**1.Въведение в JavaScript**

DHTML=HTML+CSS+JavaScript+DOM

-интерактивни уеб сайтове-сайтове, в които се зарежда динамично съдържание. Те са раздвижени, взаимодействат си с user-а

JavaScript:

-скриптови езици-много по свободни езици от статично- типизираните, по-бавни в сравнение със статично-типизираните езици

-няма компилация!

-можем runtime да променяме обектите

-има интерактивна конзола-т.е. виждаме какъв е резултата от това, което сме написали на момента

С JavaScript можем да:

-handle-ваме дадено събитие(напр. някой е натиснал някъде с мишката, направил е някоя клавишна комбинация и др.)

-печатаме по HTML документа

-пишем сървърна логика

-може да се използва като обектно-ориентиран език

JavaScript engine-и-програми, които изпълняват JavaScript

-могат да са част от уеб браузъра, а могат и да са отделно(NodeJS)

JS engine-виртуална машина, която изпълнява JavaScript код. Важното е да знам, че при различните engine-и JavaScript се държи по различен начин. Например при един engine string-а се държи по един начин, а при друг engine се държи по друг начин

Може да зареждаме script-ове async-хронно и defer-нато

1. (async(!!!само за външни скриптове)!!!) -например скрипта е 160kb. На един телефон времето за зареждане на тези 160kb е например 40 секунди. В момента,в който този скрипт се изтегли(т.е. той е available), се изпълнява

2.-(defer)-след като се зареди целия документ и се парсне, чак тогава се изпълнява

Event handling-ако искаме да хванем onclick event-а на картинка казваме <img src=”…” onclick=”alert(‘Clicked’)”/>

Синтаксис-подобен на този на C#

!!!Променливите са обекти и в тях можем да сложим каквото си поискаме

var answer=confirm("Do you want to continue");-излиза прозорче, което те пита нещо и ти му отговарящ с „да” или „отказ“

var name=prompt("What is your name?");-показва се прозорче, в което въвеждаме дадена информация

## document.body.style.background=color(цвят записан в 16-тична бройна система);-така променяме фона на дадена страница

## [2.JavaScript syntax](http://pastebin.com/gezXdgP9)

## Тип данни-множество от стойности от един и същи тип

## Var-ключова дума за дефиниране на променливи в JavaScript

## В JavaScript няма различни типове числа. Има само един тип и той е Number. При деление на положително число с 0 се получава Infinity, а на отрицателно с 0 се получава –Infinity

## В JavaScript има garbage collector

## Езикът JavaScript е typeless език защото типовете данни са скрити, т.е. няма изрично типизиране, но не е вярно, че няма типове данни в JavaScript

## JavaScript не е “strongly typed” език както са C# и Java

## Floating-point числата използват 15-16 цифрена прецизност

## Като напишем var t = new Number(5);

## След това t в конзолата на даден сайт

## Ни връща Number{ }-т.е. типът на обекта t

## Object-„майката“ на всички данни. Обектите представляват двойки ключове и стойности

## Undefined-означава, че дадено свойство на даден обект не е дефинирано

## Има local variables и global variables(не бива да се използват). Разликата е, че глобалните променливи могат да се използват в методи, въпреки, че те не се дефинират в дадените методи.

## !функциите са на практика обекти в JavaScript

## Ако дефинираме дадена променлива на дадено място, то тя започва да съществува и на горните редове

## Оператори:

## === - сравнява по стойност и тип

## !== - прави обратното на ===

## !!! ако използваме оператора % се връща floating-point число

## На if-овете в JavaScript не е задължително да подаваме true или false стойности, може и числа(0-та се счита за false, а всички числа по-големи от 0 – за true)

## new - използва се за създаване на нови обекти

## typeof - връща типа на обекта, който искаме да проверим

## Ако напишем в конзолата: а без да сме декларирали или инициализирали тази променлива със стойност, то ни изписва ReferenceError: a is not defined(т.е. тази променлива липсва)

## Ако просто декларираме променливата а: var a/var a=undefined то тогава а е undefined(т.е. не съществува стойността, но променливата съществува)

## Може стойността на а да е null: a=null/var a=null. Тогава а е обект, но обект без стойност

## Biggest num: var biggestNum=Number.MAX\_VALUE;

## Smallest num: var smallestNum=Number.MIN\_VALUE;

## Convert floating-point number to integer number:

## var valueDouble=8.75;

## var valueInt=Math.floor(valueDouble);//8

## Convert to integer number with rounding:

## Var valueDouble=8.75;

## Var valueInt=Math.round(valueDouble);//9

## Convert from string to integer

## var str="1234";

## var i=Number(str);//1234

## празният стринг е равен на 0, т.е. 0=””

## break-ваме от цикъл, който се намира в друг цикъл по следния начин:

## search:for (var i = 0; i < 2; i++) {

## for (var j = 0; j < 10; j++) {

## console.log(j);

## if(j==5)

## break search;

## }

## }

## Values converted to false:

## 0

## “0”

## “”

## [ ]

## Values converted to true:

## 1,2...

## “1”,“2“...

## null не е равно нито на true нито на false

## [3.Цикли, масиви, стрингове](http://pastebin.com/jKE8t70k)

## В JavaScript има 4 вида цикли:

## 1.while loop:

## While(condition) {

## Statements;

## }

## 5,”non-empty” are evaluated as true

## 0,”” are evaluated as false

## 2.do while loop:

## Do{

## Statements;

## }while(condition);

## 3.for loop:

## for (var i = 0; i < 10; i++) {

## …

## }

## !!! i += 1 е малко по-бързо от i++

## !!!можем да имаме повече от един брояч

## 4.for-in loop-итерира по свойствата (ключовете )на обектите, и по индексите на масивите:

## var arr = [10, 20, 30, 40, 50];

## for (var index in arr) {

## console.log(arr[index])

## }

## Iterating over the properties of an object:

## var obj = { name: 'Steve', age: 23, location: 'Sofia' };

## for (var key in obj) {

## console.log(obj[key]);

## }

## // Steve, 23 , Sofia

## Arrays

## Създаване на масив:

## var numbers=[1,2,3,4,5];

## Елементите на масивите не е задължително да са от един и същи тип!

## Using new Array(initialLength):

## var arr = new Array(10); // [undefined × 10]

## методи при масивите:

## array.unshift(element)- inserts a new element at the beginning of the array

## array.push(element)-appends a new element at the end

## array.shift()-removes and returns the element at the beginning of the array

## array.pop()-removes and returns the last element

## сортиране на масив

## var numbers = [5, 4, 23, 2];

## numbers.sort(function(a, b) {

## return a > b;

## });

## console.log(numbers.join(', ')); // returns 2, 4, 5, 23

## array.reverse()-връща нов масив с елементите в обратен ред

## arr.slice(start,end-1)-връща елементите на масива от start позиция до end позиция

## array.concat(elements)-прилепя елементи в края на масива

## array.join(separator)-прилепя елементите на масива с конкретен елемент

## array.filter(function(item){ return true / false })

## Returns a new array with the elements that satisfy condition

## 

## array.forEach(function(item){ … })

## Iterates through the array and executes the function for each item

## array.indexOf(element)

## Returns the index of the first match in the array

## Returns -1 if the element is not found

## array.lastIndexOf(element)

## Returns the index of the first match in the array from right to left or -1 (not found)

## Асоциативнни масиви:

##  Associative arrays are arrays indexed by keys, not by the numbers 0, 1, 2, 3, …

##  Initializing an associative array (object):

## var prices = { 'orange' : 2.30, 'apple' : 1.50, 'tomato' : 3.80 };

##  Taking the keys of object / array:

## var prices = { 'orange' : 2.30, 'apple' : 1.50, 'tomato' : 3.80 };

## console.log(Object.keys(prices)); // ["orange", "apple", "tomato"]

## var nums = [10, 20, 30];

## console.log(Object.keys(nums)); // ["0", "1", "2"]

## в JavaScript обектите са напрактика асоциативни масиви, но с тази разлика, че обектите нямат дължина

## Strings

## String.search(regex(регулярен израз))—спрямо регулярния израз получаваме началние индекс на 1вото срещане на string-а ,който търсим

## Конкатенацията е бърз процес в JavaScript

## !когато пишем в HTML ако искаме да напишем даден таг без да искаме да се рендира трябва да escape-ваме самите символи

## Conversion from primitive to object type:

##  new String('…') creates a string object

##  String(strObject) creates a primitive string

## [4.Функции и обекти](http://pastebin.com/pvfLAzYa)

## Функция-парче код, който върши някаква работа, може да й се подават параметри, може да връща стойност. Прави кода по-четим.

## С функциите се имплементира обектно-ориентираното програмиране, защото в JavaScript няма чиста дефиниция на клас, на виртуален метод и др.

## Functions can be defined in several ways:

## 1.By function declaration:

## function printHello() {

## console.log('Hello')

## };

## 2.By function expression:

## var printHello = function() {

## console.log('Hello')

## }; - анонимна функция

## var printHello = function printFunc() {

## console.log('Hello')

## };

## ^

## |

## В горните 2 варианта записваме в някакви променливи дадени функции

## 3.Using the constructor of the Function object:

## var printHello = new Function('console.log("Hello")');

## стек(stack) – представлява нещо като ракла, в която вкарваме данни. Пълним отдолу нагоре и когато искаме да вземем от тази „ракла“ взимаме последните вкарани данни(т.е. тези отгоре).

## Разлика между параметри и аргументи:

## Параметри- променливите, които използваме при декларацията на дадена функция

## Аргументи-стойностите, които подаваме на функцията при нейнотото извикване

## Throw-ване на exception: throw “message”

## Нормално е на дадена функция да й подаваме като параметър друга функция, защото функциите са променливи, в които вместо стойност има сорс код

## Immediately-Invoked Function Expression (IIFE)-използва се когато искаме да скрием променливите в дадената функция. Това става като създаваме анонимна функция и след това я извикваме:

## (function() {

## for (var x = 1; x < 5; x++) {

## var y = x\*x;

## console.log(x + " " + y);

## }

## })();

## console.log(typeof(x) + " " + typeof(y)); // undefined undefined

## в JavaScript няма overloading

## Обекти – нещо, което го има в реалния свят. Например куче, казва се Шаро, има черен цвят,кучето се движи, лае и т.н.

## Обектите може да са неща от реалния свят(банкови сметки, хора и т.н.) и може да са неща от компютърния свят(числа, символи, масиви и др.)

## Класовете в JavaScript се наричат обектни типове-това са образци, по които създаваме обекти

## В JavaScript обектите са на практика колекция от property-та. Не са подобни на обектите в C# и Java!

## !!!ако искаме да направим клонинг на даден елемент използваме JSON.stringify:

## var nakov={fname:’Svetlin’, lname:’Nakov’,age:25};

## var pesho=JSON.parse(JSON.stringify(nakov));

## JSON.stingify ни стрингосва даден обект

## JSON.stringify(nakov) ни връща следния резултат:

## “{“fname”:”Svetlin”, “lname”:”Nakov”,”age”:”25”}”; т.е. ни стрингосва обекта

## Когато преобразуваме обект в JSON ние записваме само данните му(без функциите му(ако има такива)). Следователно когато го парснем на обратно пак записваме само данните и функциите ги няма. Например:

## var nakov={fname:’Svetlin’,lname:’Nakov’);

## nakov.printFName=function() {return this.fname};

## nakov;

## var nakov={fname:’Svetlin’,lname:’Nakov’, function());

## след като стрингосаме обекта nakov, функциите изчезват

## има 6 вида стойности в JavaScript: number,string,Boolean,null,undefined,object