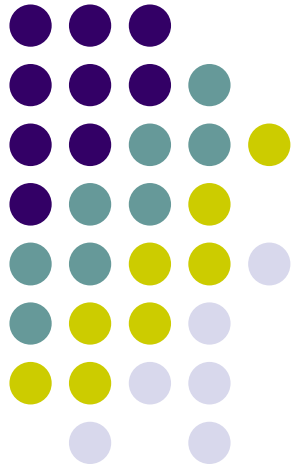
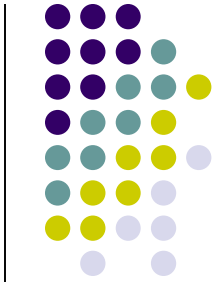
Мрежово програмиране

JavaScript



Съвременните хипертекстови информационни системи условно могат да бъдат представени във вид на съвкупност от няколко компонента:



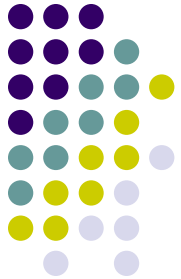
- системи за *съхраняване* на хипертекстови обекти;
- системи за *визуализация* на хипертекстови обекти;
- системи за *подготовка* на хипертекстови обекти;
- системи за *програмиране* на визуализацията на съвкупносттта от хипертекстови обекти.

От тази гледна точка технологията World Wide Web чак към 1996 година е получила функционално напълно завършен вид

- Първи са били разработени системите за съхранение и визуализация (1989 - 1991 г.), които продължават да се развиват и днес.
- След 1990 година са започнали да се появяват първите системи за подготовка на документи.
- През 1995 година са били предложени първите езици за управление на визуализацията.

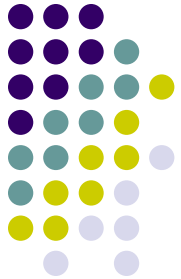
Практически всички локални хипертекстови системи в една или друга степен имат програмни средства за манипулиране с хипертекстови обекти

- В редица случаи цялата хипертекстова база данни може да бъде представена като една голяма програма, в която хипертекстовите възли са програмни модули, а връзката между тях е предаване на управлението от един модул на друг.

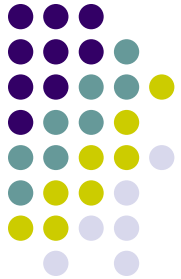


Предимствата на този подход пред традиционното статично представяне са очевидни:

- гъвкавост на построяването на хипертекстова мрежа;
- възможност за създаване на програми за скролиране на фрагменти от БД;
- генерация на съставни хипертекстови обекти от съществуващи елементарни компоненти.

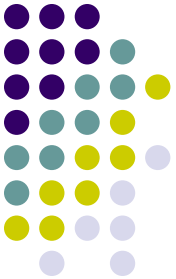


- Динамичните обекти могат да бъдат лесно получени от статичните, понеже в случая на съществуването на програма за визуализация системата може да бъде приведена от интерактивен режим на визуализация на хипертекстова БД в пакетен, когато действията на оператора ще бъдат заменени с команди на програмата.



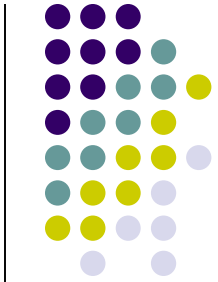
- Програмите за визуализация на хипертекстови страници традиционно се наричат скриптове по аналогия с изпълнимите файлове, написани за командни интерпретатори от типа sh.

При програмирането на визуализацията на хипертекстови документи съществуват два подхода, които се базират на обектно ориентирания подход в програмирането



- създаване на интерпретируеми от програмата за визуализация скриптове
  - следва традициите на WWW, според които за разработването на хипертекстова страница е необходим само обикновен текстов редактор и самия хипертекстов документ трябва лесно да се чете
- компилация на байт-код
  - позволява да се увеличи ефективността от изпълнението на програмата и защитеността на кода от несанкционирана модификация

ПРИЛОЖЕНИЯ, ПЛЪГИНИ, СКРИПТОВЕ

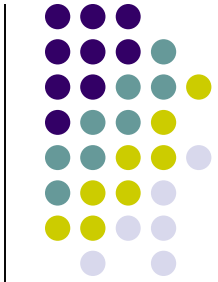


- Плъгин (plug-in) — независимо компилируем програмен модул, динамично свързван към основната програма, предназначен за разширяване или използване на нейните възможности. Обикновено се изпълняват във вид на разделяеми библиотеки;
- Скрипт (script) — програма, която автоматизира някаква задача, която потребителят изпълнява ръчно, като използва интерфейсите на програмата.

Разликата между приложенията и скриптовете е доста размита. Скриптът е програма, която има работа с готови програмни компоненти.

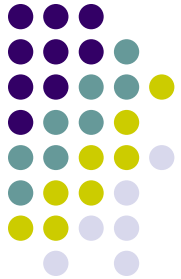
Скриптовете в терминологията на някои програми се наричат макроси.

Плъгините, работещи с апаратното осигуряване традиционно се наричат драйвери.



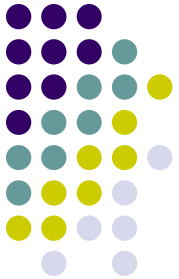
Аплетът е специализирана малка програма на Java с ограничени възможности, работеща в прозореца на WWW документа под управлението на браузъра. Като правило аплетите се вграждат в HTML документи.

- Докато приложението се стартира непосредствено от потребителската машина и има достъп до всички ресурси
- Аплетът се зарежда от WWW от външен сървър и може да предава данни на всеки сървър в WWW
- Аплетите нямат достъп до файловата система на локалната машина, т.е. аплетите нямат много от възможностите на приложенията



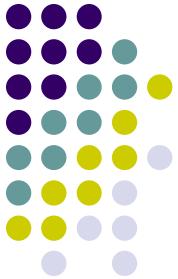
```
. . .
<title>Пример за използване на аплет в
   HTML</title>
<body>
<applet code="my_applet.class" width=200
   height=100>
<param name=YourBorn value="1992">
</applet>
</body>
. . .
```

JavaScript (JS)

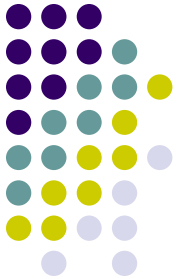


- JS е прототипно ориентиран скриптов език за програмиране
- Реализация (диалект) на езика ESMAScript (стандарт ECMA-262)
- Цели интерактивност на уеб страниците
- Лесен е за използване от непрограмисти
- Създаден е в Netscape през 1995 г.
- През 1996 г. Microsoft анонсира аналог на езика JS, наречен JScript

Елементи на ядрото на JavaScript



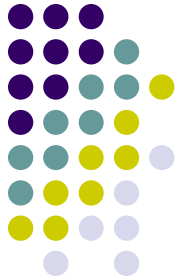
- Ключови думи
- Синтаксис за оператори и граматика
- Правила за написване на изрази, променливи и литерали
- Базов е обектния модел (макар че клиентският и сървърният JavaScript имат различни предопределени обекти)
- Предопределени обекти и функции, такива като Array, Date и Math



Сравнение на клиента и сървъра

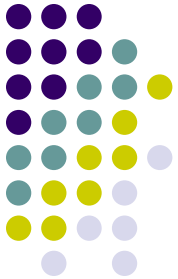
- Сървърите обикновено (не винаги) са високопроизводителни работни станции с бързи процесори и с възможност за съхраняване на големи обеми информация.
- Клиентите обикновено (не винаги) са настолни системи с маломощни процесори и относително неголям обем на съхранявани данни.

Сравнение на клиента и сървъра



- Сървърите могат да бъдат претоварени при едновременен достъп на хиляди клиенти.
- Клиентите често пъти са машини с един потребител, затова може да се предаде част от натоварването на процесинга на клиента.

Клиентски процесинг (таг SCRIPT)



- Проверка на потребителското въвеждане
- Заявка за потвърждение от потребителя и визуализация на диалогови прозорци или информационни кутии
- Изпълнение на агрегатни изчисления или друг процесинг на данните, заявени на сървъра
- Условна обработка в HTML
- Изпълнение на други функции, не изискващи информация от сървъра

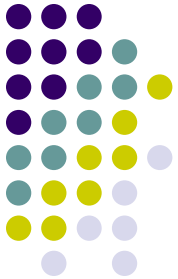
Сървърен процесинг (таг SERVER)

- Обслужване на серия от клиентски заявки
- Работа с данни, разпределени при няколко клиента или приложения
- Достъп до БД или файлове на сървъра
- Извикване на външни библиотеки на сървъра
- Динамична специализация на Java аплети

Синтаксис на JavaScript

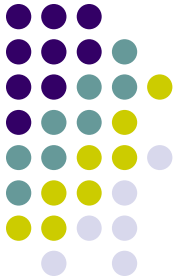
- Езикът JavaScript по синтаксис е близък до езиците C/C++, Java, PHP и другите C - подобни езици.
- Това се вижда при коментариите, операторите за присвояване, циклите и условните оператори.
- Обаче, въвеждането/извеждането напомня Visual Basic.Net (операторът alert прилича на MsgBox).
- Друга разлика от C е слабата типизация и автоматичното преобразуване на типовете.

Java и JavaScript



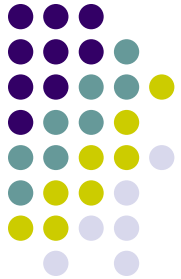
- JavaScript – скриптов език. Интерпретируем
- JavaScript – нетипизиран
- JavaScript по-прост език от Java

JavaScript и Java
JavaScript

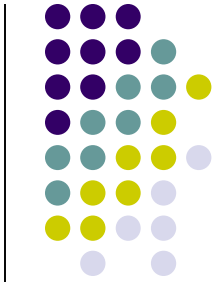


- Интерпретира се (не се компилира) от клиента.
- Обектно-ориентиран. Няма разлики между типовете обекти. Наследяването става чрез механизма на прототипите, а свойствата и методите могат да се добавят към обекта динамично.
- Кодовете са интегрирани и внедрени в HTML.
- Типовете на променливите не се обявяват (динамична типизация).

Java



- Компилираните байт-кодове, заредени от сървъра, се изпълняват на клиента.
- На базата на класовете. Обектите се делят на класове и екземпляри, наследяващи по цялата верига на йерархията на класовете. Класовете и екземплярите не могат да имат свойства и методи, добавени динамично.
- Аплетите се различават от HTML (достъпът до тях се реализира от HTML страници).
- Типовете на променливите задължително трябва да са обявени (статична типизация).

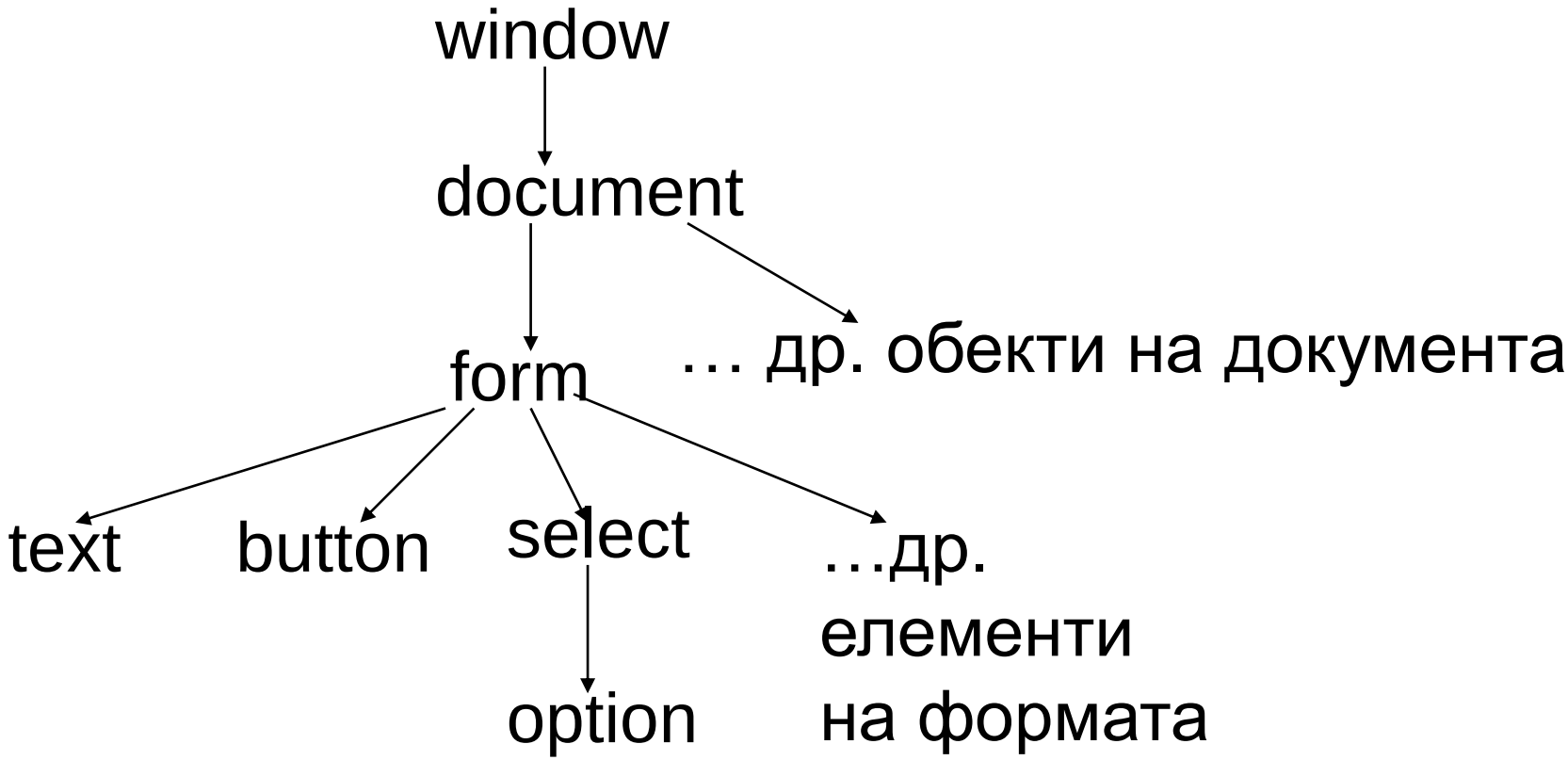


Управление на визуализацията на страниците на уеб възела

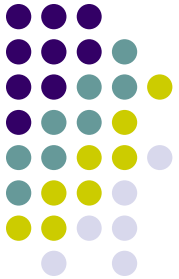
- Модел на обектите в JavaScript - обекти на Netscape Navigator

  Комбинирайки браузърния и документния обектни модели, JavaScript позволява да се манипулира с всички елементи на страницата като обекти от прозореца надолу по йерархията. Освен тези класове обекти потребителят може да създава и свои собствени.

DOM – Обектен модел на документа



```
<HTML>
<HEAD>
<TITLE>Динамично създаване на уеб страница</TITLE>
</HEAD>
<BODY>
<H1>JavaScript test</H1>
<SCRIPT LANGUAGE="JavaScript">
document.write(“Този текст е генериран динамично " +
“от програмата JavaScript");
</SCRIPT>
</BODY>
</HTML>
```



Резервирани думи

- if
- else
- for
- while
- break
- continue
- with
- function
- return
- var
- null
- void
- typeof
- true
- false
- new
- delete
- this
- in

Синтаксис

Коментари:

// това е коментар

/* това също е
коментар */

Числа:

1, 3.5, -2.45e+23
010
0xff, 0x100

Стрингове:

‘Низ от знаци‘

‘Още един "низ"'

“Пак \ "стринг\""

Променливи:

var x;
var y=0;

Масиви:

mas = new Array();
x = mas[0]; y = m[i][j];

Изрази и операции

Стрингови: + '(' + a + ',' + b + ')'

Логически: ==, !=, <, >, <=, >=,&& - и, || - или

Аритметични: ++, --, -, ~, *, /, %, <<, >>, >>>, &, ^,

|,*=, /=, %=, +=, -=, <<=, >>=, >>>=, &=, ^=, |=

= – присвояване, ?: – условен оператор

Масиви: [] – елемент на масив

Функции: () – извикване на функция

Обекти: . – поле на обект

Математически функции

- Предопределен обект Math
- cos(), acos(), sin(), asin(), tan(), atan(), atan2(), min(), max(), pow(), log(), exp(), round(), ceil(), floor(), random(), abs(), sqrt()
- E, PI, LN10, LN2, LOG10E, LOG2E

- Math.sqrt(x)

Променливи

Поддържат се пет примитивни типове данни (числови, стрингови, логически, празни и неопределени) и един съставен, обектен (Object).

Променливите се обявяват с var.

При създаване на променлива тя получава стойност undefined

- var str;
- alert(str);
- var num = 100500;
- var message = "JavaScript“;
- var COLOR_RED = "#F00";
- var array = [];
- var obj = {};
- var ex=123;

  Операторът delete не може да изтрие променлива, обявена с помощта на var;

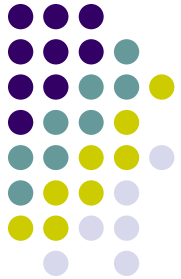
  затова ако променливата е била обявена с var, тогава единствен начин тя да се изтрие е да й се присвои стойност null или undefined.
- ex=null;

  // или
- ex=undefined;

Елементи за управление

- Това са интерактивни обекти, позволяващи получаването на данни от потребителя.
- Тяхното предназначение и външен вид са идентични с елементите на потребителския интерфейс на съвременните операционни системи с графичен интерфейс (бутони, полета и боксове за въвеждане, чекбоксове и т.н.).

Събития



- *onLoad* - изпълнение на скрипта или функцията при зареждане;
- *onChange* – поражда се при изменение стойността на елемента на формата;
- *onClick* – поражда се при избор на обект (button, checkbox и т.н.);
- *onSelect* – поражда се при избор на текстов обект (text, textarea);
- *onSubmit* - при натискане на бутон Submit;
- *onUnload* - при преход на друга страница.

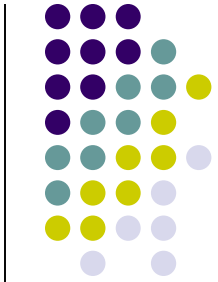
Например

MakeOnLoad

MakeOnUnload

```
. . .
<BODY onLoad="MakeOnLoad()"
   onUnload="MakeOnUnload()">
. . .
```

Обхождане на списъци



```
var prime = [2,3,5,7];
for(i in prime){
    console.log(i);
}
```

Резултат:

0

1

2

3

```
for(i in prime){
    console.log(prime[i]);
}
prime.forEach(function(number){
    console.log(number);
}
```

Резултат:

2

3

5

7

Функции

```
function test1( message ) {
    alert( message );
}
function test2() {
    alert(’Test’);
}
```

JavaScript не е чисто функционален език

Обекти

x=a.field; - поле на обект

a.method();- извикване на метод

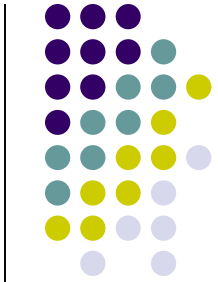
new «конструктор» - създаване на обект

```
new Array();
 var person = {
            name: “Ivan”,
            age: 21 }

 person.name=“Ivan”
 person.address = “Sofia”

 person
 { name: “Ivan”,
   age: 21,
   address: “Sofia”}
```

ООП

```
function Person(age, name) {
            this.age = age,
            this.name = name; }
Person.prototype.getName(){
            return this.name; }
function Student(fn) {
            this.fn = fn; }
Student.prototype =
        Object.create(Person.prototype)
```

Използване на масиви



Създаване:

- a = new Array(5, 4, 3, 2, 1, "testing");
- a = new Array();
- a = new Array(10);

Методи и свойства:

- length
- join()
- reverse()
- sort(), sort(“функция“)

Масивите могат да съдържат разнотипни елементи. Също така, елементите могат да бъдат асоциирани както с номера, така и със стрингове.

Оператори

Цикъл

for («иниц.»; «условие»; «стъпка») «оператор»;

for («prop» in «object») «оператор»;

Примери:

```
for(i=1;i<10;i++) { x=x+i; }
m=new Array(); … for(a in m) {alert(a);}
```

while («условие») «оператор»;

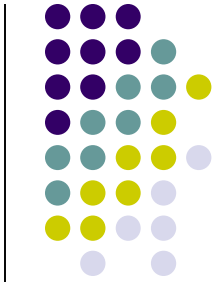
with

Преобразуване на типовете

- 'стойност x=' + x

- метод toString()
- метод valueOf()
- функции за стрингове parseInt(str), parseFloat(str)

Разполагане на JavaScript в отделен файл



```
<SCRIPT src="extern.js"></SCRIPT>
```

Файл extern.js:

```
document.write("А това е JavaScript! <br> ");
```

Атрибути на елемента SCRIPT:

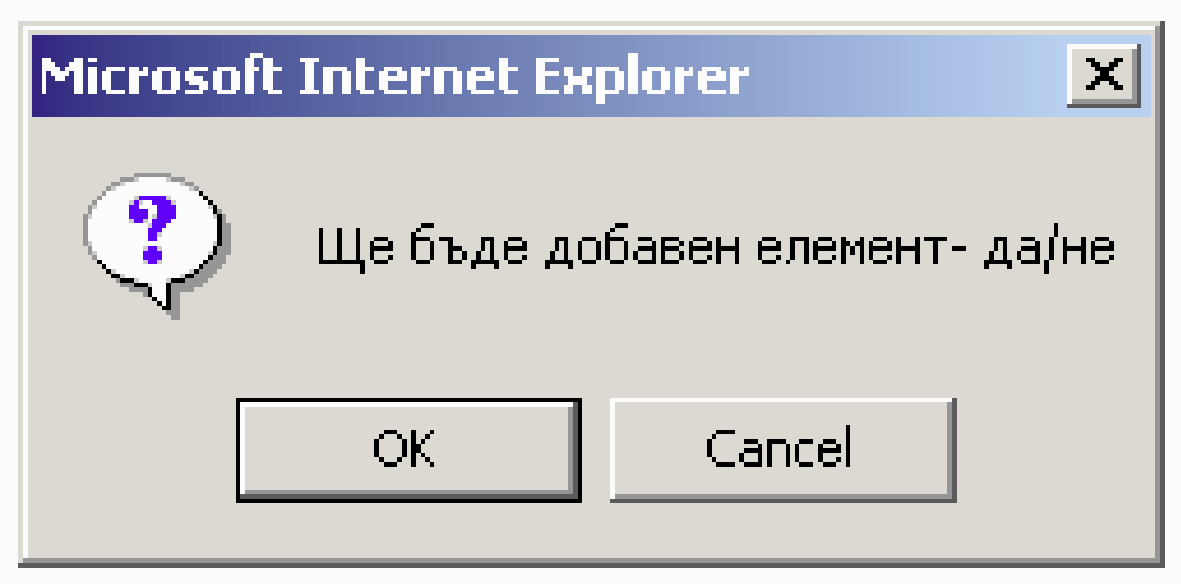
- charset
- type ("text/javascript", "text/vbscript")
- language
- src

alert – извежда съобщение за предупреждение и бутон

```
<HTML><HEAD><TITLE> Предупреждение </TITLE> </HEAD>
 <BODY>
 <SCRIPT>
 { alert ("This is a warning !! ") ;}
 </SCRIPT>
 </BODY></HTML>
```

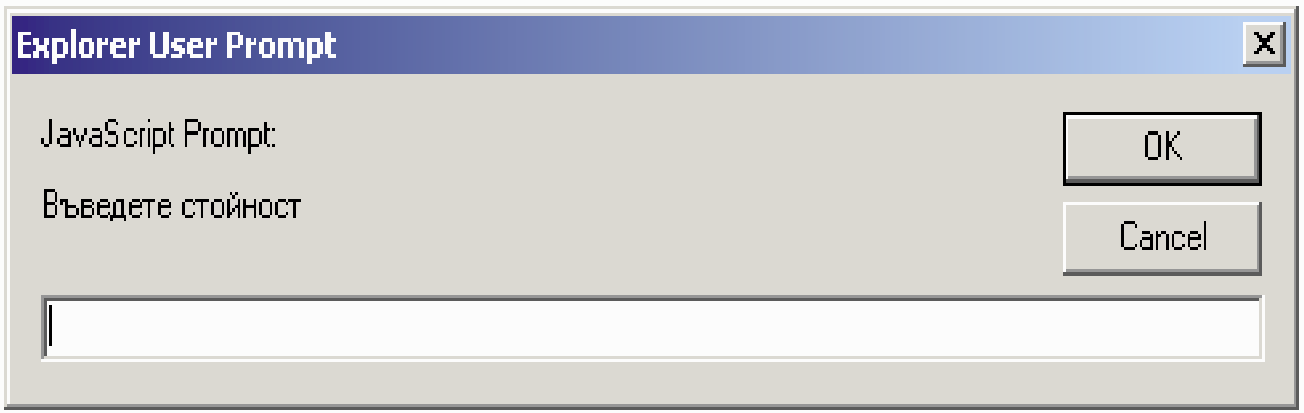
confirm – извежда съобщение и два бутона

```
<HTML><HEAD><TITLE>Confirm </TITLE></HEAD>
<BODY>
<SCRIPT>log=confirm(“Ще бъде добавен елемент – да/не”) ;
</SCRIPT>
</BODY></HTML>
```



prompt – извежда съобщение, поле за текст и два бутона

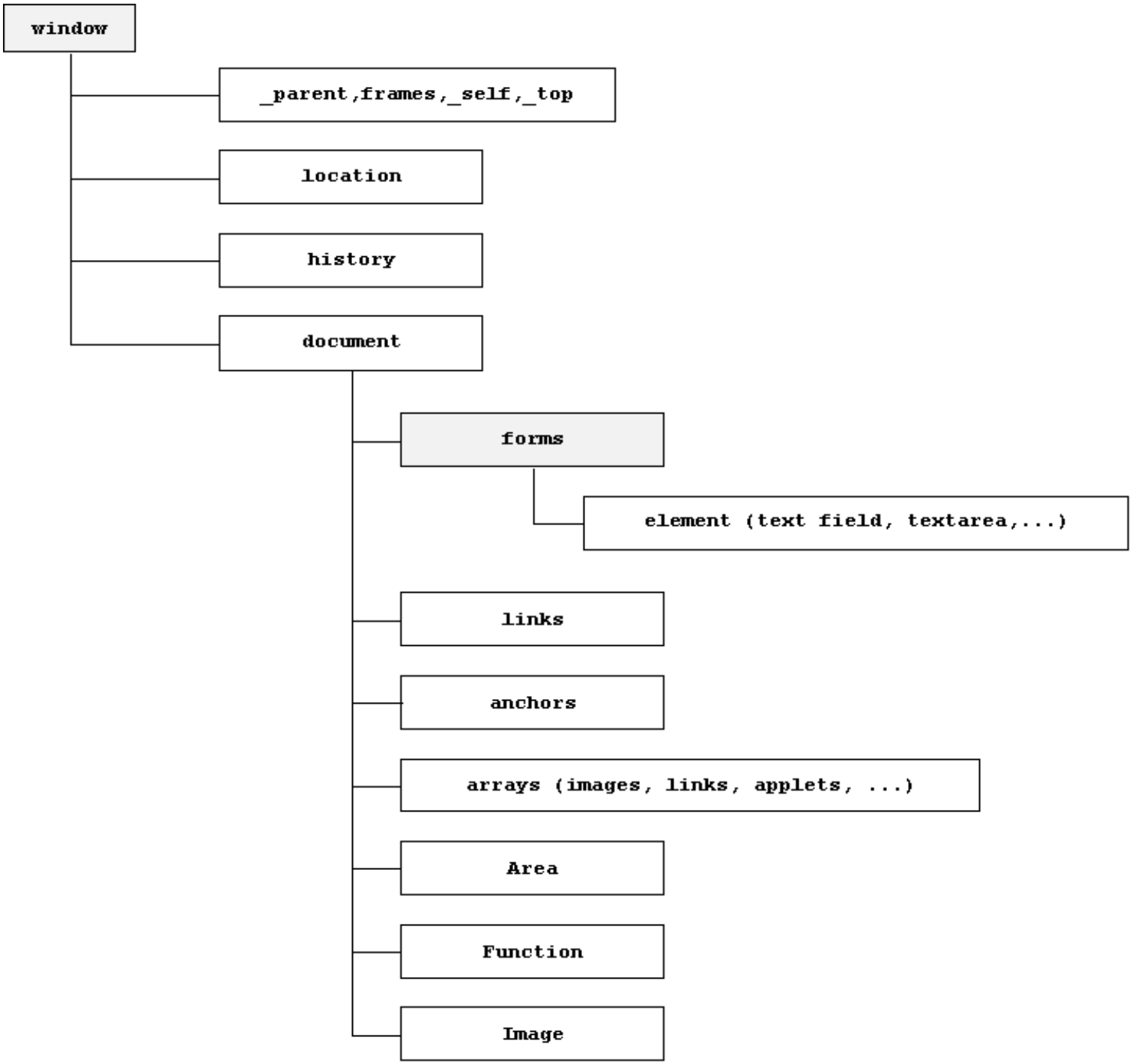
```
<HTML>
 <HEAD>
 <SCRIPT> var s=new String (" ") ;
</SCRIPT>
<TITLE> Prompt </TITLE>
</HEAD>
 <BODY>
 <SCRIPT>prompt(“Въведете стойност”,s) ; </SCRIPT>
</BODY>
</HTML>
```



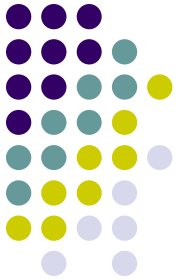
Тестване на скриптовете

- alert(«стринг»)
- document.write(«стринг»)

Методът document.write() позволява да се извежда в прозореца на браузъра списък от текстови стрингове, разделени от запетаи. Ако извежданият стринг представлява HTML код, тогава браузърът ще изобрази резултата от неговата интерпретация, а не последователност от тагове. С други думи, HTML таговете на извежданото съобщение ще се възприемат от браузъра като команди, а не като текстови символи.

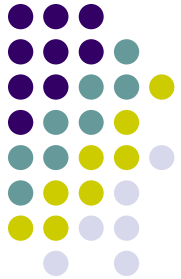

window
_parent,frames,_self,_top
location
history
document
forms
element (text field, textarea,...)
links
anchors
arrays (images, links, applets, ...)
Area
Function
Image

Обект Window

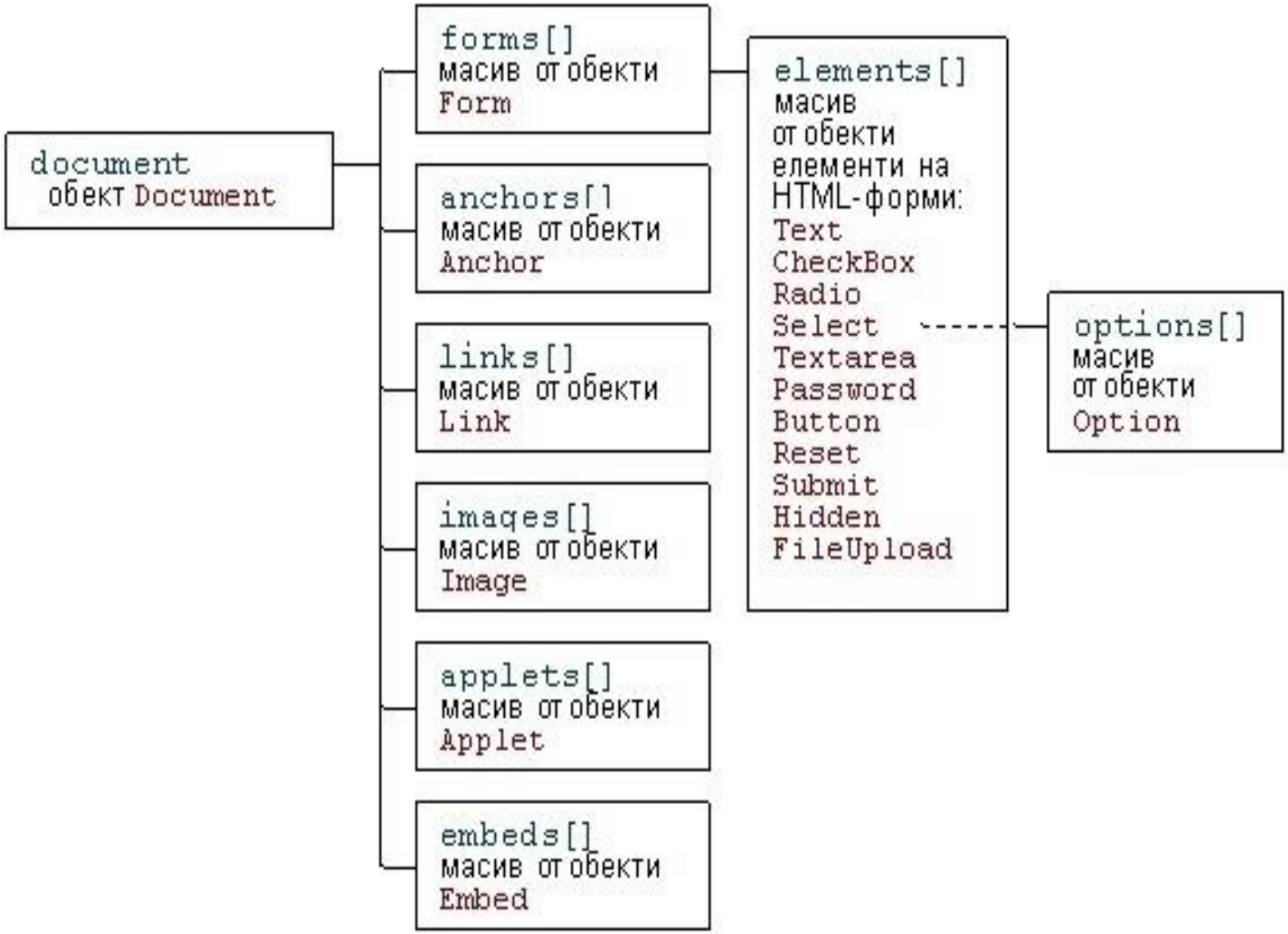


- Той представлява прозорец или кадър на браузъра. Понеже кодът на JavaScript се интерпретира в контекста на обекта Window, в който се изпълнява, този обект трябва да съдържа препратки към всички необходими обекти на JavaScript (т.е. е корен на йерархията от обекти на JavaScript). Много от свойствата на обекта Window сочат към други важни обекти.
- За обръщение към обекта Window не е необходим специален синтаксис и към свойствата на този обект може да се отнасяме като към променливи.

Обект document



- Това е абстрактна структура от данни, представляваща пълно описание на уеб страницата.
- Наборът от свойства и методи на този обект позволява да се управлява както поведението на уеб страницата като цяло, така и на отделните й обекти (елементи за управление, препратки, текстови блокове, изображения и т.н.).
- Достъпът до свойствата и методите е реализиран чрез стандартни програмни интерфейси.

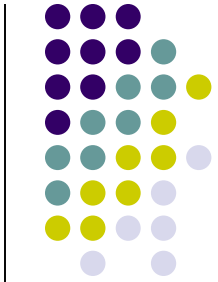

document
обект Document
forms[]
масив от обекти
Form
anchors[]
масив от обекти
Anchor
links[]
масив от обекти
Link
images[]
масив от обекти
Image
applets[]
масив от обекти
Applet
embeds[]
масив от обекти
Embed
elements[]
масив
от обекти
елементи на
HTML-форми:
Text
CheckBox
Radio
Select
Textarea
Password
Button
Reset
Submit
Hidden
FileUpload
options[]
масив
от обекти
Option

Свойства на обекта Document

Свойствата са общи за всички браузъри.

Повечето от тях са достъпни както за четене, така и за промяна.

- title — текст на заглавието на документа (съдържимото на елемента title);
- fgColor и bgColor — цвят на текста и цвят на фона на документа;
- linkColor, vLinkColor, aLinkColor — цветове на непосетените, на посетените и на активните препратки;
- lastModified (само за четене) — дата на промяната на документа;
- referrer (само за четене) — адрес на източника за преход;
- URL, location — собствен адрес на документа.



Свойства-масиви на обекта Document. Всички те имат свойството length (брой на елементите в масива). Повечето свойства, специфични за обектите, съхранявани в тези масиви, се асоциират с атрибути на съответстващите елементи на HTML:

- обектът Anchor (котва) има единствено свойство name;
- обектът Link (препратка) има свойства href, target;
- обектът Image (изображение) има свойствата src, width, height.

Към обектите на документа, съхранявани в масивите images, controls и други, а също така и към елементите на формите е възможно обръщение по име (свойство name) или по идентификатор (свойство id). Например ако в документа има описание <img src="images/cat.jpg" id="cat_id" name="cat_name"> и това е n-тото изображение в документа, тогава към елемента img е възможно обръщение по следните начини:

- Като към елемент на масива images по индекс (индексацията започва с 0): window.document.images[n-1]
- Като към елемент на хеш-масив images по ключ (стойността на name като ключ на масива): window.document.images["cat_name"]
- Използвайки стойността на атрибута name, като свойство на обекта: window.document.cat_name
- Използвайки стойността на атрибута id и свойството getElementById: window.document.getElementById("cat_id")

Методи на обекта Document

- open() — отваря нов документ, при което цялото му съдържимо се изтрива.
- close() — затваря преди това отворения документ.
- write() — записва в документа зададен в качеството на аргумент стринг.
- writeln() — аналогичен е на предишния, но тук стринга завършва със символ преход на нов ред.
- Методите write() и writeln() често се използват за динамично формиране на съдържимото на документа.

Например за да се включи в документа датата на последната му промяна:

```
<script>document.write(document.lastModified);</script>
```

Разполагане на JavaScript на html страница

- Програми на JavaScript в елемента Script

```
<HTML><BODY>
Това е обикновен HTML документ.<br>
 <SCRIPT language="JavaScript">
  document.write("А това е JavaScript! <br> ")
 </SCRIPT>
Пак документ HTML.
</BODY></HTML>
```

- В хипертекст

```
<A HREF="javascript:alert('It is an empty link')">link</A>
```

- В качеството на обработчици на събития

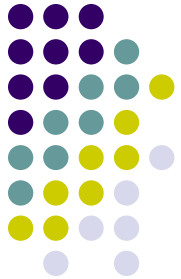
```
<DIV OnClick="alert('I am clicked')">Click me</DIV>
```

Фреймове и прозорци

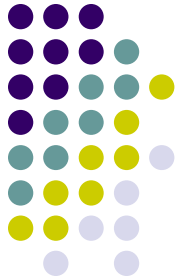
- свойство opener
- методи blur и focus

При работа с прозорци и фреймове в системата е въведено свойството opener, което е определено за текущия прозорец или фрейм, и методите blur и focus.

Това свойство може да се използва при обръщение към обектите страници-родител, което позволява да се компенсира отсъствието на наследяване и на глобални променливи в JavaScript.



- За създаването на програми на JavaScript е необходим само браузър, поддържащ езика JavaScript съответната версия и текстов редактор, позволяващ създаване на HTML документи.



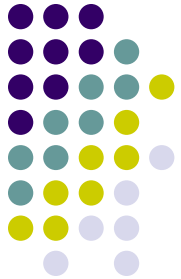
Листинг. Управление на прозорци (използване на обекта window)

```
<html>
<head>
<title>Отваряве/затваряне на нов прозорец</title>
</head>
<body>
<p><a name="demoOpen" onClick="mywindow =
window.open('window.htm','mywin','height=120, width=300, left=100,
top=30');">Да се отвори</a>
<a name="demoClose" onClick="mywindow.window.close();">Да се
затвори</a>
</body>
</html>
```

Листинг. Извеждане на текущото време (използван е вградения обект Date)

```
<html>
<HEAD>
<TITLE>Часовник, показващ текущото време</TITLE>
<script type="text/javascript">
function fulltime() {
var time=new Date();
document.clock.full.value=time.toLocaleString(); // вариант 1
document.getElementById("jsclock").innerHTML=time.toLocaleString(); // вариант 2
setTimeout('fulltime()',500) }
</script>
</head>
<body>
<form name=clock>
<input type=text size=20 name=full><!– вариант 2 -->
<span id="jsclock"></span><!-- вариант 2 -->
</form>
<script type="text/javascript"> fulltime(); </script>
</BODY>
</HTML>
```

Популярност



- JavaScript е най-популярният език за програмиране, използван за разработването на уеб приложения на страната на клиента.
- Лидиращата позиция на JavaScript е свързана с развитието на AJAX, понеже на практика браузърът се е превърнал в система за доставка на приложения.
- Растящата популярност на JavaScript е свързана с това, че езикът се вгражда в приложенията.

- В Ajax приложението част от приложната логика се пренася на браузъра.
- След регистрацията на потребителя клиентското приложение се доставя на браузъра. На много от действията на потребителя това приложение е способно да реагира самостоятелно. Ако наличните възможности не достигат, тогава то предава заявките на сървъра без да прекъсва последователността на действията на потребителя.
- При регистрацията на потребителя на браузъра се предоставя по-сложен документ, съществена част на който е код на JavaScript. Този документ остава достъпен за потребителя по време на целия сеанс, при което в зависимост от действията на потребителя си променя външния вид.
- Клиентската програма знае как да реагира на въвежданите данни и е способна да преценява дали да ги обработва самостоятелно или да изпраща заявка към сървъра (който може да се обърне към БД или към друг ресурс).