Что обозначает приставка Java в имени JavaScript?

Что такое идентификатор?

**Идентификатор** – это имя. Идентификаторы в JavaScrip выступают в качестве названия переменных и функций, а так же меток некоторых циклов.

Какие типы есть в JavaScript?

Что умеет JavaScript?

Что не умеет JavaScript?

 не умеет читать и записывать произвольные файлы на жесткий диск, запускать установку каких-либо программ. Это сделано для безопасности пользователей, чтобы злоумышленник не мог с помощью JavaScript получить личные данные или как-то навредить компьютеру пользователя.

*  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 (Isolated Storage).
*  JavaScript, работающий в одной вкладке, не может общаться с другими вкладками и окнами. За исключением случая, когда он сам открыл это окно.
*  Современные браузеры поддерживают интерфейс postMessage, который позволяет наладить взаимодействие между страницами. Страница загруженная с одного домена не может отправлять сообщения или вызывать код загруженный с другого домена, а postMessage позволяет обойти это ограничение.
*  Из JavaScript можно легко посылать запросы на сервер, с которого пришла страница. Запрос к другому домену возможен, но менее удобен, т.к. имеются ограничения.

Назовите три функции для работы с диалоговыми окнами?

**alert() -** Выводит модальное окно с сообщением.

*  Посетитель не сможет продолжить работу, пока не нажмет на кнопку "ОК" в модальном окне.

**Пример:** alert('Добрый день');

**prompt() -** Выводит сообщение в окне с текстовым полем и двумя кнопками: "ОК" и "ОТМЕНА".

*  Возвращает введенное значение или null, если посетитель нажал на кнопку "ОТМЕНА".
*  Как и в alert(), окно - модальное, то есть посетитель не может делать ничего другого, пока не выберет одну из кнопок.

**Пример:** result = prompt("text", "value");

**confirm() -** выводит окно с вопросом **question** с двумя кнопками: "ОК" и "ОТМЕНА"

**Пример:** var a = confirm("Вы - администратор?");

**JavaScript** поддерживает две разновидности массивов:

*  Ассоциативный массив (Hash), где данные хранятся по произвольному ключу.
*  Числовой массив Array, где данные хранятся по номерам.

Objects

Объекты в JavaScript обладают определенной двусмысленностью. Они сочетают в себе две важных возможности:

* 1. Являются ассоциативными массивами.
* 2. Обладают свойствами и поведением.

**Объект в JavaScript** — это в первую очередь просто коллекция свойств (можно называть ассоциативным массивом или списком), состоящая из пар ключ-значение. Причем ключом может быть только строка, а значением — любой тип данных.

**Ассоциативный массив** — структура (или тип) данных, в которой можно хранить любые данные в формате ключ-значение.

Другое название для ассоциативного массива — «**словарь**» или «**хэш**». Одному ключу соответствует единственное значение, которое можно быстро прочитать, записать или удалить.

Объект может содержать в себе любые значения, которые называются свойствами объекта. Доступ к свойствам осуществляется по имени свойства («по ключу»).

В отличии от языков, где реализована класс-объектная парадигма, нам не нужно создавать сначала класс, чтобы потом создать объект этого класса. Мы можем сразу создать объект.

**Конструктор в JavaScript** — это не часть класса (в JavaScript нет классов), а просто самостоятельная функция.

Объект может быть как угодно модифицирован в любое время — до, после и даже во время использования.

**Основные операции с объектами — это:**

* 1. Присвоение значения свойству объекта по ключу.
* 2. Чтение значения свойства по ключу.
* 3. Удаление свойства по ключу.
* 4. Добавление новых свойств и методов динамически в любое время.

# Функция-прототип

* Что такое функция-конструктор?
* 2. Правильно ли утверждать что в JavaScript есть классы?
* 3. Какое назначение оператора new при использовании совместно с функцией конструктором?
* 4. В чем разница между свойством и методом?
* 5. В чем разница между свойством экземпляра и свойством функции-конструктора?
* 6. В чем заключается разница между toString и valueOf?
* 7. Опешите действия которые нужно предпринять если нужно создать объекты которые поддерживают сравнение по значению
* В языке JavaScript не существует классов, но с помощью функций-конструкторов и прототипов можно реализовать поведение подобное использованию классов в других ООП языках программирования. Если в контексте JavaScript используется термин «класс», под ним подразумевается использование прототипа или функции-конструктора.
* Функции, которые применяются совместно с оператором new называются функциями-конструкторами. Задача оператора **new** – создать пустой объект и передать его функции конструктору в качестве значения ключевого слова this. Задача **функции-конструктора** – заполнить объект свойствами и методами.
* Пример создания объекта – new Date().
* Каждый объект имеет свой набор **свойств экземпляра**. Если имеется 10 объектов созданных функцией-конструктором, то у каждого объекта будет свое свойство (всего будет 10 копий свойств). Свойство экземпляра – свойство, которое было задано через функцию конструктор.
* **Метод экземпляра** – то же самое что и свойства экземпляра, но вместо значения в нем храниться функция.
* Если в методе экземпляра нужно обратиться к свойствам экземпляра обязательно нужно использовать ключевое слово **this.**
* В языке C# есть понятие статический метод и статическое свойство, то есть значение или действе, которое является общим для всех, а не принадлежит, конкретному экземпляру. Для того, что бы имитировать статические методы в JavaScript, метод или свойство нужно определить на функции-конструкторе, а не на объекте.
* Методы функции-конструктора не могут использовать ключевое слово this, так как они не связаны с конкретным объектом.
* **Прототип –** это объект, от которого наследуется создаваемый новый объект. Любой объект в JavaScript наследует свойства и методы своего прототипа. Если добавить значение в прототип оно будет доступно для всех объектов, которые были созданы через функцию конструктор. Для получения доступа к прототипу используется следующий синтаксис Конструктор.prototype
* Когда в JavaScript создается новая функция-конструктор, она добавит в объект предопределенный набор свойств.
* Общие методы прототипа Object:
* **toString()** – метод для преобразования объекта в строковое представление.
* **valueOf()** – во многом похож на toString(), но вызывается когда JavaScript требуется преобразовать объект к элементарному типу.
* **equals() –** метод (не является методом Object) используется для сравнения двух объектов по значению.
* **compareTo(obj)** – метод (не является методом Object) используется для определения отношения между объектами. Если метод возвращает значение меньше нуля объект на котором был произведен вызов меньше чем объект, которые передан в качестве параметра. Если метод вернет значение больше нуля – объект, на котором произведен вызов, больше чем тот, который передан в параметрах. Результат метода равен нулю если объекты равны.

# Sort function

function compare(a,b) {

if (a.last\_nom < b.last\_nom)

return -1;

if (a.last\_nom > b.last\_nom)

return 1;

return 0;

}

objs.sort(compare);

objs.sort(function(a,b) {return (a.last\_nom > b.last\_nom) ? 1 : ((b.last\_nom > a.last\_nom) ? -1 : 0);} );

array.sort(function (a, b) {

if (a.name > b.name) {

return 1;

}

if (a.name < b.name) {

return -1;

}

// a должно быть равным b

return 0;

}

)

Display(array);

# Check if undefined

if(array[i].menst!=null)

# Способы создания модулей

Способы создания модулей

**Использование объекта**

varModule = {}

Module.property= 123;

Module.method= function(){

}

**Использование анонимной функции**

(function(){

varproperty = 123;

functionMethod(){

}

})();

# DOM

var button = document.getElementsByTagName('input')[0];

button.onclick = createNode;

var body = document.getElementsByTagName('body')[0];

var countElem = 0;

function createNode()

{

if (countElem >= 10) {

for (var i = 0; i < body.childNodes.length; i++)

{

if (body.childNodes[i].nodeName == 'P')

{

body.removeChild(body.childNodes[i]);

i = 0;

}

}

countElem = 0;

}

else {

var p = document.createElement('p');

p.innerHTML = RandomChars();

body.appendChild(p);

countElem++;

}

}

function RandomChars()

{

var text = "";

var possible = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";

for (var i = 0; i < 5; i++)

text += possible.charAt(Math.floor(Math.random() \* possible.length));

return text;

}

Способы подключения JS сценария:

*  В элементе script в элементе head
*  В элементе script любом элементе страницы
*  В атрибуте определяющем событие JavaScript, например onclick
*  Используя псевдо протокол javascript:// в href ссылок.

Для получения доступа к элементам на странице используются следующие методы объекта document:

getElementById() – получение элементов по Id.

getElementsByName() – получение элементов по атрибуту name.

getElementsByTagName() – получение элементов по типу элементов.

Для того, что бы иметь возможность использовать вышеперечисленные методы нужно быть уверенным в том что они сформированы как узлы в модели страницы. Для того что бы методы работали вызывать их следует или в обработчике события onload или в элементе script, который находится после объявления элементов в документе.

Пространства имен – концепция, назначение, которой свести к минимуму возможность конфликтов при использовании нескольких модулей, которые могут объявлять глобальные переменные с одинаковыми именами. Способы создания пространства имен:

* Определение объекта, который будет выступать в роли пространства имен. Все функции и переменные должны быть определены как свойства и методы этого объекта. Имя объекта, который выступает в роли пространства имен, должно совпадать с именем файла, в котором он будет объявлен. При использовании такого подхода модуль объявляет только одну глобальную переменную, название которой совпадает с именем файла.
*  Определение анонимной функции, которая сразу же запускается. В теле анонимной функции определяются вся логика модуля. В таком случае все переменные являются локальными для анонимной функции и в итоге модуль не создает глобальных имен. Такой подход используется в jQuery и в других подобных JavaScript фреймворках

Ненавязчивый JavaScript (Unobtrusive JavaScript) – отделение функциональности страницы от ее структуры. JavaScript должен находиться в отдельных модулях и блоки JavaScript кода не должны появляться в разметке.

Document Object Model (DOM) – объектная модель документа – структура объектов представляющая загруженный HTML или XML контент и позволяющая манипулировать этим контентом.

**Закрепление материала**

# Location History Navigation

Функции для работы с таймерами setTimeout, clearTimeout, setInterval, clearInterval.

Данные функции являются глобальными (являются методами объекта Window) и используются для того что бы выполнить задержку перед тем вызвать определенный метод или для того что бы задать интервал между повторяющимися вызовами одного и того же метода.

Location – объект имеющий доступ к URL-адресу текущего документа. С помощью данного объекта можно заставить страницу обновиться или перейти на новый адрес. Свойство location объекта окна или документа, дает возможность получить доступ к экземпляру объекта Location.

History – объект для перенаправления пользователя по истории просмотра страниц. Метод forward() и backward() позволяют перемещаться вперед и назад по истории.

Navigator – объект хранящий общую информацию о веб-браузере, который выполняет сценарий.

Screen – хранит информацию о размере экрана пользователя.

Объект Window содержит свойства для получение информации о текущем окне, о высоте ширине, координатах окна на экране.

Также объект Window имеет методы для манипуляции окнами:

window.open(url, name, params) – открывает новое окно и отображает в нем HTML документ, адрес которого передан в первом параметре.

obj. close() – закрывает окно, ранее открытое сценарием.

Многие современные браузеры блокируют всплывающие окна, так как часто на сайтах их используют для отображения нежелательной рекламы.

Задание 1

Разработайте программу секундомер. Секундомер должен иметь три кнопки «Старт», «Стоп», «Сбросить». Секундомер должен выводить время в формате 00:00:00.

<script>

var sec = 0;

var min = 0;

var hour = 0

var intervalHandler;

window.onload = function () {

var output = document.getElementById('output');

var start = document.getElementById('btn1');

var stop = document.getElementById('btn2');

var reset = document.getElementById('btn3');

start.onclick = function ()

{

intervalHandler = setInterval(Count, 1);

}

stop.onclick = function ()

{

clearInterval(intervalHandler);

}

reset.onclick = function ()

{

sec = 0;

min = 0;

hour = 0;

output.innerHTML = hour + ":" + min + ":" + sec;

}

function Count()

{

sec++;

if (sec > 59)

{

min++;

sec = 0;

}

if (min > 59)

{

hour++

min = 0;

}

if (hour > 23)

{

hour = 0;

}

output.innerHTML = hour + ":" + min + ":" + sec;

}

}

</script>

</head>

<body>

<p id="output">-</p>

Задание 2

Разработайте страницу, которая будет выводить сумму двух GET параметров a и b. Например, если на страницу заходят по адресу, page.html?a=**100**&b=**200** то в теле страницы должно отображаться

<script type="text/javascript">

// Функция для чтения данных, которые переданы при GET запросе (в адресной строке).

function getQueryString() {

var args = {}; // пустой объект

// login=admin password=123

var query = location.search.substring(1); // Получение строки запроса.

var pairs = query.split("&"); // Разделение строки по амперсанду

for (var i = 0; i < pairs.length; i++) {

var pos = pairs[i].indexOf('='); // Проверка пары "name=value"

if (pos == -1) { // Если не найдено - пропустить

continue;

}

var argname = pairs[i].substring(0, pos); // Получение имени

var value = pairs[i].substring(pos + 1); // Получение значения

args[argname] = value; // Сохранение как свойства

}

return args;

}

// Для теста примера перейдите по URL текущей страницы + ?login=admin&password=123

function print() {

var args = getQueryString();

var e = document.createElement("p");

var a = parseInt(args.a);

var b = parseInt(args.b);

var rez = a + b;

e.innerHTML = "a = " + args.a + "+ b =" + args.b +" = "+rez+"<br />";

document.body.appendChild(e);

}

</script>

</head>

<body>

<input type="button" name="name" value="Sum" onclick="print()" />

Задание 5

Создайте бегущую строку. Пользователь вводит текст в поле ввода и нажимает на кнопку. После чего в элементе страниц, отображается бегущая строка с введенным текстом.

window.onload = function ()

{

var x = 0;

var textInput = document.getElementById('text1');

var button = document.getElementById('btn1');

var running = document.getElementById('str');

running.style.position = "absolute";

var handler;

button.onclick = function ()

{

clearInterval(handler);

running.innerHTML = text1.value;

handler = setInterval(function () { if (x > screen.width) { x = -10;} Run(x++);},10);

}

function Run(x)

{

running.style.left = x + 'px';

}

}

</script>

</head>

<body>

<input id="text1" type="text" name="texBox" value="" />

<input id="btn1" type="button" name="go" value="Go" />

<div id="str"></div>

# Css

Преимущества CSS:

*  Разграничение кода и оформления
*  Разное оформление для разных устройств
*  Расширенные по сравнению с HTML способы форматирования
*  Ускорение загрузки сайта
*  Единое стилевое оформление множества документов

Способы подключения стилей:

*  Связанные стили – в отдельном css файле, которые связан со страницей через элемент link добавленный в head документа
*  Глобальные стили – создание стиля на странице в элементе style. Менее гибкий способ чем предыдущий, так как CSS правила доступны только одной странице.
*  Внутренние стили (inline) – стили определенные в атрибуте style конкретного элемента разметки.
* Dynamic HTML (DHTML) – это способ создания интерактивного веб-приложения при помощи комбинирования статической HTML разметки, клиентского JavaScript, CSS стилей и манипуляции DOM.
* Для того что бы через JavaScript изменить оформление элемента следует использовать следующий код:
* [элемент].[style].[имя\_css\_атрибута] = “значение”
* Например:
* e.style.color = “green”;
* e.style.width = “100px”;