Мрежово програмиране

HTML + CGI

доц. д-р Йордан Денев
denev@fmi.uni-sofia.bg

HTML

- Standard Generalized Markup Language (SGML) – метаезик, който дефинира език за описание на структурата на дадено семейство документ.
- Document Type Definition (DTD) – стандартна SGML дефиниция на език за даденото семейство документи.

- HTML - език, дефиниран чрез SGML за семейството документи, предназначено за опсисание на хипермедиа документи, използвани във WWW .
- Cascading Style Sheets (CSS) – стандарт за описване на стилове, използвани в HTML документи. HTML таговете, описващи даден стил са предвидени за премахване в бъдеще (deprecated).

- Extensible Markup Language (XML) – широко изплзвано подмножество на SGML.

- Extensible Hypertext Markup Language (XHTML) – версия на HTML, която е XML коректна.

- Markup language – tag's structure

  The majority of HTML tags belong to a *container* tag set. A container tag set consists of a start tag delimited by "<" and ">" and an end tag, delimited by "</" and ">." In some cases, the end tag is optional.

SYNTAX

```
<ELE [ATTRIB1[="val1"]] [ATTRIB2[="val2"]]
  ... [ATTRIBx[="valx"]]>
      Content
</ELE>
```

EXAMPLE

```
<FORM ACTION="url.cgi" METHOD=post> ...
  </FORM>
```

- HTML document structure

1. Head

   Starts with <head> and ends with</head>

2. Body

   Starts with <body> and ends with </body>

3. Typical structure

```
<html>
<head> …..</head>
<body>….. </body>
</html>
```

HTML basics

- Elements of the head part,
  - <title> *text* </title>

Define the document title:

```
<head>
  <title>Title of the document</title>

</head>
```

- Elements of the body
  - Headings

    Headings are defined with the <h1> to <h6> tags.

```
<h1>This is a heading</h1>
<h2>This is a heading</h2
```

  - Line breaks

Line breaks are defined with the
 tags

This is the line
This is the next line

  - Paragrahps

Paragraphs are defined with the <p> tag.

```
<p>This is a paragraph</p>
```

  - Horisontal line
    <hr> tag draws horisontal line.

- HTML Text Formatting Tags

  <b> Defines bold text

  <big> Defines big text

   Defines emphasized text

  <i> Defines italic text

  <small> Defines small text

  Defines strong text

  <sub>Defines subscripted text

  <sup> Defines superscripted

   Defines centered text

- An image tag

The <img> tag embeds an image in an HTML page. It has two required attributes: src and alt.

```
<img src="angry.gif" alt="Angry face" >
```

- The link tag

A hyperlink (or link) is a word, group of words, or image that you can click on to jump to a new document or a new section within the current document.

```
<a href="url">Link text</a>
```

- An HTML example

```
<HTML>
<HEAD>
<TITLE> An HTML example </TITLE>
</HEAD>
<BODY>
<h3> HTML Text Formatting Tags</h3>
   <br><b> Defines bold text</b>
   <br><big> Defines big text</big>
   <br><em> Defines emphasized text</em>
   <br><i> Defines italic text</i>
   <br><small> Defines small text</small>
   <br><strong>Defines strong text</strong>
   <br>X<sub>Defines subscripted text</sub>
   <br>Y<sup> Defines superscripted</sup>
<p> <center> <a href = "example2.html”> Skip to next page
   </a> </center></p>
<body>
```

HTML Programming

- Client side processing
  - Java applets;
  - JavaScript;
- Server side processing
  - CGI;
  - PHP;
  - Java Tools – servlets, JSP, JEE
- Data base connections

Client Side Processing

- Изпълнението на програмируемите модули се извършва в средата на потребителския агент.
- Java applets can run in a Web browser using a Java Virtual Machine (JVM), or in Sun's AppletViewer, a stand-alone tool for testing applets. Java applets were introduced in the first version of the Java language in 1995. Java applets are usually written in the Java .

- Активиране на аплет – HTML tag

```
< APPLET
   CODE = appletFile
   WIDTH = pixels
   HEIGHT = pixels
   [ALIGN = alignment]
   [VSPACE = pixels]
   [HSPACE = pixels]
>
[< PARAM NAME = appletParameter1 VALUE = value >]
[< PARAM NAME = appletParameter2 VALUE = value >]
   . . .
</APPLET>
```

HTML page example:

```
<HTML>
<HEAD></HEAD>
<BODY>
<P> An Java Simple Applet example
<br>
<APPLET CODE = "Simple.class‟  WIDTH = 300 HEIGHT =
  300 >
</APPLET>
<br><hr><br>
<a
  href="http://java.sun.com/docs/books/tutorial/depl
  oyment/applet/index.html">
Read More for Applets</a>
</BODY>
</HTML>
```

- Жизнен цикъл на аплета:

```
import java.awt.*;
import java.applet.Applet;
public class AppletStructure extends Applet {
  public void init() {
  System.out.println("initializing"); } // init
 public void start() {
  System.out.println("starting"); }// start public
 public void paint(Graphics g) {
  System.out.println("painting");
  g.drawString("Hello World!", 30, 30); } // paint
 public void stop() {
  System.out.println("stopping"); }// stop
}
```

init () method: The life cycle of an applet is begin on that time when the applet is first loaded into the browser and called the init() method. The init() method is called only one time in the life cycle on an applet. The init() method is basically called to read the PARAM tag in the html file..

start () method: This method may be called multiples time when the Applet needs to be started or restarted.

stop () method: The stop() method can be called multiple times in the life cycle of applet like the start () method.

destroy() method: The destroy() method is called only one time in the life cycle of Applet like init() method. This method is called only on that time when the browser needs to Shut down.

❑ JavaScript

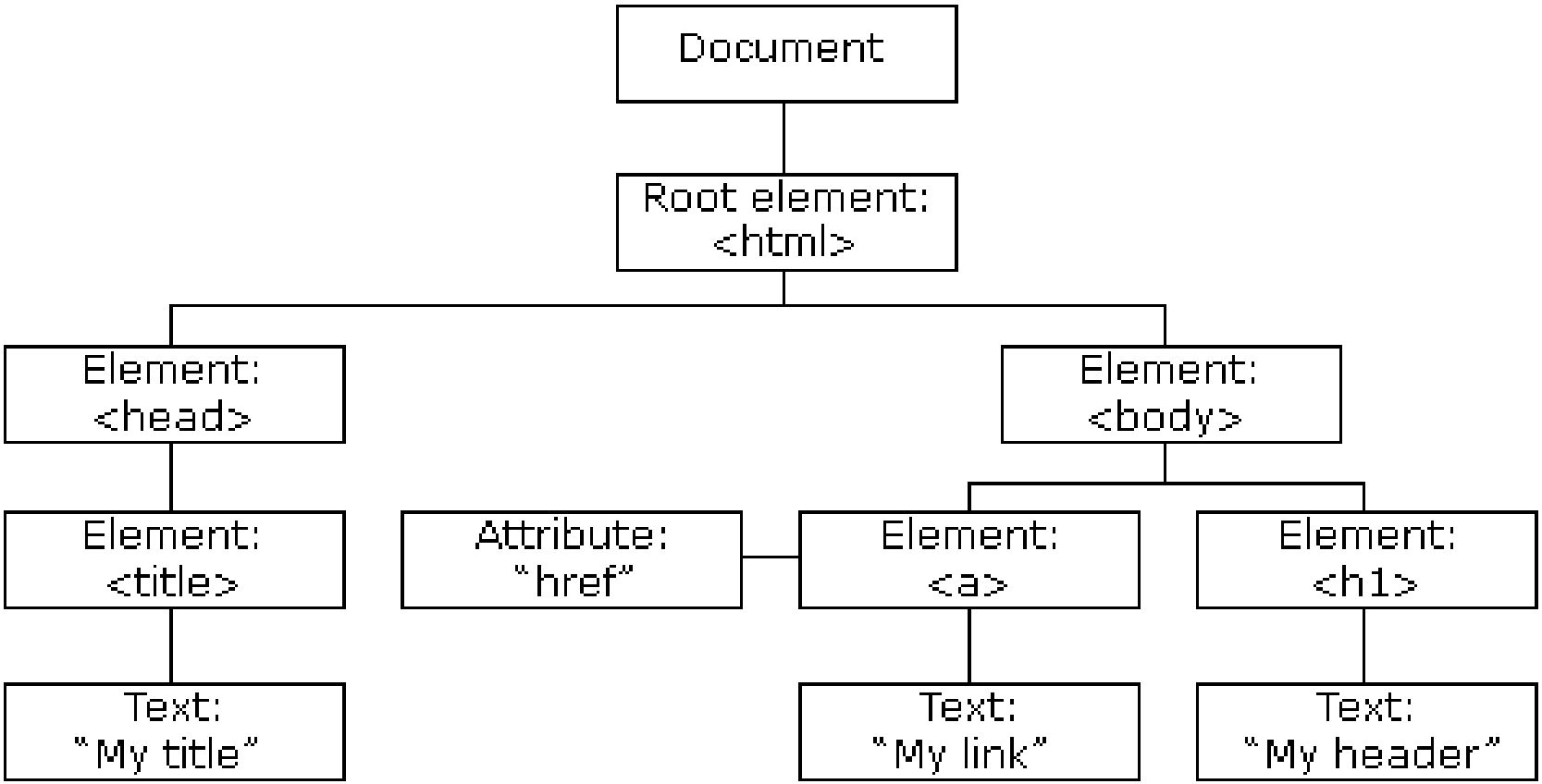
- JavaScript was designed to add interactivity to HTML pages JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation) .
- Everyone can use JavaScript without purchasing a license

- JavaScript използва HTML DOM.
  - The HTML Document Object Model (HTML DOM) defines a standard way for accessing and manipulating HTML documents.
  - The DOM presents an HTML document as a tree-structure (a node tree), with elements, attributes, and text.
  - the DOM will provide you with methods and properties to retrieve, modify, update, and delete parts of the document you are working on

According to the DOM, everything in an HTML document is a node.

The DOM says:

- The entire document is a document node
- Every HTML tag is an element node
- The text in the HTML elements are text nodes
- Every HTML attribute is an attribute node
- Comments are comment nodes



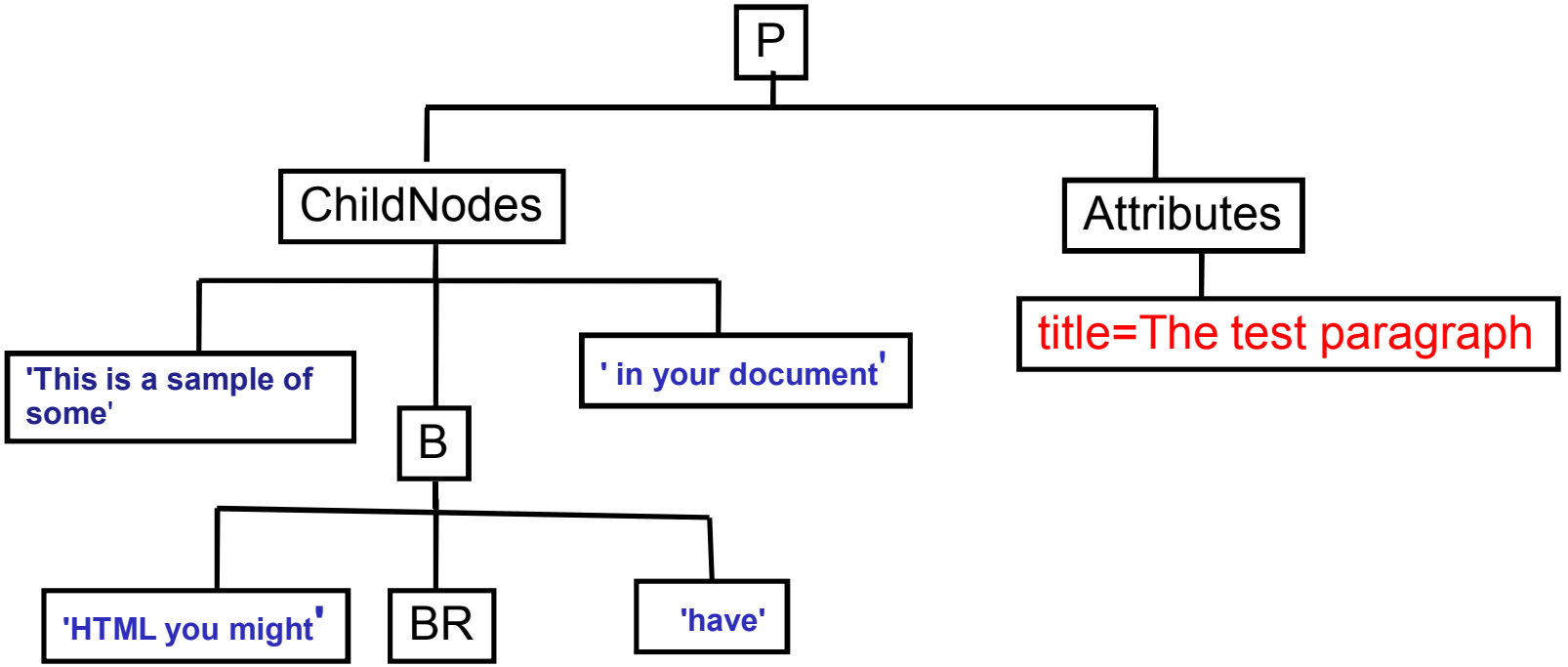
Какво реферира `window.document.childNodes[0].childNodes[1]`?

- Пример

```
<html>
<body>
<script type="text/javascript">
<!-
document.write("Hello World!");
//-->
</script>
</body>
</html>
```

Пример:

```
<p title="The test paragraph">This is a sample of
   some <b>HTML you might<br>have</b> in your
   document</p>
```



- Синтаксисът на JavaScript наподобява C или Java.
- JavaScript НЕ е Java.
- JavaScript е език със свободна типизация, променливите не се декларират, а получават тип от контекста, в който се използват.

- Flash

  Flash is a multimedia plug-in and standalone application for playing Flash movies. A flash movie is authored in Macromedia Flash.

Server Side Processing

Програмните модули се изпълняват на сървера и генерират HTML съдържание, което сървърът изпраща към клиента.

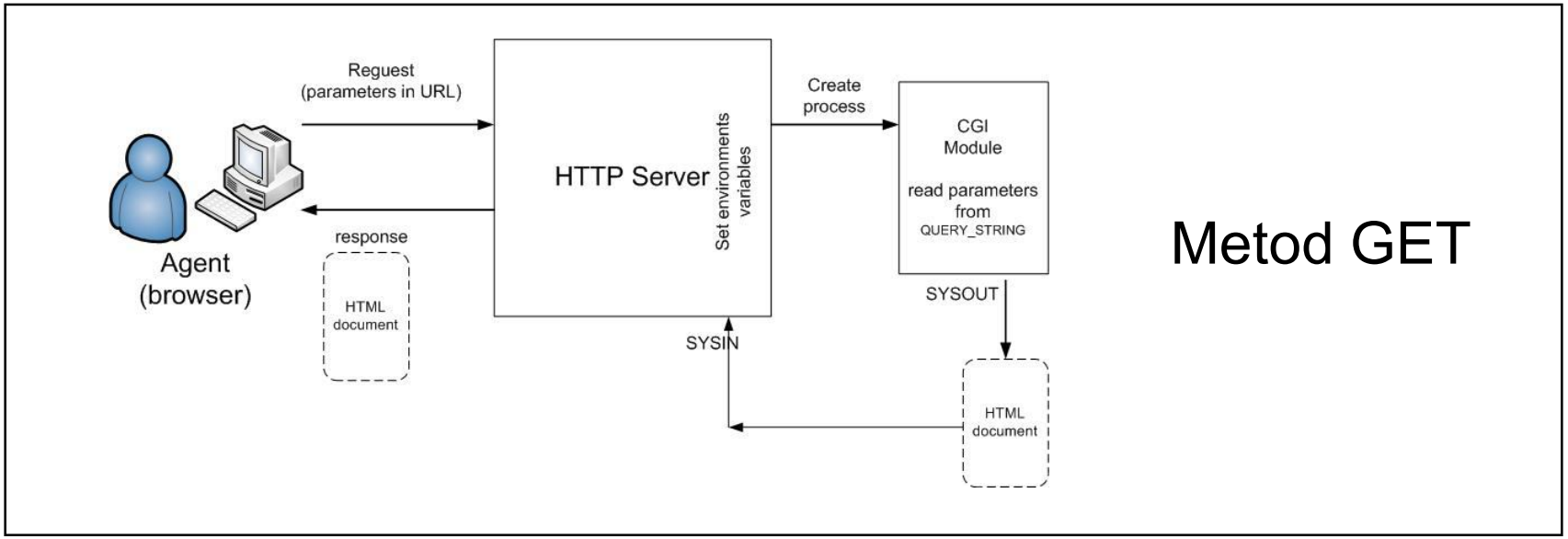
Common Gateway Interface (CGI)

- HTTP request-а може да реферира два вида ресурси:
  1. Файл от файловата система на сървера. Тогава сървърът връща съдържанието на този файл.
  2. Изпълним модул (програма, командна процедура, скрипт,...). Тогава сървърът приема информация, която предава на изпълнимия модул и връща неговият изход.

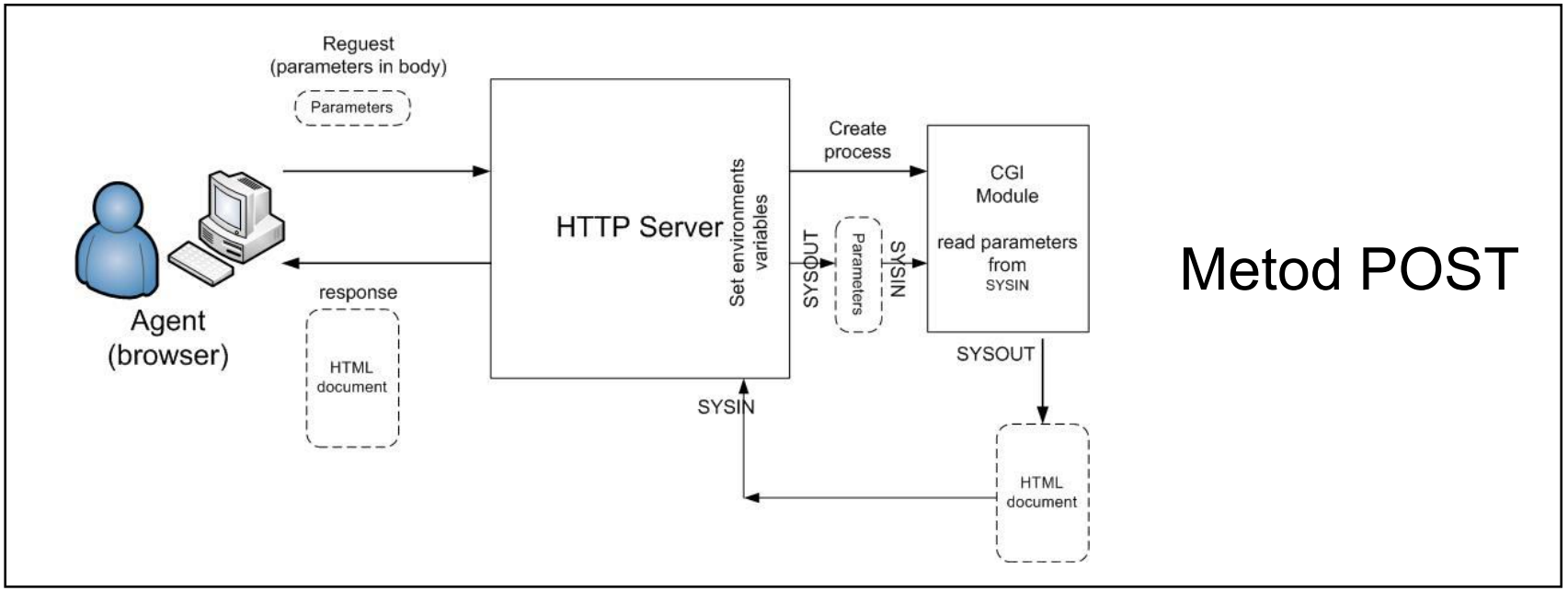
- CGI дефинира стандартите, които се следват във втория случай - как сървърът предава на изпълнимият модул изпратената от агента информация и как сървърът може да добави допълнителна информация към изпратения изход (нпр. чрез header-ите).

CGI методи

- GET – информацията от агента се включва в URL-а като параметри. CGI модулът намира тази информация в променлива на обкъжението.
- POST - информацията се изпраща в тялото на request-а, а CGI модулът е чете от стандартния си вход.



Metod GET



Metod POST

CGI реализация

1. CGI модулът се активира чрез създаване на нов процес.
2. Процесът получава изпратената от агента информация съгласно на дефинираният метод.
3. Процесът произвежда стандартен изход, който се изпраща обратно на агента.
4. Процесът завършва своята работа.

Оптимизация на реализацията

- Fast CGI –сървърът използва постоянен (persistent) процес, където се изпълняват CGI модулите (частично реализиран в Apache – `mod_fastcgi`).
- Сървърни модули – CGI процесорите се включват в тялото на Web сървера.
- Резидентни процеси (ако OS ги поддържа).

Форми

Начин за предаване на данни от агента (клиента) към сървера.

```
<FORM  ACTION=cgi-модул  METOD={GET |POST}>
   Html оператори и form контроли
</FORM>
```

FORM контроли

Всяка контрола има име и получава евентуано стойност. Формата генерира URL от вида

http://*cgi-модул*?*и*$_1$=*с*$_1$&*и*$_2$=*с*$_2$&...>

където u_j и c_j са името и стойността на j-тата контрола.

Текстови контроли

```
<INPUT TYPE="TEXT" NAME=име  [SIZE=размер-на-
  прозореца]  [MAXLENGTH=дължина]  [VALUE=стойност]>
<INPUT TYPE="PASSWD" NAME=име  [SIZE=размер-на-
  прозореца]  [MAXLENGTH=дължина] >
```

Пример:

```
Enter your name: <INPUT NAME=name TYPE="TEXT"
  SIZE=20 MAXLENGTH=40> and password: <INPUT
  NAME=passwd TYPE="PASSWORD" SIZE=8
  MAXLENGTH=8><BR>
```

<INPUT TYPE="HIDDEN" NAME=*име* VALUE=*стойност*>

<TEXTAREA NAME=име ROWS=редове COLS=колони>

[опционален текст]

</TEXTAREA>

ПРИМЕР:

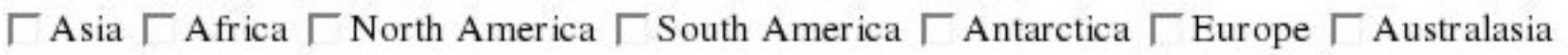
```
Please enter your address:<BR>
<TEXTAREA NAME=address ROWS=5 COLS=50>
</TEXTAREA>
```

Списъци от опции

<INPUT TYPE="CHECKBOX" NAME=*име* VALUE=*стойност* [CHECKED]>

ПРИМЕР:

```
<INPUT TYPE="CHECKBOX" NAME=cb VALUE="1">Asia
<INPUT TYPE="CHECKBOX" NAME=cb VALUE="2">Africa
<INPUT TYPE="CHECKBOX" NAME=cb VALUE="3">North America
<INPUT TYPE="CHECKBOX" NAME=cb VALUE="4">South America
<INPUT TYPE="CHECKBOX" NAME=cb VALUE="5">Antarctica
<INPUT TYPE="CHECKBOX" NAME=cb VALUE="6">Europe
<INPUT TYPE="CHECKBOX" NAME=cb VALUE="7">Australasia
```



```
<INPUT  TYPE="RADIO" NAME=име VALUE=стойност [CHECKED]>
```

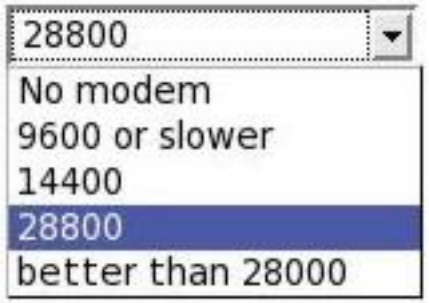
Пример:

```
<INPUT TYPE="RADIO" NAME=rb VALUE="1">Asia<BR>
<INPUT TYPE="RADIO" NAME=rb VALUE="2">Africa<BR>
<INPUT TYPE="RADIO" NAME=rb VALUE="3"CHECKED>North America<BR>
<INPUT TYPE="RADIO" NAME=rb VALUE="4">South America<BR>
<INPUT TYPE="RADIO" NAME=rb VALUE="5">Antarctica<BR>
<INPUT TYPE="RADIO" NAME=rb VALUE="6">Europe<BR>
```

```
<SELECT NAME=име>
  <OPTION VALUE=стойност> текст
  <OPTION  SELECTED VALUE=стойност> текст
     ……
</SELECT>
```

Пример:

```
<SELECT NAME="Speed">
<OPTION value="none">No modem
<OPTION value="vslow">9600 or slower
<OPTION value="slow">14400
<OPTION SELECTED value="OK">28800
<OPTION value="quick">better than 28000
</SELECT>
```



Изпълнителни контоли

```
<INPUT  TYPE="SUBMIT" NAME=име  VALUE=текст>
<INPUT  TYPE="RESET" VALUE=текст>
```

Пример:

```
<INPUT TYPE=RESET VALUE="Clear fields">
<INPUT TYPE=SUBMIT VALUE="Send Information">
```

Други контроли

<INPUT TYPE="IMAGE" SRC="image-url" NAME=име>

Като стойност връща координатите на курсора върху картинката от **image-url**.

CGI модули

- Команден интерпретатор – метод GET

```
#!/bin/sh
# cgi1.sh
# A simple script for showing environment variable
# information passed to a CGI program.
echo Content-type: text/plain
echo

# Next, we want to display the arguments.

echo argv is "$*".
echo

# Then we show the environment variables under which the
# CGI request was made.
```

```
echo SERVER_SOFTWARE=$SERVER_SOFTWARE
echo SERVER_NAME=$SERVER_NAME
echo GATEWAY_INTERFACE=$GATEWAY_INTERFACE
echo SERVER_PROTOCOL=$SERVER_PROTOCOL
echo SERVER_PORT=$SERVER_PORT
echo REQUEST_METHOD=$REQUEST_METHOD
echo PATH_INFO=$PATH_INFO
echo PATH_TRANSLATED=$PATH_TRANSLATED
echo SCRIPT_NAME=$SCRIPT_NAME
echo REMOTE_HOST=$REMOTE_HOST
echo REMOTE_ADDR=$REMOTE_ADDR
echo REMOTE_IDENT=$REMOTE_IDENT
echo QUERY_STRING=$QUERY_STRING
echo CONTENT_TYPE=$CONTENT_TYPE
echo CONTENT_LENGTH=$CONTENT_LENGTH
    exit 0
```

```
argv is .

SERVER_SOFTWARE=Apache/2.2.4 (Fedora)
SERVER_NAME=localhost
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
SERVER_PORT=80
REQUEST_METHOD=GET
PATH_INFO=
PATH_TRANSLATED=
SCRIPT_NAME=/cgi-bin/cgi1.sh
REMOTE_HOST=
REMOTE_ADDR=127.0.0.1
REMOTE_IDENT=
QUERY_STRING=name=Denev&passwd=hhhhh&cb=4&cb=7&rb=5&Speed=
   slow&address=Sofia+%0D%0A5+James+Boucher++Blvd
CONTENT_TYPE=
CONTENT_LENGTH=
```

❑ Командeн интерпретатор - метод POST

```
#!/bin/sh
# cgi2.sh
# A simple script for showing environment variable
# information passed to a CGI program by method POST.
echo Content-type: text/plain

.....................................................

echo CONTENT_LENGTH=$CONTENT_LENGTH
echo The data was
read X
while [ “$X” != “” ]
do
   echo $X
   read X
done
exit 0
```

```
argv is .

SERVER_SOFTWARE=Apache/2.2.4 (Fedora)
SERVER_NAME=localhost
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
SERVER_PORT=80
REQUEST_METHOD=POST
PATH_INFO=
PATH_TRANSLATED=
SCRIPT_NAME=/cgi-bin/cgi2.sh
REMOTE_HOST=
REMOTE_ADDR=127.0.0.1
REMOTE_IDENT=
QUERY_STRING=
CONTENT_TYPE=application/x-www-form-urlencoded
CONTENT_LENGTH=45
name=ffff&passwd=hhhhh&rb=5&Speed=OK&address=
```

`application/x-www-form-urlencoded`

- Допускат се само ASCII символи. Няма интервали, табулации, нови редове.
- Интервалите се заменят с +.
- Не ASCII символите се заменят с %HH, където HH е 16-тичен код на символа. Нпр. край на ред се кодира с %0D%0A.
- Разделител между двойките име=стойност е &.

CGI Модул –C програма

Четене на стойностите на променливите на обкръжението от C programa.

```
char_ptr = getenv("REQUEST_METHOD");
```

Пример:

```
#include <stdlib.h>
#include <stdio.h>
int main(int argc, char *argv[])
{
        printf("Content-type: text/plain\n\n");
        printf("Argument number is %d\n", argc);
```

```
 printf(" SERVER_SOFTWARE=%s\n", getenv("SERVER_SOFTWARE"));
        printf(" SERVER_NAME=%s\n", getenv("SERVER_NAME"));
        printf(" GATEWAY_INTERFACE=%s\n", getenv("GATEWAY_INTERFACE"));
        printf(" SERVER_PROTOCOL=%s\n", getenv("SERVER_PROTOCOL"));
        printf(" SERVER_PORT=%s\n", getenv("SERVER_PORT"));
        printf(" REQUEST_METHOD=%s\n", getenv("REQUEST_METHOD"));
        printf(" PATH_INFO=%s\n", getenv("PATH_INFO"));
        printf(" PATH_TRANSLATED=%s\n", getenv("PATH_TRANSLATED"));
        printf(" SCRIPT_NAME=%s\n", getenv("SCRIPT_NAME"));
        printf(" REMOTE_HOST=%s\n", getenv("REMOTE_HOST"));
        printf(" REMOTE_ADDR=%s\n", getenv("REMOTE_ADDR"));
        printf(" REMOTE_IDENT=%s\n",getenv("REMOTE_IDENT"));
        printf(" QUERY_STRING=%s\n", getenv("QUERY_STRING"));
        printf(" CONTENT_TYPE=%s\n", getenv("CONTENT_TYPE"));
        printf(" CONTENT_LENGTH=%s\n", getenv("CONTENT_LENGTH"));
}
```

```
Argument number is 1
 SERVER_SOFTWARE=Apache/2.2.4 (Fedora)
 SERVER_NAME=localhost
 GATEWAY_INTERFACE=CGI/1.1
 SERVER_PROTOCOL=HTTP/1.1
 SERVER_PORT=80
 REQUEST_METHOD=GET
 PATH_INFO=(null)
 PATH_TRANSLATED=(null)
 SCRIPT_NAME=/cgi-bin/cgi
 REMOTE_HOST=(null)
 REMOTE_ADDR=127.0.0.1
 REMOTE_IDENT=(null)
 QUERY_STRING=name=ffff&passwd=hhhhh&rb=5&Speed=OK&address=
 CONTENT_TYPE=(null)
 CONTENT_LENGTH=(null)
```

❑ CGI модул – decode

Information decoded

Name=name, Value=ffff

Name=passwd, Value=hhhhh

Name=cb, Value=5

Name=cb, Value=7

Name=rb, Value=5

Name=Speed, Value=OK

Name=address, Value=София