' двумя способами: через квадратные скобки и как свойство объекта:

```
var obj = {a: 1, b: 2, c: 3};
```

Создайте массив заработных плат obj. Выведите на экран зарплату Пети и Коли.

```
var obj = {Коля: '1000', Вася: '500', Петя: '200'};
```

Создайте объект с днями недели. Ключами в нем должны служить номера дней от начала недели (понедельник - первый и т.д.). Выведите на экран текущий день недели.

Пусть теперь номер дня недели хранится в переменной day, например там лежит число 3. Выведите день недели, соответствующий значению переменной day.

Здание № 16. Работа с многомерными массивами

Дан массив [[1, 2, 3], [4, 5, 6], [7,8,9]]. Выведите на экран цифру 4 из этого массива.

Дан объект {js:['jQuery', 'Angular'], php: 'hello', css: 'world'}. Выведите с его помощью слово 'jQuery'.

Создайте двухмерный массив. Первые два ключа - это 'ru' и 'en'. Пусть первый ключ содержит элемент, являющийся массивом названий дней недели по-русски, а второй - по-английски. Выведите с помощью этого массива понедельник по-русски и среду по английски (пусть понедельник - это нулевой день).

Пусть теперь в переменной lang хранится язык (она принимает одно из значений или 'ru', или 'en' - либо то, либо то), а в переменной day - номер дня. Выведите словом день недели, соответствующий переменным lang и day. То есть: если, к примеру, lang = 'ru' и day = 3 - то выведем 'среда'.

Задание № 17. Условия

Если переменная a равна 10, то выведите 'Верно', иначе выведите 'Неверно'.

Задание № 18. В переменной min лежит число от 0 до 59. Определите в какую четверть часа попадает это число (в первую, вторую, третью или четвертую).

Задание № 19.

Переменная lang может принимать 2 значения: 'ru' 'en'. Если она имеет значение 'ru', то в переменную arr запишем массив дней недели на русском языке, а если имеет значение 'en' – то на английском. Решите задачу через 2 if, через switch-case и через многомерный массив без ифов и switch.

Задание № 20. Работа с if-else

Если переменная a равна нулю, то выведите 'Верно', иначе выведите 'Неверно'. Проверьте работу скрипта при a, равном 1, 0, -3.

Если переменная a равна 'test', то выведите 'Верно', иначе выведите 'Неверно'. Проверьте работу скрипта при a, равном 'test', 'тест', 3.

Если переменная a равна '1' и по значению и по типу, то выведите 'Верно', иначе выведите 'Неверно'. Проверьте работу скрипта при a, равном '1', 1, 3.

Задание № 21. Самостоятельное решение

В переменной month лежит какое-то число из интервала от 1 до 12. Определите в какую пору года попадает этот месяц (зима, лето, весна, осень).

Дана строка, состоящая из символов, например, 'abcde'. Проверьте, что первым символом этой строки является буква 'a'. Если это так - выведите 'да', в противном случае выведите 'нет'.

Дана строка с цифрами, например, '12345'. Проверьте, что первым символом этой строки является цифра 1, 2 или 3. Если это так - выведите 'да', в противном случае выведите 'нет'.

Дана строка из 3-х цифр. Найдите сумму этих цифр. То есть сложите как числа первый символ строки, второй и третий.

Дана строка из 6-ти цифр. Проверьте, что сумма первых трех цифр равняется сумме вторых трех цифр. Если это так - выведите 'да', в противном случае выведите 'нет'.

Задание № 21.Работа с циклами в JS

Выведите столбец чисел от 1 до 50.

Задание № 22.

Дан массив с элементами [1, 2, 3, 4, 5]. С помощью цикла for выведите все эти элементы на экран.

Задание № 23. Цикл for-in

Дан объект obj с ключами 'Минск', 'Москва', 'Киев' с элементами 'Беларусь', 'Россия', 'Украина'. С помощью цикла for-in выведите на экран строки такого формата: 'Минск - это Беларусь.'.

Задание № 24. Самостоятельное решение

Дан массив с элементами 2, 5, 9, 15, 0, 4. С помощью цикла for и оператора if выведите на экран столбец тех элементов массива, которые больше 3-х, но меньше 10.

Дан массив с числами. Числа могут быть положительными и отрицательными. Найдите сумму положительных элементов массива.

Дан массив с элементами 1, 2, 5, 9, 4, 13, 4, 10. С помощью цикла for и оператора if проверьте есть ли в массиве элемент со значением, равным 4. Если есть - выведите на экран 'Есть!' и выйдите из цикла. Если нет - ничего делать не надо.

Дан массив числами, например: [10, 20, 30, 50, 235, 3000]. Выведите на экран только те числа из массива, которые начинаются на цифру 1, 2 или 5.

Дан массив с элементами 1, 2, 3, 4, 5, 6, 7, 8, 9. С помощью цикла for создайте строку '-1-2-3-4-5-6-7-8-9-'.

Составьте массив дней недели. С помощью цикла for выведите все дни недели, а выходные дни выведите жирным.

Составьте массив дней недели. С помощью цикла for выведите все дни недели, а текущий день выведите курсивом. Текущий день должен храниться в переменной day.

Дано число n=1000. Делите его на 2 столько раз, пока результат деления не станет меньше 50. Какое число получится? Посчитайте количество итераций, необходимых для этого (итерация - это проход цикла), и запишите его в переменную num.

Задание № 25. Работа с функциями в JS

Сделайте функцию, которая возвращает куб числа. Число передается параметром.

Задание № 26. Самостоятельное решение

Сделайте функцию, которая возвращает квадрат числа. Число передается параметром.

Сделайте функцию, которая возвращает сумму двух чисел.

Сделайте функцию, которая отнимает от первого числа второе и делит на третье.

Сделайте функцию, которая принимает параметром число от 1 до 7, а возвращает день недели на русском языке.

Задание № 27. Работа с массивами и циклами

Дан массив с числами. Создайте из него новый массив, где останутся лежать только положительные числа. Создайте для этого вспомогательную функцию isPositive(), которая параметром будет принимать число и возвращать true, если число положительное, и false - если отрицательное.

Задание № 28. Самостоятельное решение

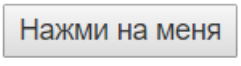
Сделайте функцию isNumberInRange, которая параметром принимает число и проверяет, что оно больше нуля и меньше 10. Если это так - пусть функция возвращает true, если не так - false.

Дан массив с числами. Запишите в новый массив только те числа, которые больше нуля и меньше 10-ти. Для этого используйте вспомогательную функцию isNumberInRange из предыдущей задачи.

Сделайте функцию getDigitsSum (digit - это цифра), которая параметром принимает целое число и возвращает сумму его цифр.

Задание № 29. Работа с DOM

Сделайте так, чтобы при нажатии на кнопку выводился текст с помощью команды alert «Привет»:

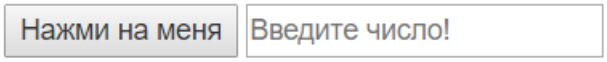


Задание № 30.

Сделайте так, чтобы при нажатии на кнопку она меняла текст в импуте:

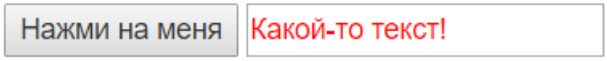
Задание № 31.

Повторите поведение кнопки по нажатию на нее (она выводит алертом содержимое инпута, возведенное в квадрат):



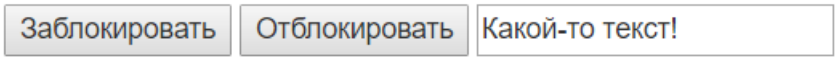
Задание № 32.

Повторите поведение кнопки по нажатию на нее (она меняет цвет текста в инпуте):



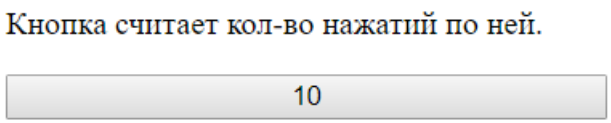
Задание № 33.

Повторите поведение кнопок по нажатию на них (одна из них блокирует инпут с помощью атрибута disabled, а другая разблокирует):

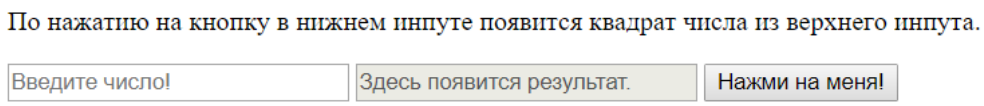


Задание № 34. Самостоятельное решение

Повторите страницу по данному по образцу:



Повторите страницу по данному по образцу:

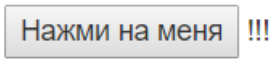


С помощью стандартных элементов html создайте простой калькулятор, выполняющий простые математические операции

Вводить информацию в калькулятор можно с помощью кнопок, вся вводимая информация появляется в текстовом поле для ввода, а в другое поле выведите результат набранной команды на калькуляторе:

Задание № 35.Функция innerHTML

Повторите поведение кнопки по нажатию на нее (она меняет текст в span):



Задание № 36.Функция innerHTML + цикл

Дан HTML код (см. под задачей). Поменяйте содержимое абзацев на их порядковый номер в коде.

```
<h2>Заголовок, не поменяется.</h2>
<p>Абзац, поменяется.</p>
<p>Абзац, поменяется.</p>
<p>Абзац, поменяется.</p>
```

Задание № 37.Функция getElementsByClassName

Дан HTML код (см. под задачей). Поменяйте содержимое элементов с классом zzz на их порядковый номер в коде.

```
<h2 class="zzz">Заголовок с классом zzz.</h2>
<p class="zzz">Абзац с классом zzz.</p>
<p class="zzz">Абзац с классом zzz.</p>
<p>Просто абзац, не поменяется.</p>
```

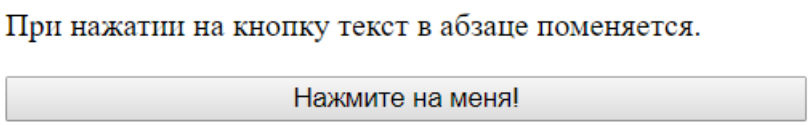
Задание № 38.Функция querySelectorAll

Дан HTML код. Поменяйте содержимое абзацев с классом zzz на их порядковый номер в коде.

```
<h2 class="zzz">Заголовок с классом zzz, не поменяется.</h2>
<p class="zzz">Абзац с классом zzz.</p>
<p class="zzz">Абзац с классом zzz.</p>
<p>Просто абзац, не поменяется.</p>
```

Задание № 39. Самостоятельное решение

Повторите страницу по данному по образцу (заменить текст можно на любой):



Задание № 40. Работа с датами в JS

Выведите на экран текущие день, месяц и год в формате 'год-месяц-день'.

Задание № 41.

Выведите на экран текущий месяц словом, по-русски.

Задание № 42. Самостоятельное решение

Для решения задач данного блока вам понадобятся следующие методы объекта Date: getFullYear, getMonth, getDate, getHours, getMinutes, getSeconds.

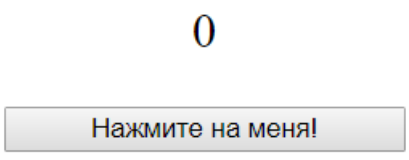
Выведите на экран текущий месяц.

Выведите на экран текущий год.

Выведите на экран текущую дату-время в формате '12:59:59 31.12.2014'. Для решения этой задачи напишите функцию, которая будет добавлять 0 перед днями и месяцами, которые состоят из одной цифры (из 1.9.2014 сделает 01.09.2014).

Задание № 43. Работа с таймерами в JS

Создайте отсчет от 0 до бесконечности:



Задание № 43. Самостоятельное решение

Создайте тикающие часики:

14:16:05

Инструкция по выполнению задания:

Решение задачи № 2. Решение: будем обращаться к отдельным символам этой строки, к примеру, буква 'a' имеет номер 0 и ее можно вывести так - str[0], буква 'b' имеет номер 1 и так далее:

```
var str = 'abcde';
alert(str[0]); //выведем букву 'a'
alert(str[1]); //выведем букву 'b'
alert(str[4]); //выведем букву 'e'
```

Решение задачи № 3. Решение: так как в минуте 60 секунд, а в часе - 60 минут, то умножив 60 на 60 мы получим количество секунд в часе:

```
alert(60 * 60);
```

Если нам нужно получить количество секунд в сутках, то умножим еще и на 24 (так как в сутках 24 часа):

```
alert(60 * 60 * 24);
```

Решение задачи № 4. Решение: заменим все подходящие места на сокращенную форму записи. К примеру, вместо num = num + 12 можно написать num += 12, а вместо num = num + 1 будет num++. Результат переделки будет выглядеть так:

```
var num = 1;
num += 12;
num -= 14;
num *= 5;
num /= 7;
num++;
num--;
alert(num);
```

Решение задачи № 10.

Решение:

```
//Выведем фразу 'Привет, мир!':
var arr = ['Привет, ', 'мир', '!'];
alert(arr[0] + arr[1] + arr[2]);
```

Разберем это решение: слово 'Привет, ' хранится под номером 0, это значит, что для доступа к нему мы должны написать arr[0]. Для доступа к слову 'мир' мы должны написать arr[1], а arr[2] содержит в себе '!'. Далее с помощью оператора '+' мы сложим три наши строки ('Привет, ', 'мир' и '!') в одну строку таким образом arr[0] + arr[1] + arr[2], и выведем на экран с помощью alert.

Решение задачи № 11.

Решение:

```
var arr = ['Привет, ', 'мир', '!'];
arr[0] = 'Пока, '; //перезапишем нулевой элемент массива
alert(arr); //посмотрим на массив и убедимся в том, что он изменился
```

Решение задачи № 12.

Решение: Чтобы вывести зарплату Коли следует вывести значение элемента объекта с ключом 'Коля'. Сделаем это:

```
var obj = {'Коля':'1000', 'Вася':'500', 'Петя':'200'};
alert(obj['Коля']); //выведет 1000
```

Решение задачи № 13.

Решение:

Первый способ создать массив - объявить его через [ ]:

```
var arr = [1, 2, 3, 4, 5];
```

Второй способ создания массива - это поступить таким образом:

```
var arr = [];
arr[0] = 1;
arr[1] = 2;
arr[2] = 3;
arr[3] = 4;
arr[4] = 5;
```

Решение задачи № 14.

Решение: так как массив многомерный (в нашем случае двухмерный), то нам придется написать несколько квадратных скобок подряд. Поясню это по шагам. Давайте сделаем так:

```
alert(arr['ru']);
```

В этом случае результатом будет массив ['голубой','красный', 'зеленый'], который является частью нашего многомерного массива. Чтобы вывести слово 'голубой', необходимо дописать еще одну квадратную скобку с ключом, соответствующим этому элементу (у него нет явного ключа - значит его ключ 0):

```
alert(arr['ru'][0]); //выведет 'голубой'
```

Выведем теперь слово 'красный':

```
alert(arr['ru'][1]); //выведет 'красный'
```

Выведем 'red':

```
alert(arr['en'][1]); //выведет 'red'
```

Решение задачи № 17.

Решение:

```
var num = 10;
if (num == 10) {
    alert('Верно');
} else {
    alert('Неверно');
}
```

Решение задачи № 18.

Решение:

```
var min = 10;
if (min >= 0 && min <= 14) {
    alert('В первую четверть.');
}
if (min >= 15 && min <= 30) {
    alert('Во вторую четверть.');
}
if (min >= 31 && min <= 45) {
    alert('В третью четверть.');
}
if (min >= 46 && min <= 59) {
    alert('В четвертую четверть.');
}
```

Решение задачи № 19.

Решение: через 2 if:

```
var lang = 'ru';
if (lang == 'ru') {
    var arr = ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];
}
if (lang == 'en') {
    arr = ['mn', 'ts', 'wd', 'th', 'fr', 'st', 'sn'];
}
alert(arr);
```

Решение через switch-case:

```
var lang = 'ru';
switch (lang) {
    case 'ru':
        var arr = ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];
    break;
    case 'en':
        arr = ['mn', 'ts', 'wd', 'th', 'fr', 'st', 'sn'];
    break;
}
alert(arr);
```

Решение через многомерный массив:

```
var lang = 'ru';
var arr = {
    'ru':['пн', 'вт', 'ср','чт', 'пт', 'сб', 'вс'],
    'en':['mn', 'ts', 'wd', 'th', 'fr', 'st', 'sn'],
};
alert(arr[lang]);
```

Решение задачи № 21.

Решение: воспользуемся циклом while (отделим числа тегом br друг от друга, чтобы получить столбец, а не строку):

```
var i = 1;
while (i <= 50) {
    document.write(i + '<br>');
    i++;
}
```

Можно также воспользоваться и циклом for:

```
for (var i = 1; i <= 50; i++) {
    document.write(i + '<br>');
}
```

Решение задачи № 22.

Решение: будем повторять цикл for от 0 до номера последнего элемента массива. Этот номер на единицу меньше количества элементов в массиве, которое можно найти с помощью свойства length таким образом: arr.length.

Чтобы цикл прокрутился на единицу меньше длины массива, в условие окончания мы поставим <, а не <=.

К элементам массива будем обращаться так: arr[i]. При этом переменная i - это счетчик цикла, который будет меняться от нуля до arr.length (не включительно). Таким образом мы последовательно выведем все элементы массива на экран (отделив их тегом br друг от друга):

```
var arr = [1, 2, 3, 4, 5];
for (var i = 0; i < arr.length; i++) {
    document.write(arr[i] + '<br>');
}
```

Решение задачи № 23.

Решение: задача не представляет сложности если уметь работать с циклом for-in. Давайте решать задачу поэтапно. Для начала выведем на экран все ключи объекта (это названия городов):

```
var obj = {
    'Минск': 'Беларусь',
    'Москва': 'Россия',
    'Киев': 'Украина'
};

for (var key in obj) {
    alert(key);
}
```

А теперь выведем все значения объекта (это страны):

```
var obj = {
    'Минск': 'Беларусь',
    'Москва': 'Россия',
    'Киев': 'Украина'
};

for (var key in obj) {
    alert(obj[key]);
}
```

Ну, а теперь сформируем строки нужного нам формата:

```
var obj = {
    'Минск': 'Беларусь',
    'Москва': 'Россия',
    'Киев': 'Украина'
};

for (var key in obj) {
    alert(key + ' - это ' + obj[key] + '.');
}
```

Решение задачи № 25.

Решение:

```
function cube($num) {
    return $num * $num * $num
}
```

Решение задачи № 27.

Решение:

```
var arr = [1, 2, 3, -1, -2, -3];

function isPositive(num) {
    if (num >=0) {
        return true;
    } else {
        return false;
    }
}

var newArr = [];
for (var i = 0; i <= arr.length; i++) {
    if (isPositive(arr[i])) {
        newArr.push(arr[i]);
    }
}

console.log(newArr);
```

Решение задачи № 29.

Решение:

```
<button onclick="alert('Привет!')">Нажми на меня</button>
```

Решение задачи № 30.

Решение:

HTML - разметка

```
<button onclick="buttonClick()">Нажми на меня</button>
<input type="text" id="input" value="???">
```

JavaScript код:

```
function buttonClick() {
    var input = document.getElementById('input');
    input.value = '!!!';
}
```

Решение задачи № 31.

Решение:

HTML – разметка

```
<button onclick="buttonClick()">Нажми на меня</button>
<input type="text" id="input" placeholder="Введите число!">
```

JavaScript код:

```
function buttonClick() {
    var input = document.getElementById('input');
    var number = Number(input.value);
    var square = number*number;
    alert(square);
}
```

Решение задачи № 32.

Решение:

HTML – разметка

```
<button onclick="buttonClick()">Нажми на меня</button>
<input type="text" id="input" value="Какой-то текст!">
```

JavaScript код:

```
function buttonClick() {
    var input = document.getElementById('input');
    input.style.color = 'red';
}
```

Можно не вводить промежуточную переменную input, а сразу поменять цвет:

```
function buttonClick() {
    document.getElementById('input').style.color = 'red';
}
```

Решение задачи № 33.

Решение:

HTML – разметка

```
<button onclick="button1Click()">Заблокировать</button>
<button onclick="button2Click()">Отблокировать</button>
<input type="text" id="input" value="Какой-то текст!">
```

JavaScript код:

```
//Заблокирует элемент
function button1Click() {
    var input = document.getElementById('input');
    input.disabled = true;
}

//Отблокирует элемент
function button2Click() {
    var input = document.getElementById('input');
    input.disabled = false;
}
```

Решение задачи № 35.

Решение:

HTML – разметка

```
<button onclick="buttonClick()">Нажми на меня</button>
<span id="elem">Это span с текстом.</span>
```

JavaScript код:

```
function buttonClick() {
    var elem = document.getElementById('elem');
    elem.innerHTML = '!!!';
}
```

Решение задачи № 36.

Решение: создадим кнопку, по нажатию на которую будет вызываться функция func, которая выполнит указанные в условии задачи действия:

HTML – разметка

```
<input type="submit" onclick="func()">
```

JavaScript код:

Получим теперь элементы по имени тега и переберем их циклом, используя метод getElementsByTagName:

```
function func() {
    var elems = document.getElementsByTagName('p');
    for (var i = 0; i < elems.length; i++) {
        elems[i].innerHTML = i+1;
    }
}
```

HTML код станет выглядеть так:

```
<h2>Заголовок, не поменяется.</h2>
<p>1</p>
<p>2</p>
<p>3</p>
<input type="submit" onclick="func()">
```

Решение задачи № 37.

Решение: создадим кнопку, по нажатию на которую будет вызываться функция func, которая выполнит указанные в условии задачи действия:

HTML – разметка

```
<input type="submit" onclick="func()">
```

JavaScript код:

Получим теперь элементы по имени класса и переберем их циклом, используя метод getElementsByClassName:

```
function func() {
    var elems = document.getElementsByClassName('zzz');
    for (var i = 0; i < elems.length; i++) {
        elems[i].innerHTML = i+1;
    }
}
```

HTML код станет выглядеть так:

```
<h2 class="zzz">1</h2>
<p class="zzz">2</p>
<p class="zzz">3</p>
<p class="zzz">4</p>
<p>Просто абзац, не поменяется.</p>
<input type="submit" onclick="func()">
```

Решение задачи № 38.

Решение: создадим кнопку, по нажатию на которую будет вызываться функция func, которая выполнит указанные в условии задачи действия:

HTML – разметка

```
<input type="submit" onclick="func()">
```

JavaScript код:

Получим теперь элементы с помощью селектора p.zzz (абзацы с классом zzz) и переберем их циклом, используя метод querySelectorAll:

```
function func() {
    var elems = document.querySelectorAll('p.zzz');
    for (var i = 0; i < elems.length; i++) {
        elems[i].innerHTML = i+1;
    }
}
```

HTML код станет выглядеть так:

```
<h2 class="zzz">Заголовок с классом zzz, не поменяется.</h2>
<p class="zzz">1</p>
<p class="zzz">2</p>
<p class="zzz">3</p>
<p>Просто абзац, не поменяется.</p>
<input type="submit" onclick="func()">
```

Решение задачи № 40.

Решение:

```
var date = new Date();
alert(date.getFullYear() + '-' + date.getMonth() + '-' + date.getDate());
```

Решение задачи № 41.

Решение: создадим массив месяцев months, затем получим номер текущего месяца с помощью getMonth, и выведем месяц словом, обратившись к элементу массива months с ключом, равным номеру текущего месяца, вот так - months[month]:

```
var months = [
    'янв', 'фев', 'мар', 'апр', 'май', 'июн',
    'июл', 'авг', 'сен', 'окт', 'ноя', 'дек'
];

var date = new Date();
var month = date.getMonth();
alert(months[month]);
```

Решение задачи № 43.

HTML - код

```
<div id="wrapper">
    <p id="test">
        0
    </p>
    <input type="submit" value="Нажмите на меня!"  onclick="go()">
</div>
```

JavaScript – код

```
<script>
    //Запускает таймер
    function go(){
        window.setInterval(timer, 1000);
    }
    function timer(){
        var test = document.getElementById('test');
        test.innerHTML = parseInt(test.innerHTML)+1;
    }
</script>
```

Средства обучения:

- ✓ Персональный компьютер с выходом в сеть Интернет
- ✓ Редактор Notepad++ или Sublime text 3
- ✓ Браузер Google Chrome или Mozilla Firefox или Opera или любой другой браузер для запуска файлов html

Критерии оценки результатов выполнения практического задания:

Оценка «отлично» ставится, если работа выполнена в полном объеме с соблюдением необходимой последовательности действий; продемонстрирована самостоятельность выполнения практического задания, умение работать с инструкцией.

Оценка «хорошо» ставится, если выполнены требования к оценке "отлично", но допущены 2-3 недочета.

Оценка «удовлетворительно» ставится, если работа выполнена не полностью, но объем выполненной части таков, что позволяет получить правильные результаты и выводы; в ходе проведения работы были допущены ошибки.

Оценка «неудовлетворительно» ставится, если работа выполнена не полностью или объем выполненной части работы не позволяет сделать правильных выводов