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”;
* 1. Что такое CSS?
* 2. Назовите способы использования CSS на странице.
* 3. Какие преимущества дает использование CSS?
* 4. Что такое DHTML?
* 5. Какие свойства CSS часто используются в DHTML?
* 6. Опишите процесс смены фона элемента div при нажатии по кнопке.

Разработайте игру «Угадай значение». Страница загадывает число от 1 до 100. Пользователь в поле ввода вводит значение пытаясь угадать загаданное число. Если пользователь не угадывает значение страница выводит сообщение с текстом «загаданное значение больше введенного вами» или «загаданное значение меньше введенного вами». Подумайте как можно применить DHTML в таком приложении.

<script>

var max=100;

var min = 1;

var randNumber;

var myNumber;

var flag = false;

window.onload = function ()

{

randNumber = Math.floor(Math.random() \* (max + min + 1)) + min;

while (!Checker(randNumber));

}

function Checker(randNumber)

{

var number = prompt("Put number between 1 - 100", "1");

var flag = false;

if (number < randNumber)

{

alert("Your number is less than random number!")

}

if (number > randNumber) {

alert("Your number is greater than random number!")

}

if (number == randNumber)

{

alert("You got it! Congratz")

flag = true;

}

return flag;

}

</script>

Задание 2

Создайте страницу для авторизации. На странице должны находиться поля вводов для логина и пароля, кнопка «Вход» и checkbox «Запомнить меня». С помощью CSS, который расположен в отдельном файле, выровняйте форму для авторизации по центру страницы. Разработайте сценарий, который будет проверять срабатывать по нажатию на кнопку «Вход». Сценарий должен реализовывать следующее поведение:

*  Если при нажатии на кнопку поля ввода пустые – выводиться сообщение (в теле страницы, не через alert) «Вы не заполнили поля логин и пароль», также поля вводов должны получить красный фон.
*  Если введен логин admin и пароль 12345 то отобразить пользователю зеленым цветом сообщение «Вы авторизированы».

<script>

window.onload = function ()

{

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

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

var Login =

{

login: document.getElementsByTagName('input')[0],

logTip: document.getElementById('logTip')

};

var Password =

{

password:document.getElementsByTagName('input')[1],

passTip: document.getElementById('passTip')

}

button.onclick = function ()

{

AuthCheck(Login, Password,status);

}

Login.login.onclick = function()

{

Login.login.style.backgroundColor = "white";

Login.logTip.innerHTML = "";

}

Password.password.onclick = function () {

Password.password.style.backgroundColor = "white";

Password.passTip.innerHTML = "";

}

}

function AuthCheck(Login,Password,status)

{

if(Login.login.value === "")

{

Login.login.style.backgroundColor = "#ff9999";

Login.logTip.innerHTML = "Login field is empty"

Login.logTip.style.color = "red";

}

if(Password.password.value ==="")

{

Password.password.style.backgroundColor = "#ff9999";

Password.passTip.innerHTML = "Password field is empty"

Password.passTip.style.color = "red";

}

if (Login.login.value == 'admin' && Password.password.value == 12345)

{

status.style.color = "green";

status.innerHTML = "You are succesesfully sign up";

}

else

{

status.style.color = "red";

status.innerHTML = "Login or password does't input properly";

}

}

</script>

<style type="text/css">

input {

margin-top:10px;

}

div {

display:inline-block;

}

h1 {

font-family:'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;

}

</style>

</head>

<body>

<input type="text"> <div id="logTip"></div> <br>

<input type="password"> <div id="passTip"></div> <br>

<input type="checkbox" checked ="checked">

<input id="btn" type="button" value="Sign in">

<h1 id="status"> </h1>

</body>

</html>

Задание 3

Разработайте сценарий, который каждую секунду будет случайным образом менять положение трех элементов div. Задайте элементам фиксированную ширину и высоту и запустите сценарий при загрузке страницы.

<title>Ламав бред</title>

<script>

window.onload = function ()

{

var divs = document.getElementsByTagName('div');

setInterval(ChangePosition, 1000);

function ChangePosition()

{

var divs = document.getElementsByTagName('div');

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

{

var pos = (Math.floor((Math.random() \* 1000+1) )- 1)+'px';

divs[i].style.left = pos;

pos = (Math.floor((Math.random() \* 600 + 1)) - 1) + 'px';

divs[i].style.top = pos;

}

}

}

</script>

<style>

div {

background-color:green;

height:5px;

width:5px;

margin:10px 10px;

position:absolute;

}

</style>

</head>

<body>

<div id="ds"></div>

<div></div>

<div></div>

</body>

</html>

# EventListener

Особенности распространения событий в DOM Level 2.

События проходят через три этапа:

* 1. Этап перехвата – событие распространяется от корневого элемента (Document) до элемента в котором произошло.
* 2. Этап обработки в целевом узле – событие происходит в элементе, который его инициировал.
* 3. Этап всплывания – событие поднимается по дереву элементов от целевого элемента к корневому.

В модели событий Internet Explorer есть только два этапа обработки события – в целевом объекте и всплывание.

**stopPropagation()** – метод, который можно вызвать на объекте события для того что бы прекратить его распространение на этапе перехвата или всплывания.

**preventDefault()** – метод для отмены действия по умолчанию, связанного с событием. Например, если этот метод вызвать в обработчике события submit формы, то браузер не выполнит отправку данных формы на сервер.

* 1. Назовите события, которые были изучены на уроке.
* 2. Какие модели обработки событий есть в JavaScript?
* 3. Какие способы присвоения обработчиков событий в DOM Level 0 Вы знаете?
* 4. Как отменить действие по умолчанию связанное с событием в DOM Level 0?
* 5. Как отменить действие по умолчанию связанное с событием в DOM Level 2?
* 6. Назовите этапы распространения события в модели DOM Level 2

Разместите на странице два поля ввода и кнопки для арифметических операций над данными введенными в поля ввода. Реализуйте данную страницу двумя способами – с использованием модели DOM Level 0 (через свойства) и DOM Level 2

<!DOCTYPE html>

<html>

<head>

<title>Ламав бред</title>

<script>

window.addEventListener('load', function () {

var field1 = document.getElementsByTagName('input')[0];

var field2 = document.getElementsByTagName('input')[1];

field1.value = 0;

field2.value = 0;

var buttons = document.getElementsByClassName('btn');

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

{

buttons[i].addEventListener("click", function () {

ExecuteOperation(field1.value, field2.value, this.attributes[2].value)

}, false);

}

},

false

);

function ExecuteOperation(val1, val2,operator)

{

document.getElementById('log').innerHTML = operator;

var field3 = document.getElementsByTagName('input')[2];

if (operator == '+')

field3.value = Add(val1, val2);

if (operator == '-')

field3.value = Minus(val1, val2);

if (operator == '\*')

field3.value = Multy(val1, val2);

if (operator == '/')

field3.value = Devide(val1, val2);

}

function Add(val1,val2)

{

var rez = parseInt(val1) +parseInt(val2);

return rez;

}

function Devide(val1, val2) {

var rez = parseInt(val1) / parseInt(val2);

return rez;

}

function Multy(val1, val2) {

var rez = parseInt(val1) \* parseInt(val2);

return rez;

}

function Minus(val1, val2) {

var rez = parseInt(val1) - parseInt(val2);

return rez;

}

</script>

<style>

input {

width:40px;

margin-top:5px;

}

span {

margin-left:5px;

margin-right:5px;

}

</style>

</head>

<body>

<input type="text"> <span id="log">?</span> <input type="text"> = <input type="text"> <br>

<input class="btn" type="button" value="+"> <input class="btn" type="button" value="-"> <input class="btn" type="button" value="\*"> <input class="btn" type="button" value="/">

</body>

</html>

# Events 2

Интерфейс объекта Event и значения свойств зависят от браузера выполняющего сценарий.

На странице не может одновременно произойти несколько событий, все события выстраиваются в очередь, соответственно в момент выполнения сценария может обрабатываться только одно событие, которое доступно через свойство window.event или через параметр метода обработчика.

Свойства объекта события для браузеров поддерживающих стандарт DOM Level 2

Интерфейс Event:

type - тип события, например “click”

target – элемент в котором произошло событие.

currentTarget – узел в котором в данный момент обрабатывается событие

eventPhase – 1 – этап перехвата, 2 - этап обработки в целевом элементе, 3 - этап всплывания

timestamp – дата и время, когда произошло событие.

Bubbles – всплывает ли данное событие.

Cancelable – связанно ли событие с действием по умолчанию.

События мыши:

onmousedown

omousemove

onmouseout

onmouseover

onmouseup

События клавиатуры:

onkeydown

onkeypress

onkeyup

* 1. Какие свойства интерфейса Event Вы запомнили?
* 2. Как можно получить объект события в сценарии?
* 3. Назовите события связанные с мышью.
* 4. Назовите события связанные с клавиатурой.
* 5. В чем разница между событиями onkeyup и onkeypress?

Forms

Form – элемент разметки, который позволяет отправить данные введенные пользователем на сервер. Обязательный атрибут элемента form – action. С помощью атрибута action можно указать адрес, на который должны быть отправлены данные введенные пользователем в элементы формы. Все элементы формы представлены тегом input. В зависимости от значения атрибута type элемент может менять визуальное представление.

Возможные значения для type:

Button

Checkbox

File

Hidden

Password

Radio

Reset

Text

Submit

Кроме элемента input форма может содержать элементы textarea, select, option.

Для того что бы данные из элементов корректно обрабатывались на стороне сервера, каждый из элементов формы должен иметь уникальное значение атрибута name.

Для получения доступа к формам на странице используется document.forms[индекс]

Для получения доступа к элементам формы используется код document.forms[индекс].elements[индекс]

Если элемент имеет имя обращение к элементу может быть организовано через код форма.имя\_элкмента.

Свойства элементов формы:

type – строковое значение с типом элемента

form – ссылка на форму в которой элемент объявлен

name – имя элемента

value – значение элемента.

События, которые используются с элементами:

onclick

onchange

onfocus

onblur

* 1. Для чего используется элемент form?
* 2. С помощью каких элементов разметки можно создать элементы управления для формы?
* 3. Назовите основные свойства элемента формы.
* 4. Какое событие происходит при нажатии указателем мыши по полю ввода?
* 5. Назовите способы получения элементов размещённых в форме.
* 6. Опишите принцип проверки данных в форме.
* 7. Какое событие происходит в форме при отправке данных на сервер?
* 8. Как создать элемент управления, который отправит все данные из формы на сервер?
* 9. Как создать элемент управления, который сбросит все значения введенные в форме?
* 10. Какой атрибут обязательный для корректной работы элемента form?
* 11. Какой атрибут обязательный для корректной работы элемента input?

# Cookies

Cookie – небольшой набор данных сохранённых веб-браузером и привязанных к определенной странице или сайту.

Для получения доступа к cookie наборам необходимо выполнить инструкцию document.cookie.

Каждый cookie набор характеризуется именем и значением, а также необязательными атрибутами:

*  Expires – дата, когда cookie набор будет считаться не действительный или max-age время жизни набора в секундах.
*  Path – страница или директория на сайте к которой относятся cookie наборы.
*  Domain – позволяет задать сабдомены, на который можно будет отправлять cookie наборы.
*  Secure – при значении true cookie набор будет отправлен на сервер только при HTTPS протоколе.

Для создания cookie набора с определенными параметрами следует использовать следующий синтаксис name=value; max-age=seconds; path=\admin;

При использовании cookie наборов действует ограничение на объем данных, которые будут сохранены. Размер данных не может превышать 4Kb.

Session Storage - Значение существует до тех пор, пока открыта вкладка или окно, в которой они сохранены. Значения видимы в окне или вкладке в которой они были созданы

Local Storage - Значения существуют независимо от закрытия окна или вкладки.

Значения совместно используются всеми окнами и вкладками, выполняющимися на том же источнике.

Лимит на данные для Session и Local Storage составляет 5 Mb. Данное поведение может завесить от браузера.

**Закрепление материала**

* 1. Что такое Cookie?
* 2. Как можно прочитать или записать cookie для страницы?
* 3. Какие дополнительные атрибуты можно задать для cookie набора?
* 4. Какое ограничение на размер данных для cookie набора?
* 5. Что такое sessoinStorage и в чем его отличие от localStorage?
* 6. Какое размер данных позволяют сохранить localStorage и sessionStorage?