

WEB Програмиране - HTML, CSS и Java Script

HTML

HTML, което означава HyperText Markup Language , е език за създаване на web страници. Страниците създадени с HTML могат да съдържат графика, текст, музика, анимация, както и връзки към други страници(хиперлинкове). HTML е лесен за научаване език и вярвам, че в края на този урок Вие ще можете да създадете първата си web страница. От какво се нуждаете за да създадете такава страница? Трябват ви две неща - елементарен текстов редактор и интернет браузър. Ако сте потребители на Windows98 имате и двете. За текстов редактор използвайте вградения в Windows NOTEPAD (не ви препоръчвам да ползвате Word) а за браузър можете на използвате Internet Explorer или Netscape Navigator. Файлът, който ще създадете с Notepad е текстов файл, който обаче трябва да запишете с разширение .html (или .htm). По нататък ще го наричам HTML документ. Сега нека направим една web страница. За целта стартирайте Notepad и напишете следното:

```
<HTML>
<HEAD>
<TITLE> My first page </TITLE>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Сега запазете на твърдия диск написаното от вас във файл с разширение .html . За целта отворете менюто File в Notepad, изберете Save As и в полето име на файл напишете [index.html](#)

Можете да отворите току що направената страница и да я разгледате с някой браузър. За целта кликнете два пъти върху файла в папката, където се намира и той ще бъде отворен от браузъра по подразбиране. В горния случай няма да видите нищо, защото страницата не съдържа никаква информация, която да се визуализира на екрана. Нека ви обясня кое какво означава.

Текста между символите < и > се нарича елемент(tag). Елемента показва на браузъра какво действие да извърши. Можем да разделим елементите на 2 вида - отварящи и затварящи. Отварящите задават на браузъра да започне някакво действие а затварящите да го приключи. Затварящия елемент се различава от отварящия само по символа / преди името на елемента. От примера се вижда, че елементите вървят по двойки, отварящ и затварящ.

HTML документа започва винаги с елемента <HTML> и завършва съответно с </HTML>. <HTML> означава начало на HTML документ, а </HTML> край на HTML документ. Следващата задължителна двойка елементи е <HEAD> и </HEAD>, което буквално означава глава. В главата обикновено се съдържа информация за самия HTML документ. В примера там стои заглавието на документа <TITLE>.

Между елементите `<TITLE>` и `</TITLE>` се съдържа името на HTML документа. Когато срещне тази двойка елементи браузъра визуализира текста между тях на заглавната си лента най-отгоре.

Същинската част на HTML документа се намира между елементите `<BODY>` и `</BODY>`. Виждате, че елементите могат да се влагат един в друг, но внимавайте да спазвате последователността на двойките елементи. Например неправилно е да се напише следното:

неправилно

правилно

`<HTML>`

`<HTML>`

`<HEAD>`

`<HEAD>`

`</HTML>`

`</HEAD>`

`</HEAD>`

`</HTML>`

Затварящия елемент `</HEAD>` трябва да бъде преди затварящия `</HTML>`. Трябва да се получи нещо като огледален образ, първия затварящ трябва да отговаря на последния отварящ.

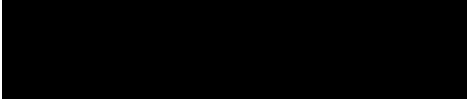


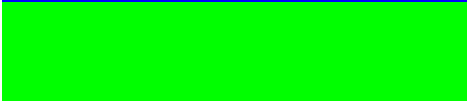
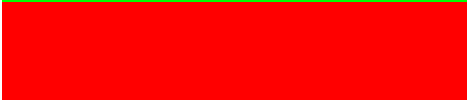



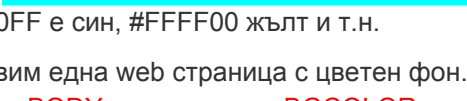
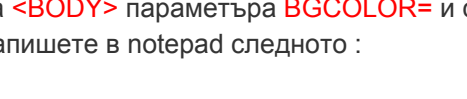




Ето какво научихте до тук.

1. WEB страницата е всъщност текстов файл, но с разширение .html (или .htm), който лесно можете да създадете с помощта на Windows notepad.
2. HTML документа съдържа елементи, които са заградени със символите `<` и `>` и показват на браузъра какво действие трябва да извърши на екрана Ви.
3. Елементите условно се разделят на два вида - отварящи и затварящи, като вървят по двойки, на всеки отварящ следва един затварящ(с малки изключения).
4. Задължителните елементи за всеки HTML документ са `<HTML></HTML>` за начало и край на документа, `<HEAD></HEAD>` за информация за самия документ и `<BODY></BODY>` за съдържанието на документа.

ЦВЕТОВЕ

Цветовите в web страниците се представят със шестразредни шестнадесетични числа, започващи със символа "#". Например `#000000`, `#FFFFFF`, `#C0BB56`. Първите двойка цифри в числото означават количеството на червения цвят, вторите на зеления, третите на синия. Всеки цвят съдържа 256 нюанса, от 00 до FF, като 00 е липса на цвят, а FF пълен цвят. При смесването на основните цветове, браузъра визуализира на вашия монитор получения цвят. Ще ви дам пример.

Когато стойностите за трите цвята са 00, тогава се показва черен цвят. При стойности и за трите цвята FF цветът е бял. Разгледайте следната таблица :

ч	з	с	получен цвят
е	е	и	
р	л	н	
в	е		
е	н		
н			
0	0	0	
0	0	0	
F	F	F	
F	F	F	
0	0	F	
0	0	F	
0	F	0	
0	F	0	
F	0	0	
F	0	0	
F	F	0	
F	F	0	
F	0	F	
F	0	F	
0	F	F	
0	F	F	

Както се вижда цвета #0000FF е син, #FFFF00 жълт и т.н.

Сега вече можем да направим една web страница с цветен фон. За целта трябва да се добави към елемента <BODY> параметъра BGCOLOR= и съответния цвят заграден с кавички. Сега напишете в notepad следното :

```
<HTML>
```

```
<HEAD>
```

```
  <TITLE> blue background </TITLE>
```

```
</HEAD>
```

```
  <BODY BGCOLOR="#0000FF">
```

```
    Тази страница е на син фон
```

```
  </BODY>
```

```
</HTML>
```

Запазете файла като color.html в някоя папка на твърдия си диск и го стартирайте с браузъра (кликнете два пъти върху иконката на файла). Браузъра показва вашата страница на син фон и текста "Тази страница е на син фон". Експериментирайте със стойностите след **BGCOLOR=** като внимавате стойностите за всеки цвят да са в границите между 00 и FF.

Нека направим текста в горния случай да се изписва с жълти букви. За целта добавяме още един параметър след **<BODY>**.

```
<HTML>
<HEAD>
  <TITLE> blue background </TITLE>
</HEAD>
<BODY BGCOLOR="#0000FF" TEXT="#FFFF00">
  Тази страница е на син фон със жълти букви
</BODY>
</HTML>
```

Параметъра **TEXT=** задава цвета на текста на цялата страница. Обърнете внимание, че затварящата скоба ">" на елемента **<BODY>** се поставя след параметрите.

Ето няколко примера със стойностите на различни цветове:

#00336 6	#CCFF CC	#FF66F F	#FFFF6 6
#99FF6 6	#66330 0	#CCCC CC	#0066C C
#FF006 6	#00330 0	#66FFF F	#99996 6
#CC990 0	#66666 6	#FFFFC C	#98E9B 8
#B632F C	#B3111 5	#68E6C C	#E6ED DE

Ето какво научихте за цветовете :

1. Цветовете в HTML се представят със шестразредни шестнадесетични числа, като първата двойка числа означава количеството на червения цвят, втората на зеления, третата на синия цвят. Цветът, който се показва на екрана се образува от смесването на тези три основни цвята.
2. Можете да направите Вашата страница с цветен фон, като добавите параметъра **BGCOLOR=** след елемента **<BODY>**. С параметъра **TEXT=** се задава цвят на текста на страницата.

ТЕКСТ

Езика HTML ви позволява да разполагате текст на страницата, да го подравнявате, оцветявате, да задавате големината на шрифта и самия шрифт. Нека да направим една страница и да разположим някакъв текст :

```
<HTML>
<HEAD>
  <TITLE>Text</TITLE>
</HEAD>
<BODY BGCOLOR="#0000FF" TEXT="#FF0000">
Това е web страница с червен текст на син фон
</BODY>
</HTML>
```

Запишете файла на диска с името text.html и го стартирайте с браузъра. Трябва да видите нещо подобно :

Това е web страница с червен текст на син фон

Текста, който се намира между елементите `<BODY>` и `</BODY>` се визуализира на екрана с червен цвят. Това става с атрибута `TEXT="#FF0000"` на елемента `<BODY>`. Ако промените параметрите на атрибута `TEXT` на `#00FF00` например, текста ще е със зелен цвят (виж [ЦВЕТОВЕ](#)).

Ако искате да удебелите, наклоните или подчертаете текста, трябва да използвате двойките елементи `` и `` за удебеляване(Bold), `<I>` и `</I>` за наклон (Italic) и `<U>` и `</U>` за подчертаване (Underline). Вижте горния пример, само че с удебелен, наклонен и подчертан текст.

```
<HTML>
<HEAD>
  <TITLE>Text</TITLE>
</HEAD>
<BODY BGCOLOR="#0000FF" TEXT="#FF0000">
<B><I><U>Това е web страница с червен текст на син фон</U></I></B>
</BODY>
</HTML>
```

Това е web страница с червен текст на син фон

Можете да използвате тези елементи за да форматирате различни части от текста по различен начин. Например да изпишете "Това е web" с наклонен шрифт, "страница с червен текст с наклонен и удебелен шрифт, "на син фон" с удебелен и подчертан шрифт. Ето как става :

<I>Това е web</I><I> страница с червен текст</I><U> на син фон</U>

Внимавайте с вложените елементи. Трябва първия затварящ да отговаря на последния отварящ от същия вид .Понеже така изглежда малко нечетливо, можете да го напишете така :

<I>Това е web</I>

<I> страница с червен текст</I>

<U> на син фон</U>

.....ефекта ще е същия. Въпреки че пренасяте различни части от текста на нов ред, браузъра не го разбира и слепва думите една зад друга. За да минете на нов ред трябва да използвате елемента
 на края на текста. Този елемент е единичен и на него не отговаря затварящ елемент. Вижте следващия пример :

<HTML>

<HEAD>

<TITLE> Font </TITLE>

</HEAD>

<BODY>

Червен текст.

Зелен текст на нов ред.

Син текст на нов ред.

</BODY>

</HTML>

На екрана трябва да видите ето това :

Червен текст.

Зелен текст на нов ред.

Син текст на нов ред.

Параметъра **TEXT** на елемента <BODY> задава цвета на текста на цялата страница. Ако искате да зададете различен цвят за различни части от текста, трябва да използвате елемента заедно с параметъра **COLOR**. Този елемент показва как да се форматира текста между и . В случая на първия текст му е зададен червен цвят с атрибута **COLOR** , на втория зелен а на третия син. Елемента
 ви прехвърля на нов ред.

Елемента има и други атрибути. Нека разгледаме атрибута **SIZE**. Той задава размера на шрифта, като стойностите му са от 1 до 7. 1 е най-малкия шрифт а 7 най-големия. Вижте горния пример, където за първия ред ще зададем шрифт с големина 4, за втория 1 а за третия 7.

<HTML>

<HEAD>

```
<TITLE> Font </TITLE>
</HEAD>
<BODY>
  <FONT COLOR="#FF0000" SIZE=4>Червен текст.</FONT><BR>
  <FONT COLOR="#00FF00" SIZE=1>Зелен текст на нов ред.</FONT><BR>
  <FONT COLOR="#0000FF" SIZE=7>Син текст на нов ред.</FONT><BR>
</BODY>
</HTML>
```

Червен текст.

Зелен текст на нов ред.

Син текст на нов ред.

Можете на един ред или в една дума да използвате различни по големина и цвят символи.

```
<HTML>
<HEAD>
  <TITLE> Font </TITLE>
</HEAD>
<BODY>
  <FONT COLOR="#FF0000" SIZE=2>Текст</FONT>
  <FONT COLOR="#00FF00" SIZE=3>с различна</FONT>
  <FONT COLOR="#0000FF" SIZE=7>г</FONT>
  <FONT COLOR="#00FFFF" SIZE=6>о</FONT>
  <FONT COLOR="#FF00FF" SIZE=5>л</FONT>
  <FONT COLOR="#AA0066" SIZE=4>е</FONT>
  <FONT COLOR="#0033CC" SIZE=3>м</FONT>
  <FONT COLOR="#2200FF" SIZE=2>и</FONT>
  <FONT COLOR="#FF7700" SIZE=1>н</FONT>
  <FONT COLOR="#555555" SIZE=2>а</FONT>
  <FONT COLOR="#DD1188" SIZE=3> и цвят.</FONT>
</BODY>
</HTML>
```

Текс с различна **Го**Лемина и цвят.

Буквите в този случай се прилепват една зад друга, защото липсва елемента **
** за пренос на нов ред.

Друг атрибут на **** е **FACE**. С **FACE** можете да указвате на браузъра с какъв точно шрифт да се показва текста. Ако на **FACE** не е зададена стойност, браузъра показва шрифт по подразбиране. Ако държите обаче текста да се показва с някакъв определен шрифт, трябва да го зададете:

**** Това е текст със шрифт Arial.****

**** Това е текст със шрифт Times New Roman.****

В първия случай текста ще бъде със шрифт "Arial" , а във втория с "Times New Roman". Не задавайте много екзотични шрифтове, защото има опасност този който разглежда страницата ви нищо да не види. Придържайте се към оригиналните шрифтове, инсталирани с Windows. Моят съвет е, ако можете да избягвате да включвате атрибута **FACE**, така оставяте на браузъра сам да намери подходящия шрифт.

Друг начин да уголемите символите е чрез елементите **<H1>** до **<H6>**. Тези елементи се употребяват за изписване на заглавия. **<H1>** е най-големия използван шрифт, а **<H6>** най-малкия. Вижте следващия пример :

```
<HTML>  
<HEAD>  
  <TITLE>Н елемент</TITLE>  
</HEAD>  
<BODY>  
  <CENTER>  
    <H1>Голямо заглавие</H1>  
    <H3>Средно заглавие</H3>  
    <H6>Малко заглавие</H6>  
  </CENTER>  
</BODY>  
</HTML>
```

Голямо заглавие

Средно заглавие

Малко заглавие

Забележете, че текста между тези два елемента излиза винаги удебелен. Друга особеност на елемента `<H>` е това, че не се налага да ползвате `
` за да минете на нов ред. Браузъра автоматично прескача един ред щом срещне `</H>`.

В горния пример има нов елемент - `<CENTER>`. Всичко което се намира между `<CENTER>` и `</CENTER>` се показва в средата на страницата, независимо дали е текст или графика.

Друг елемент за прескачане на ред е `<P>`. За разлика от `
` той не минава на следващия ред, а прескача един ред. Друго различие е, че елемента `<P>` се поставя преди текста, а не след него както при `
`. Елемента `<P>` може да се употребява като единичен елемент или като двойка елемента `<P></P>`.

`<P>` единичен елемент

`<P>` двойка елемента `</P>`

И в двата случая ефекта е един и същ. За разлика от `
` елемента `<P>` притежава някои атрибути. Когато тези атрибути се употребяват трябва да използвате двойката елемента `<P></P>`. Един от атрибутите на `<P>` е `ALIGN`. Той може да има стойности `left`, `right` или `center` и подравнява текста съответно вляво, вдясно и в средата. Вижте как става това :

```
<HTML>
```

```
<HEAD>
```

```
  <TITLE> P елемент </TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
<P ALIGN="left">текст, подравнен вляво</P>
```

```
<P ALIGN="center">текст, подравнен в средата</P>
```

```
<P ALIGN="right">текст, подравнен вдясно</P>
```

```
</BODY>
```

```
</HTML>
```

Визуализирането на тази страница ще изглежда така :

текст, подравнен вляво

текст, подравнен в средата

текст, подравнен вдясно

Забележете, че текстовете вече са през един ред, поради това, че използвахме елемента `<P>`.

Какво научихте дотук :

1. В езика HTML можете да разполагате текст, да му задавате цвят, големина, шрифт. Можете също така да удебелявате, наклоняте и подчертавате текст.
2. С елемента `<H>` можете да задавате големина на заглавния шрифт, както и да подравнявате текста с параметрите на елемента `<P>`.

СПИСЪЦИ

Езика HTML ви дава възможност да използвате три вида списъци - подредени, неподредени и списъци с обяснения. Подредените списъци са номерирани и изглеждат така :

1. Иван

2. Георги
3. Захари

Неподредените списъци използват вместо номерация, някакъв символ. Например точка:

- Иван
- Георги
- Захари

Списъците с обяснения не използват символ или цифра пред текста. Вместо това поставят обяснителен текст след всеки елемент от списъка:

Иван счетоводител
Георги касиер
Захари шофьор

Подреден списък можете да създадете с елемента `` и съответно затварящия ``. Елементите на списъка се задават с елемента ``, който може да се използва като двойка елементи или като единичен елемент. Нека направим един такъв списък :

```
<HTML>  
<HEAD>  
  <TITLE> ordered list</TITLE>  
</HEAD>  
<BODY>  
да не забравя да купя:  
<OL>  
<LI>захар  
<LI>яйца  
<LI>сода каустик за тъщата  
</OL>  
</BODY>  
</HTML>
```

Вижте как ще изглежда :

да не забравя да купя:

1. захар
2. яйца
3. сода каустик за тъщата

Въпреки, че не сме задали да се показват номера 1. 2. 3. , те се добавят автоматично в списъка. Ако искате списъка да се показва с римски вместо с арабски цифри, трябва да добавите атрибут `TYPE` на елемента ``. Атрибута `TYPE` показва какви символи да се визуализират преди елементите на списъка. Стойностите на атрибута `TYPE` за подредените списъци са :

`TYPE=1` за списъци
1.
2.
3.

`TYPE=a` за списъци
a.
b.

с.

TYPE=A за списъци
A.
B.
C.

TYPE=i за списъци
i.
ii.
iii.

TYPE=I за списъци
I.
II.
III.

Така че ако искате вместо

1. захар
2. яйца
3. сода каустик за тъщата

на екрана да се вижда

- I. захар
- II. яйца
- III. сода каустик за тъщата

трябва да добавите към елемента ** TYPE=I** . С други думи трябва да напишете **<OL TYPE=I>**. Ако пропуснете атрибута **TYPE**, по подразбиране ще се показва списък от типа 1. 2. 3.

Неподреден списък се създава с двойката елементи ****. Този списък за разлика от подредения не е номериран, а използва някакъв символ за всеки елемент от списъка, например точка.

Да направим горния пример да се показва като неподреден списък:

```
<HTML>  
<HEAD>  
  <TITLE> unordered list</TITLE>  
</HEAD>  
<BODY>  
да не забравя да купя:  
<UL>  
<LI>захар  
<LI>яйца  
<LI>сода каустик за тъщата  
</UL>  
</BODY>  
<HTML>
```

Тогава на екрана ще се вижда следното:

да не забравя да купя:

- захар

- яйца
- сода каустик за тъщата

При неподредените списъци също можете да използвате атрибута **TYPE**.

Валидните стойности за него са **disk**, **square** и **circle**. Вижте как ще изглежда същия списък, само че с различни стойности на **TYPE** :

<UL TYPE=disk>

- захар
- яйца
- сода каустик
за тъщата

<UL TYPE=square>

- захар
- яйца
- сода каустик
за тъщата

<UL TYPE=circle>

- захар
- яйца
- сода каустик
за тъщата

Списъците с обяснения се създават с двойката елементи **<DL></DL>**. Елементите на списъка се задават с **<DT>**, а обяснението към него с **<DD>**. Примера в началото на страницата с имената и професиите ще изглежда по следния начин:

```
<HTML>
<HEAD>
  <TITLE>definition list</TITLE>
</HEAD>
<BODY>
<DL>
  <DT>Иван
  <DD>счетоводител
  <DT>Георги
  <DD>касиер
  <DT>Захари
  <DD>данъчен инспектор
</DL>
</BODY>
</HTML>
```


Ако желаете, можете да влагате един в друг списъци от един вид или от различни видове. Ще ви дам пример с вложени един в друг подреден и неподреден списък, като този например :

1. Да купя за къщи:
 - сладолед
 - торта
2. Да купя за офиса:
 - дискети
 - химикалки
 - нова мишка

HTML документа на горния пример изглежда ето така :

```
<HTML>
<HEAD>
  <TITLE>вложени списъци</TITLE>
</HEAD>
<BODY>
<OL>
```

```
<LI>Да купя за въщи:
  <UL TYPE=circle>
    <LI>сладолед
    <LI>торта
  </UL>
<LI>Да купя за офиса:
  <UL TYPE=circle>
    <LI>дискети
    <LI>химикалки
    <LI>нова мишка
  </UL>
</OL>
</BODY>
</HTML>
```

Можете да използвате и собствени картинки при неподредените списъци. За да ви покажа, направих една картинка -червена точка  с размер 9x9 пиксела.

дискети

химикалки

нова мишка

Така списъка стана доста по хубав.

```
<HTML>
<HEAD>
  <TITLE>списък с картинка</TITLE>
</HEAD>
<BODY>
<UL imagesrc="redpoint.gif">
  <LI>дискети</LI>
  <LI>химикалки</LI>
  <LI>нова мишка</LI>
</UL>
</BODY>
</HTML>
```

Изполвайки атрибута **imagesrc** на елемента **** зададох на списъка името на файла, който да се показва пред елементите. Картинката е файл с име redpoint.gif и трябва да се намира в основната папка, където разполагате HTML документите си. Можете да я сложите и в подпапка, но тогава трябва да зададете и пътя до нея, например: **<UL imagesrc="images/redpoint.gif">** ако подпапката се казва images. Ако сте забелязали, в края на тази страница използвам също списък, само че със сини точки и включени хиперлинкове към други страници :-)

Ето какво научихте за списъците:

1. Списъците са три вида, подредени, неподредени и списъци с обяснения.
2. Подредените списъци са номерирани с арабски или римски числа, както и с малки или големи букви от А до Z.

3. Неподредените списъци вместо нумерация използват някакъв символ, обикновено диск, квадрат или окръжност, но можете да ползвате и собствени картинки.
4. Списъците могат да се влагат един в друг.

ХОРИЗОНТАЛНИ ЛИНИИ

Езика HTML позволява да се изчертават хоризонтални линии с елемента `<HR>`. Този елемент няма съответен затварящ елемент, но има някои атрибути. Дължината на линията се задава с атрибута `WIDTH` на елемента `<HR>`. Дължината можете да зададете в пиксели или в проценти от вирината на екрана. Нека начертаем 2 линии, едната е дължина 100 пиксела, а другата 100% от ширината на екрана :

```
<BODY>
<HEAD>
  <TITLE>Хоризонтални линии</TITLE>
</HEAD>
<BODY>
<HR WIDTH=100>
<HR WIDTH=100%>
</BODY>
</HTML>
```



Когато промените размера на прозореца на браузъра втората линия ще се променя с него, докато първата ще си остане 100 пиксела. Обърнете внимание, че линиите се подравняват в средата по подразбиране. Ако искате да промените това положение трябва да използвате атрибута `ALIGN`, и неговите параметри `left`, `center` и `right`, съответно за подравняване вляво, в средата и вдясно на страницата. Параметъра `center` спокойно можете да го пропуснете, защото линията така или иначе се подравнява в средата при липсата на атрибута `ALIGN`. Сега да начертаеме две линии с дължина по 300 пиксела и да ги подравним съответно вляво и вдясно:

```
<BODY>
<HEAD>
  <TITLE>Хоризонтални линии</TITLE>
</HEAD>
<BODY>
<HR WIDTH=300 ALIGN="left">
```

```
<HR WIDTH=300 ALIGN="right">
</BODY>
</HTML>
```

Дебелината на линията се променя с атрибута **SIZE**. Параметрите на **SIZE** се задават в пиксели. Например **SIZE=5** означава, че линията е с дебелина 5 пиксела. Ето какво представлява такава линия с дължина 80%, подравнена в средата:

```
<BODY>
<HEAD>
  <TITLE>Хоризонтални линии</TITLE>
</HEAD>
<BODY>
<HR WIDTH=80% SIZE=5>
</BODY>
</HTML>
```

Ако зададем и атрибута **NOSHADE** линията ще се покаже плътна, без сянка.

```
<BODY>
<HEAD>
  <TITLE>Хоризонтални линии</TITLE>
</HEAD>
<BODY>
<HR WIDTH=80% SIZE=5>
</BODY>
</HTML>
```

Цвят на линията можете да зададете с **COLOR**, но се опасявам, че почитателите на Netscape няма да го видят. Ето една синя линия с дебелина 8 пиксела и ширина 90% :

```
<BODY>
<HEAD>
  <TITLE>Хоризонтални линии</TITLE>
```

```
<HEAD>
<BODY>
<HR WIDTH=90% SIZE=8 COLOR="#0000FF">
</BODY>
</HTML>
```

ХИПЕРЛИНКОВЕ

Езика HTML ви позволява да правите препратки към други страници, към части от същата страница, към файлове, картинки, електронна поща. Тези препратки се наричат ХИПЕРЛИНКОВЕ (или хипервръзки). Без хиперлинковете един web сайт не би имал никакъв смисъл. Хиперлинк се създава с двойката елементи `<A>` и ``. Когато правите препратка към някакъв интернет адрес, трябва да използвате атрибута `HREF` на елемента `<A>`. Ето един пример, който препраща към адрес `www.search.bg` :

```
<HTML>
<HEAD>
  <TITLE>hyperlink</TITLE>
</HEAD>
<BODY>
  <A HREF="http://www.search.bg">SEARCH.BG</A>
</BODY>
</HTML>
```

Нека да анализираме този пример. С `<A HREF=` се задава адреса, към който води препратката. В случая това е адреса на `search.bg`, предхождан от `http://`. Този адрес се нарича още URL. След това изписвате текста, който ще води до съответния адрес, в случая текста е `SEARCH.BG`. На екрана ви този сайт трябва да изглежда така :

[SEARCH.BG](http://www.search.bg)

Когато посочите с мишката този надпис, курсора се сменя от стрелка на сочеща ръка. Ако кликнете върху него отивате директно на адреса на `search.bg`. Забележете, че цвета на текста е син, а самият текст е подчертан без вие да сте задавали такива параметри. Това е така, защото браузъра по подразбиране показва хиперлинк подчертан и със син цвят. Ако кликнете върху него, отидете на `search.bg` и после пак се върнете на тази страница, цвета вече няма да бъде син, а лилав. Причината за това е, че браузъра визуализира посетените от вас линкове с лилав цвят. Можете да промените цвета на линка по ваше желание. Това става с атрибутите на елемента `<BODY>` `LINK`, `ALINK` и `VLINK`. `LINK` задава какъв да бъде цвета на линка, `ALINK` (active link) задава цвета на активния линк (този, на който се

намирате в момента), а VLINK (visited link) задава цвета на посетените вече от вас линкове. Ето един такъв пример:

```
<HTML>
<HEAD>
  <TITLE>hyperlink</TITLE>
</HEAD>
<BODY LINK="#00FF00" ALINK="#00FFFF" VLINK="#FF0000">
```

Посетете нашия

```
<A HREF="http://forum.search.bg"><B>FORUM</B></A>
</BODY>
</HTML>
```

Натиснете [ТУК](#) за да видите как изглежда този пример на нова страница. Линка ще бъде със зелен цвят, активния линк със светлосин, а посещения със червен. Забележете, че в този случай се отвори нов прозорец на браузъра. Това стана, защото използвах атрибута **TARGET** на елемента **<A>**. Атрибута **TARGET** има няколко параметъра. За да се отвори линка в нов прозорец се ползва параметъра **"_blank"**. Ако пропуснете **TARGET** линка по подразбиране се отваря в същия прозорец. Вижте примера с линка към SEARCH.BG, но отварящ се в нов прозорец :

```
<HTML>
<HEAD>
  <TITLE>hyperlink</TITLE>
</HEAD>
<BODY>
  <A HREF="http://www.search.bg" TARGET="_blank">SEARCH.BG</A>
</BODY>
</HTML>
```

[SEARCH.BG](#)

Кликнете върху SEARCH.BG и ще видите сами. Другите параметри на **TARGET** ще ги разгледаме в урока "РАМКИ".

Освен препратки към други сайтове, можете да правите препратка и към собствения си сайт. За целта трябва да слагате всички web документи, създадени от вас в една папка или в нейните подпапки. Нека да направим един сайт (не страница, а именно сайт, съвкупност от страници с препратки една към друга). Ще направим сайта от 3 страници, като ще ги именуваме "начална страница", "лични данни" и "свържете се с мен". Направете си нова папка на твърдия диск и и дайте името "website". Сега нека направим първата страница:

```
<HTML>
<HEAD>
```

```
<TITLE>начална страница</TITLE>
</HEAD>
<BODY>
<CENTER>
<H1>ЛИЧНАТА МИ СТРАНИЦА</H1>
</CENTER>
<HR WIDTH=90% SIZE=6>
<BR>
<A HREF="forme.html">Нещо повече за мен.</A><BR>
<A HREF="contact.html">Пишете ми</A>
</BODY>
</HTML>
```

Запазете този документ във същата папка под името index.html. Сега да направим другите две страници от сайта:

```
<HTML>
<HEAD>
  <TITLE>лични данни</TITLE>
</HEAD>
<BODY>
<CENTER>
<H1>ЗА МЕН</H1>
</CENTER>
<HR WIDTH=90% SIZE=6>
<BR>
Тук напишете нещо за себе си
<BR>
<A HREF="index.html">Към началната страница</A><BR>
<A HREF="contact.html">Пишете ми</A>
</BODY>
</HTML>
```

Този документ го запазете в същата папка по името forme.html и направете последната страница с името contact.html.

```
<HTML>
<HEAD>
  <TITLE>свържете се с мен</TITLE>
```

```
</HEAD>
<BODY>
<CENTER>
<H1>ПИШЕТЕ МИ</H1>
</CENTER>
<HR WIDTH=90% SIZE=6>
<BR>
Тук напишете адреса си, e-mail и т.н.
<BR>
<A HREF="index.html">Към началната страница</A><BR>
<A HREF="forme.html">За мен</A><BR>
<A HREF="mailto:www4u@search.bg">Моя e-mail</A>
</BODY>
</HTML>
```

Ето как ще изглежда този сайт. Натиснете [ТУК](#), или го стартирайте от папка website на твърдия си диск. Виждате, че вече отделните страници са свързани а препратки помежду им. Когато задавате линкове към файлове, които са във същата папка, изписвате само името им и разширението (forme.html). Ако HTML документа е в някаква подпапка на вашата папка, например подпапка main, трябва да зададете и пътя до там [За мен](#).

Новото в горния пример е [Моя e-mail](#). Когато срещне mailto:име_на_мейла и посетителят на страницата кликне върху текста след него браузъра стартира програмата му за електронна поща, която вписва в полето за изпращане адреса на мейла, който е зададен след mailto:

По този начин спестявате на посетителят на страницата, който иска да ви пише, ръчното стартиране и настройка на програмата си за електронна поща. Ако посетителят няма такава програма (Eudora или Outlook Express например) или използва webmail в hotmail или yahoo например, параметърът mailto: няма да върши работа. Затова винаги пишете вашия e-mail между [<A>](#) и [](#). Ето така:

```
<A HREF="mailto:www4u@search.bg">Моя e-mail      www4u@search.bg</A>
```

Освен към други страници и e-mail можете да правите препратки в рамките на същата страница. Това става по следния начин. Изполвайки елемента [<A>](#) давате някакво име на определена част от HTML документа, нещо като маркировка. Например, в началото на страницата която разглеждате в момента съм дал име "begin":

```
<A NAME="begin">Езика HTML</A>
```

Ако си спомняте така започваше тази страница. На първите две думи съм им дал име за препратка "begin". Сега ако напиша [](#)върнете се в

началото на страницата

 и кликнете върху текста, ще се върнете в началото на страницата. Ето, убедете се сами:

[върнете се в началото на страницата](#)

Можете да слагате навсякъде в документа такива маркировки и да правите препратки към тях. Също така, можете да препращате към друг HTML документ, в който има маркировка. Това става като първо напишете името на документа, после # и после името на маркировката.

``Отивате на index.html с препратка към част begin

или

``Отивате на index.html с препратка към част begin

Какво научихте за хиперлинковете :

1. Хиперлинковете (или хипервръзките) са препратки към други страници от същия сайт или към други сайтове, към електронна поща или към части от същата страница или от сруги страници.
2. Хиперлинк се реализира с елемента `<A>` и атрибута `HREF`. Име на част от страницата се дава с атрибута `NAME`.

КАРТИНКИ

Една от най-силните характеристики на езика HTML е възможността да разполагате картинки върху страницата си. Това става с елемента `` и атрибута му `SRC`, като след `SRC` записвате името на файла:

`<HTML>`

`<HEAD>`

`<TITLE> DOLPHIN </TITLE>`

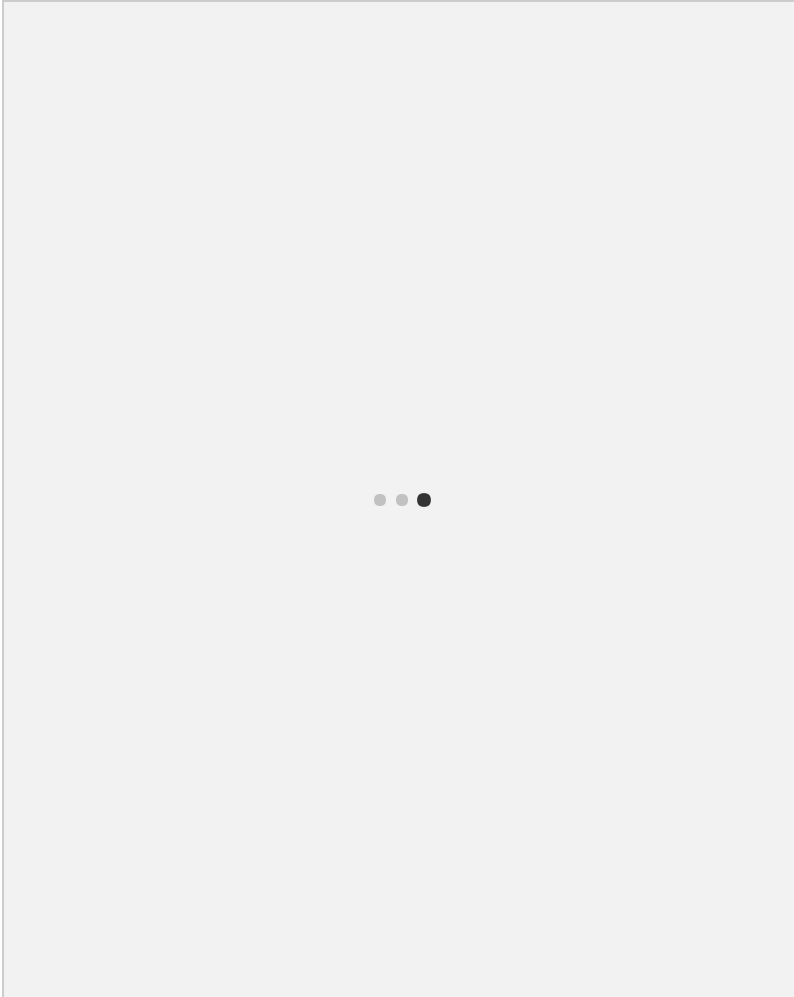
`</HEAD>`

`<BODY>`

``

`</BODY>`

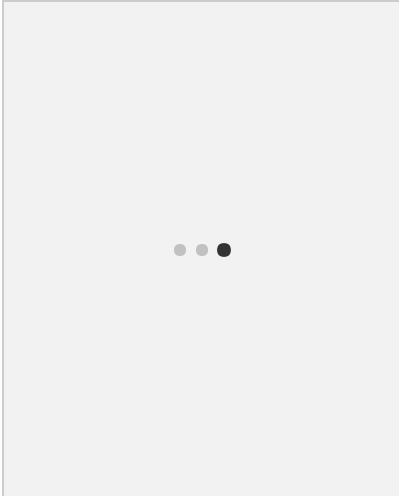
`</HTML>`



Така на екрана си вече имате картинка на делфин. В езика HTML най-често се използват графичните формати GIF(.gif) и JPEG(.jpg). Предимството на първия е че може да съдържа анимация и прозрачен фон, но се ограничава само до 256 цвята. При втория формат няма ограничение за цветовете и картинките са с по-високо качество, но пък нямате прозрачен фон, нито анимация. Картинката на делфина е от типа JPEG.

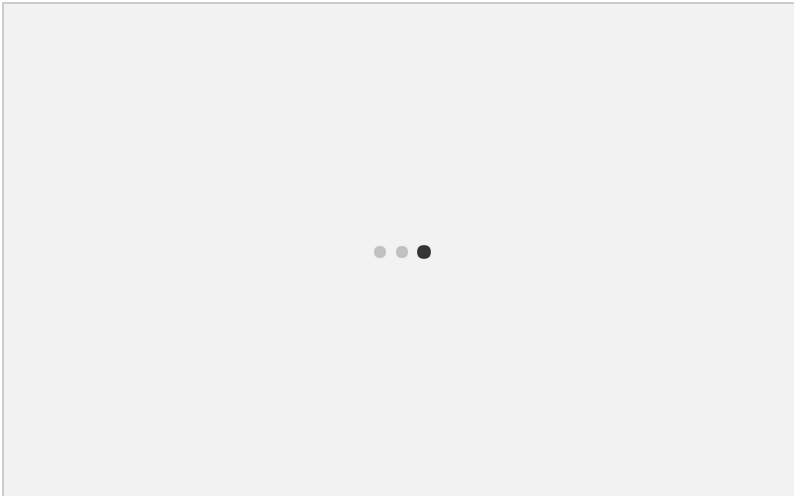
Когато слагате картинка на web страницата си трябва да знаете от колко точки е съставена хоризонтално и вертикално. Например картинката на делфина е от 300 точки хоризонтално и 375 вертикално(кликнете с десния бутон на мишката върху картинката и от менюто изберете Properties за да видите от колко точки е картинката). Вие разбира се можете да свиете или разширите изображението. Това става с атрибутите **WIDTH** и **HEIGHT** на елемента ****. Например нека направим същата картинка на делфин да се показва като 150X187 точки(пиксела) :

```
<IMG SRC="dolphin.jpg" WIDTH="150" HEIGHT="187">
```



В този случай картинката се намалява точно наполовина. Ако не бяхме спазили пропорцията да намалим точно на 2 ширината и височината на картинката изображението щеше да се "смачка", например ето така:

```
<IMG SRC="dolphin.jpg WIDTH="300" HEIGHT="187">
```



Тук използвахме оригиналната ширина от 300 точки, но намалена наполовина височина - 187 точки.

Можете да използвате също така картинки за фон на web страници. Това става с атрибута **BACKGROUND** на елемента **<BODY>**. Нека направим страница с фон същата картинка на делфин:

```
<HTML>
```

```
<HEAD>
```

```
    <TITLE>background</TITLE>
```

```
</HEAD>
```

```
<BODY BACKGROUND="dolphin.jpg">
</BODY>
</HTML>
```

Натиснете [ТУК](#) за да видите как изглежда една такава web страница.

Можете да направите фона на страницата неподвижен(при скролиране на страницата нагоре-надолу фона няма да се мести заедно с нея). Това става с атрибута **BGPROPERTIES="fixed"** разположен след атрибута **BACKGROUND**. Ето така : `<BODY BACKGROUND="dolphin.jpg" BGPROPERTIES="fixed">`. За съжаление, **BGPROPERTIES** се разпознава само от Internet Explorer. При разглеждане с Netscape Navigator фона няма да е неподвижен.

Картинките често служат и като хиперлинкове към други страници (виж [ХИПЕРЛИНКОВЕ](#)). За да добавите хиперлинк към картинка, трябва да използвате двойката елементи `<A>` и ``. Вижте един пример :

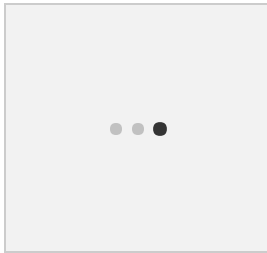
```
<HTML>
<HEAD>
  <TITLE>image</TITLE>
</HEAD>
<BODY>
  <A HREF="http://www.search.bg"><IMG SRC="banners/fan_small.gif"
  BORDER="0" ALT="search.bg"></A>
  <A HREF="http://www.dir.bg"><IMG SRC="banners/dirbg.gif" BORDER="0"
  ALT="dir.bg"></A>
</BODY>
</HTML>
```



При разглеждане на горния html документ с някой браузър, на екрана ще се появят две картинки. При кликане върху първата (кучето) ще отивате на адрес www.search.bg, а втората ще води до www.dir.bg.

Първото изображение е файл с име fan_small.gif, а второто dirbg.gif. И двете се намират в поддиректория banners, затова след атрибута **SRC** се

указва цели път до файла, в случая banners/dirbg.gif. Когато изображението е комбинирано с хиперлинк, то се показва оградено със синя рамка. За да махнете досадната синя рамка, използвайте атрибута **BORDER="0"**. **BORDER** задава дебелината на рамката около изображението в пиксели. При промяна на стойността на BORDER се променя и дебелината на рамката. Вижте картинката с кучето, само че с рамка с дебелина 10 пиксела :



Атрибута ALT задава алтернативния текст, който да се показва, когато вие сочите с мишката върху изображението, без да кликвате върху него. Обикновено това е пояснителен текст, най-често докъде води хиперлинка. Застанете с мишката върху картинката и ще видите за какво става въпрос.

МЕЛОДИИ

Езикът HTML ви позволява да слагате музикален фон на страницата си. За целта трябва да използвате някои широко използвани музикални формати, като .mid и .wav. Тук за първи път ще се сблъскате с разликата в поддръжката на HTML от страна на Netscape Navigator и Internet Explorer.

За да демонстрирам музикалните възможности избрах един midi файл, който се намира на адрес www4u.search.bg/music.mid

Нека започнем с поддръжката от страна на Internet Explorer. За да включите този файл като музикален фон в Internet Explorer трябва да използвате елемента **<BGSOUND>** в главата **<HEAD>**. Като атрибут на **<BGSOUND>** за указване на пътя до музикалния файл се използва **SRC**.

```
<HTML>
<HEAD>
<TITLE>music</TITLE>
<BGSOUND SRC="http://www4u.search.bg/music.mid" LOOP="-1">
</HEAD>
<BODY>
</BODY>
</HTML>
```

Разбира се, ако файла се намира във вашата директория на сървъра не е нужно да указвате целия път, а само да напишете **<BGSOUND SRC="music.mid" LOOP="-1">**. Атрибута **LOOP** задава броя на повторенията на мелодията. Когато

LOOP="-1" тогава мелодията започва отново, веднага след като свърши. Ако искате да се повтори два пъти например и после да спре, трябва да зададете **LOOP="2"**. Netscape Navigator не разпознава елемента **<BGSOUND>**.

Възпроизвеждането на звук от Netscape Navigator става с елемента **<EMBED>**, разположен в тялото **<BODY>**. Всъщност **<EMBED>** показва едно меню за управление на звука. То изглежда ето така :

<EMBED> е елемент за визуализация на мултимедия. С негова помощ можете освен музика да сложите и видео на страницата си, например някой .avi файл. Разбира се трябва да се съобразите с големината на файла, защото никой няма да чака 5-6 минути например да се отвори вашата страница, ако сте прикачили голям .avi файл. Ето как се ползва елемента **<EMBED>** за закачане на midi файла като музикален фон :

```
<HTML>  
<HEAD>  
<TITLE>embed</TITLE>  
</HEAD>  
<BODY>  
<EMBED SRC="music.mid" WIDTH="128" HEIGHT="128" LOOP="true">  
</BODY>  
</HTML>
```

Както виждате, пътя до файла и тук се указва с атрибута **SRC** . Атрибутите **WIDTH** и **HEIGHT** задават ширината и височината на панела за контрол на звука в пиксели. Ако искате панела да се вижда е добре задължително да задавате стойности на атрибутите **WIDTH** и **HEIGHT** . Ако пропуснете да ги зададете браузъра може да покаже нещо, което няма да ви хареса. Атрибута **LOOP** може да има две стойности - **true** и **false** . Когато стойността е **true** , мелодията започва отново веднага след като свърши, докато при **false** се просвирва само веднаж и спира. Ако зададете числова стойност на **LOOP** , мелодията се просвирва толкова пъти, колкото е числото, например **LOOP="3"** ще изсвири мелодията три пъти и ще спре.

Менюто за управление на звука може да не се визуализира. Това става с атрибута **HIDDEN** . Той има 2 стойности - **true** и **false** . При **true** панела остава скрит а при **false** се визуализира. Когато създавате музикален фон, разбира се трябва да използвате стойност **true** .

Елемента **<EMBED>** се разпознава и от Internet Explorer и от Netscape Navigator, но двата браузъра работят с различни атрибути. Например Internet Explorer вместо **LOOP** използва атрибута **PLAYCOUNT** за повторение на изпълнението. Netscape Navigator от своя страна позволява да контролирате силата на звука с атрибута **VOLUME** . Internet Explorer не разпознава атрибута **VOLUME** . Така че стигнахме до извода, че за да добавите музикален фон към своята страница трябва да използвате и двата метода, и **<EMBED>**, и **<BGSOUND>**. Ето как би изглеждала една готова страница с музикален фон :

```

<HTML>
<HEAD>
<TITLE>music</TITLE>
<BGSOUND SRC="http://www4u.search.bg/music.mid" LOOP="-1">
</HEAD>
<BODY>
<EMBED SRC="http://www4u.search.bg/music.mid" HIDDEN="true" LOOP="true">
</BODY>
</HTML>

```

ТАБЛИЦИ

Таблицата служи за по-нагледно представяне на Вашата информация. Например по-добре изглежда следния ред :

www.dir.bg	www.search.bg	www.gyuvetch.bg
--	--	--

..отколкото :

www.dir.bg www.search.bg www.gyuvetch.bg

Таблицата се състои от редове и колони. В горния пример имаме таблица с 3 колони и 1 ред. Всяка клетка от таблицата може да съдържа някакви данни, в примера URL-ите на български портали. За да направите таблица трябва да използвате елемента **<TABLE>** и съответно **</TABLE>** за да я завършите. За изграждане на ред в таблицата се използва **<TR>** и **</TR>** за края на реда (съкращение от **TABLE ROW** - ред в таблица) . Колоните във всеки ред се изписват с **<TD>** и **</TD>** (**TABLE DATA** - данни в таблицата), като между тях се вписва съдържанието на клетката. Всъщност, когато използвате **<TD>** вие създавате една клетка в реда, които сте започнали с **<TR>**. Колкото елемента **<TD>** има между два елемента **<TR></TR>**, толкова колони ще има в таблицата. Когато приключите реда с **</TR>** автоматично минавате на следващия. Сега ще направя една таблица с 2 колони и 3 реда, като клетките в нея ще оставя празни :

```

<table align="center" border="1" width="90%">
<tr>
    <td></td>
    <td></td>
</tr>
<tr>
    <td></td>
    <td></td>
</tr>

```

```
<tr>
    <td ></td>
    <td></td>
</tr>
</table>
```


Новото тук е допълнението `border="1"` към елемента `<TABLE>`, както и `width="90%"`. Числото в кавичките след `border` указва дебелината на рамката в пиксели, а това след `width` широчината на таблицата в проценти от екрана на брауъра. Подравняването на таблицата става с `align`, като са възможни 3 варианта, `"left"` (ляво), `"center"` (в средата) и `"right"` (вдясно). Особеното при задаването на широчината на таблицата е това, че тя може да се зададе в процент от ширината на екрана, но може да се зададе и в брой пиксели. В първия случай след числото се поставя знака %, а във втория не се поставя. Така че `width="50%"` задава широчина на таблицата половината от екрана, а `width="50"` ширина само 50 пиксела. Ето и два примера. Първия е таблица със широчина 80 % и рамка с дебелина 2 пиксела, подравнена вдясно, а втория таблица със ширина 200 пиксела с рамка 5 пиксела, подравнена вляво:

```
<table align="right" border="2" width="80%">
```

```
<table align="left"
border="5"
width="200">
```

Във втория пример се вижда, че когато съдържанието на клетката е по голямо от размера и, тя автоматично се разширява във височина за да го побере. Вижте как изглежда по-дълъг текст в таблица с една колона и 1 ред със синя рамка с дебелина 3 пиксела и ширина 50% от екрана, подравнена в средата:

```
<table align="center" border="3" width="50%" bordercolor="#0000FF">
```

```
<tr>
```

```
<td>
```

```
<p align="center">Текста в тази клетка е по-дълъг
```

от широчината на клетката и за това

таблицата се разширява автоматично във

височина за да побере целия текст.

```
</td>
</tr>
</table>
```

Текста в тази клетка е по-дълъг от широчината на клетката и за това таблицата се разширява автоматично във височина за да побере целия текст.

Цвета на рамката се задава след **bordercolor** и представлява познатото ви вече шестнадесетично число, с което се определя съотношението на трите цвята, червено, зелено и синьо.

В горния пример вече имаме съдържание на клетка. То се поставя между елементите **<TD>** и **</TD>** и може да представлява текст, графика, хиперлинк или каквото ви хрумне!

Сега ще ви покажа как изглежда таблицата в началото на тази страница с препратките до дир.бг, сърч.бг и гювеч.бг, само че без рамка :

```
<table align="center" border="0" width="90%">
<tr>
<td width="33%">
<p align="center"><a href="http://www.dir.bg">www.dir.bg</a> </td>
<td width="33%">
<p align="center"><a href="http://www.search.bg">www.search.bg</a> </td>
<td width="34%">
<p align="center"><a href="http://www.gyuvetch.bg">www.gyuvetch.bg</a> </td>
</tr>
</table>
```

www.dir.bg

www.search.bg

www.gyuvetch.bg

За първи път тук правиме таблица без рамка (**border="0"**). Другото ново нещо е, че задаваме ширината на всяка колона поотделно с **<td width="33%">**. Това се прави за да се разпредели поравно ширината на трите колони в таблицата. Опитайте се да направите горната таблица, като махнете **width="33%"** след **<TD .**, ефекта ще е по-различен.

Можете да променяте цвета на фона на таблицата или да отделните клетки, както и цвета на шрифта ето така:

```
<table align="center" border="1" width="50%">
```

```

<tr>
<td width="50%" align="center" bgcolor="#00FF00"><font color="#0000FF">зелен
фон</font></td>
<td width="50%" align="center" bgcolor="#FF0000"><font color="#FFFF00">червен
фон</font></td>
</tr>
</table>

```



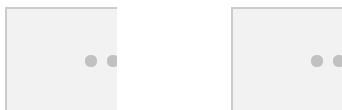
`align="center"` след `<TD>` указва, че съдържанието на клетката трябва да се подравни в средата. `bgcolor` дава цвета на фона на клетката. `<font color` задава цвета на шрифта. Ако `bgcolor` е след `<TABLE>` тогава указва цвета на фона на цялата таблица.

Сега да сложим в таблицата някаква картинка.

```

<table align="center" border="0" width="200">
<tr>
<td valign="middle" align="center"></td>
<td valign="middle" align="center"></td>
</tr>
</table>

```



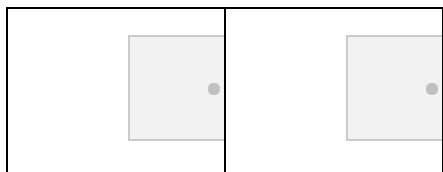
За тази цел направих 2 картинки във вид на бутончета с размер 80X40 пиксела. Тук виждате, че с командата `valign` може да се задава и вертикално подравняване в клетката, в случая `middle` (в средата).

Самата картинка се поставя с ``. Хубаво е сложите `border="0"` след `<img`, иначе картинката ви ще бъде заградена с рамка, което няма да ви хареса много. Следва `src="името на файла"` и след това размера му, `width="80"` ширина и `height="40"` височина. Ако искате клетката да съдържа линк към друга страница, преди `<img..` трябва да добавите познатото ви вече `` и да завършите с `` накрая.

Има още 2 параметъра, които указват как да се подравни клетката в таблицата. Първият е `cellpadding` а втория `cellspacing`. Първият задава разстоянието от

съдържанието на клетката до ръба на клетката, а втория разстоянието между клетките. Пример:

```
<table align="center" border="1" cellpadding="10" cellspacing="10" width="250">
<tr>
<td valign="middle" align="center"></td>
<td valign="middle" align="center"></td>
</tr>
</table>
```

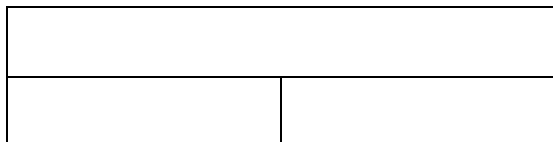


`cellpadding="10"` задава разстояние от ръба на клетката до картинката 10 пиксела, а `cellspacing="10"` задава разстоянието между клетките да бъде 10 пиксела.

Експериментирайте с други параметри. **ВНИМАНИЕ!** Ако не зададете параметрите `cellpadding` и `cellspacing`, техните стойности по подразбиране са "1", а не "0", така че ако не искате да имате разстояние между клетките и ограничаване на разстоянието между съдържанието на клетката и ръба на клетката трябва винаги да пишете `cellpadding="0" cellspacing="0"`.

Можете да обединявате редове и колони в таблицата с `colspan=..` и `rowspan=..` например :

```
<table align="center" border="1" cellspacing="0" cellpadding="0" width="250">
<tr>
<td colspan="2"></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</table>
```



```
<table align="center" border="1" cellpadding="0" cellspacing="0" width="250">
<tr>
```

```
<td rowspan="2"></td>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</table>
```


Хитро, нали ? **colspan="2"** задава да се обединят съседните 2 клетки хоризонтално, а **rowspan="2"** вертикално. Числото в кавички показва броя на клетките, които искате да се обединят. Ето един по-сложен пример :

```
<table align="center" border="1" cellpadding="0" cellspacing="0" width="350">
<tr>
<td rowspan="3"></td>
<td colspan="3"></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</table>
```


Освен **width** за широчина на таблицата можете да използвате и параметъра **height** за да задетете и нейната височина. Стойностите също може да са в проценти от екрана на брауъра или в брой пиксели. Също така може освен цвета на рамката да зададете и цвета на светлата и тъмната част от рамката. Това става с **bordercolorlight** и **bordercolordark**. Вижте таблица с ширина 200 пиксела, височина 20% от екрана, дебелина на рамката 7 пиксела и два цвята на рамката, съответно тъмносин и светлосин, както и червен фон за клетката:

```
<table align="center" border="7" cellpadding="0" cellspacing="0" width="200"
height="20%" bordercolorlight="#00FFFF" bordercolordark="#000080"
bgcolor="#FF0000">
<tr>
<td ></td>
</tr>
</table>
```



Забележка! Netscape Navigator не разпознава **bordercolorlight** и **bordercolordark** и показва цветовете по подразбиране.

Замалко да забравя, вместо **<TD>** можете да използвате **<TH>** (**Table Heading** -заглавие в таблицата). В такъв случай текста в клетката автоматично се центрира в средата и се показва удебелен.

Ето какво научихте за таблиците :

1. Таблица се създава с двойката елементи **<TABLE>** и **</TABLE>**.
2. Редовете в таблицата се създават с **<TR>** и **</TR>** (table row). Клетките във всеки ред (колониите) се създават с **<TD></TD>**
3. Таблицата може да се подравнява спрямо екрана вляво, вдясно и в средата, както и да се задава нейната ширина в пиксели или в проценти от екрана.
4. Таблицата може да се изчертава със или без рамка. Може да се определи цвета на рамката, както и на светлата и тъмната и част.
5. В клетките на таблицата могат да се съдържат картинки, текст, хиперлинкове, цвят за фон.
6. Клетките в таблицата могат да се обединяват.

ФОРМУЛЯРИ

Освен за показване на информация WEB страниците могат да служат и за събиране на информация от потребителя. Това става с така наречените формуляри. За съжаление езика HTML не ви позволява да управлявате информацията във формулярите, а само да ги разполагате на страницата си. Двойката елементи за разполагане на формуляри е **<FORM></FORM>**. Между тази двойка елементи могат да се разполагат неограничен брой формуляри. Всеки един

формуляр се разполага с единичния елемент `<INPUT>`. Този елемент трябва задължително да съдържа двата атрибута `NAME` и `TYPE`. Атрибута `NAME` задава името на формуляра а `TYPE` видът му. Нека разгледаме различните стойност на `TYPE`.

```
<HTML>
<HEAD>
  <TITLE>forms</TITLE>
</HEAD>
<BODY>
<FORM>
<INPUT NAME="form1" TYPE="text">
</FORM>
</BODY>
</HTML>
```

Този HTML документ ще покаже на екрана следното текстово поле:

Такъв тип едноредово текстово поле се задава със стойността на атрибута `TYPE="text"`. Вие можете да промените дължината на полето с атрибута `SIZE`. Ако пропуснете `SIZE` се показва поле с дължина 20 символа. По подразбиране полето се показва празно, но вие можете да зададете някаква начална стойност на полето. Това става с атрибута `VALUE`. Нека направим едно текстово поле с дължина 50 символа и надпис в него "това поле съдържа текст и има дължина от 50 символа".

```
<HTML>
<HEAD>
  <TITLE>forms</TITLE>
</HEAD>
<BODY>
<FORM>
<INPUT NAME="form2" TYPE="text" SIZE="40" VALUE="това поле съдържа текст и
има дължина от 40 символа">
</FORM>
</BODY>
</HTML>
```

Понеже текста който се показва е по-дълъг от 40 символа, последните няколко символа се скриват. Ако кликнете върху полето можете да местите

видимата част на текста наляво и надясно с помощта на стрелките за движение от клавиатурата. Също така можете да напишете свой текст в полето. Ако текста който пишете е по дълъг от дължината на полето, той автоматично започва да се скролира наляво и разкрива новонаписаните символи. Опитайте! Можете да ограничите дължината на изписвания текст с атрибута **MAXLENGTH**. Например с **MAXLENGTH="40"** ограничавате надписа до 40 символа.

Друг параметър на атрибута **TYPE** е **password**. Той действа по същия начин като **text**, с изключение на това, че в полето не се показва самия текст, който въвеждате, а звездички.

```
<HTML>
<HEAD>
  <TITLE>forms</TITLE>
</HEAD>
<BODY>
<FORM>
<INPUT NAME="form3" TYPE="password">
</FORM>
</BODY>
</HTML>
```

Напишете нещо в това поле и ще видите как всеки символ се заменя със звездичка. Това е особено полезно, когато текста е някаква парола и не трябва да се вижда от хората около вас.

Друг тип формуляри са кутиите с отметки. Те се задават със стойността **checkbox** на атрибута **TYPE**.

```
<HTML>
<HEAD>
  <TITLE>forms</TITLE>
</HEAD>
<BODY>
<FORM>
<INPUT NAME="form4" TYPE="checkbox" ><BR>
<INPUT NAME="form5" TYPE="checkbox"><BR>
<INPUT NAME="form6" TYPE="checkbox" CHECKED>
</FORM>
```

```
</BODY>
```

```
</HTML>
```

Кутиите с отметки служат за избор на една или няколко възможности едновременно. Ако кликнете в кутията, отметката се появява. Ако кликнете повторно отметката изчезва. Можете да правите отметки в повече от една кутия. По подразбиране кутиите с отметки се показват празни. Ако искате да се показва отметка в дадена кутия, използвайте атрибута **checked**.

Подобни на кутиите с отметки са радиобутоните. Разликата е, че при радиобутоните може да бъде отметнат само един бутон. Когато изберете друг бутон, отметката в предишния изчезва. Особеното при радиобутоните е, че всички трябва да носят едно и също име, в случая **NAME="R1"**. Радиобутон се създава с атрибута **TYPE="radio"**.

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>radiobutton</TITLE>
```

```
<BODY>
```

```
<FORM>
```

```
<INPUT TYPE="radio" VALUE="V1" checked NAME="R1"><BR>
```

```
<INPUT TYPE="radio" NAME="R1" VALUE="V2"><BR>
```

```
<INPUT TYPE="radio" NAME="R1" VALUE="V3">
```

```
</FORM>
```

```
</BODY>
```

```
</HTML>
```

Друг тип формуляри са бутоните SUBMIT и RESET. Когато се използват в комбинация с друг формуляр, тези два бутона съответно потвърждават или изчистват въведената информация. Вижте :

Бутона RESET изчиства полето, в случай че сте въвели нещо в него, а бутона SUBMIT потвърждава въведената информация. В случая при натискане на бутона SUBMIT няма да има ефект. Опитайте! Въведете нещо

в полето и натиснете бутоните! Ето как изглежда HTML кода на горния пример :

```
<HTML>
<HEAD>
<TITLE>submit and reset</TITLE>
</HEAD>
<BODY>
<FORM >
<INPUT TYPE="text" NAME="T1" SIZE="20">
<INPUT TYPE="submit" VALUE="Submit" NAME="B1">
<INPUT TYPE="reset" VALUE="Reset" NAME="B2">
</FORM>
</BODY>
</HTML>
```

SUBMIT бутон се създава с атрибута **TYPE="submit"**. Атрибута **VALUE="Submit"** задава какъв да бъде надписа на бутона. Можете да го промените по свое желание. За бутона RESET се ползва атрибута **TYPE="reset"**. Останалите атрибути на RESET действат аналогично с тези на SUBMIT.

Можете да създадете формуляр във вид на многоредово поле с елемента **<TEXTAREA>**. Размера на полето се задава с атрибутите **COLS** и **ROWS**. **COLS** показва колко символа да е широко полето, в случая 30, а **ROWS** от колко реда да се състои, в случая 3.

textarea

```
<HTML>
<HEAD>
<TITLE>textarea</TITLE>
</HEAD>
<BODY>
<FORM >
<TEXTAREA ROWS="3" NAME="S1" COLS="30">textarea</TEXTAREA>
</FORM>
</BODY>
</HTML>
```

Когато използвате елемента `<TEXTAREA>`, не се използва елемент `<INPUT>`. Текста, който се намира между `<TEXTAREA>` и `</TEXTAREA>`, се показва по подразбиране в полето. Ако пропуснете текста полето ще се покаже празно.

Последния тип формуляри, който ще ви покаже са падащите менюта. Такъв вид формуляри се създават с елемента `<SELECT>`. Вижте :

избор 1 избор 2 избор 3

Кликнете с мишката върху стрелката надолу и ще се покаже меню с 3 избора, избор 1, избор 2 и избор 3. Стойностите, които да се показват в менюто се задават с елемента `<OPTION>`. Вижте HTML кода на горния пример :

```
<HTML
<HEAD>
<TITLE>drop-down menu</TITLE>
</HEAD>
<BODY>
<FORM >
<SELECT SIZE="1"NAME="D1">
    <OPTION>избор 1</OPTION>
    <OPTION>избор 2</OPTION>
    <OPTION>избор 3</OPTION>
</SELECT>
</FORM>
</BODY>
</HTML>
```

Атрибута SIZE на елемента `<SELECT>` задава броя на редовете, които да се показват в менюто. Когато SIZE="1", тогава се наблюдава т.нар. падащо меню. Ако зададете стойност по-голяма от 1, тогава вече ще се показва вертикална превъртаща лента. Вижте горното меню, само че със стойност SIZE="5" :

Избор 1 Избор 2 Избор 3 Избор 4 Избор 5 Избор 6 Избор 7 Избор 8

РАМКИ

Сигурно често сте виждали страници, които са разделени на отделни части. Такива части в езика HTML се наричат рамки(frames) или фреймове, ако предпочитате. Разделянето на рамки позволява да се показва различен html

документ във всяка рамка, както и хиперлинкове от една рамка да се показват в друга. За да направите страница с рамки, трябва да създадете отделен html документ, чиято функция е да раздели страницата, да укаже в коя рамка кой друг html документ да се показва, да зададе хиперлинковете от една рамка да се показват в друга. Този html документ е по-различен от другите. Например тялото **<BODY>** при него е заменено с **<FRAMESET>**. Вижте един такъв html документ, който разделя страницата на 2 еднакви хоризонтални рамки. В първата рамка се показва страницата 1.html а във втората 2.html.

```
<HTML>
<HEAD>
  <TITLE>рамки</TITLE>
</HEAD>
<FRAMESET ROWS="50%,50%">
  <FRAME NAME="first" SRC="1.html">
  <FRAME NAME="second" SRC=2.html">
</FRAMESET>
</HTML>
```

Разделянето се задава от елемента **<FRAMESET>**. Той има 2 атрибута, **ROWS** и **COLS**. Първия задава хоризонтално разделяне, втория вертикално. Атрибутите **ROWS** и **COLS** имат параметри за височина и ширина на рамката. Параметрите могат да се задават по три начина. Първия е в процент от размера на прозореца, при него след числото задължително се слага знака за процент %. В горния пример се ползва точно такова разделяне. Втория начин е в абсолютни единици - пиксели. Те се задават само с число без никакъв знак отзад. Например **ROWS="200,300"** разделя прозореца на 2 части, едната с височина 200 пиксела а другата 300. Третия начин е да се зададат относителни стойност. За целта се използва знака звездичка *. Например **ROWS="*,3*"** означава, че прозореца се разделя на две части, втората от които е три пъти по-голяма от първата. Параметрите се разделят със запетая и са заградени в кавички. Сега ще ви направя една такава страница, ще я разделя вертикално на 3 части, първата ще има ширина 100 пиксела, втората 30 процента от ширината на прозореца на браузъра, а третата ще заема всичкото свободно пространство, независимо колко е то. Понеже няма да задам да се отварят други страници във всяка рамка, те ще бъдат празни:

```
<HTML>
<HEAD>
  <TITLE>рамки</TITLE>
</HEAD>
<FRAMESET COLS="100,30%,*">
  <FRAME NAME="first" >
  <FRAME NAME="second">
```

```
<FRAME NAME="third">
</FRAMESET>
</HTML>
```

Натиснете [ТУК](#) за да видите как изглежда такава страница. Всяка рамка се създава с елемента **<FRAME>**. Чрез него се задават атрибутите на рамката. Един от тях е името и чрез атрибута **NAME**. Как се използва името ще обясня след малко. Друг атрибут на **<FRAME>** е **SRC**, следван от името на страницата която искате да се показва в тази рамка. Ако обърнахте внимание, след като стартирахте страницата в горния пример, всяка рамка си има някакъв ръб(border). Ако посочите с курсора на мишката върху ръба, кликнете върху него и докато държите натиснат бутона влачите мишката перпендикулярно на ръба, то размера на рамката се променя. Можете да забраните това с атрибура **NORESIZE** на елемента **<FRAME>**. Също така можете да зададете дебелината на ръба в пиксели с атрибута **BORDER** и цвета му с **BORDERCOLOR** на елемента **<FRAMESET>**. Нека направим предишния пример така, че да забраним промяната на размера на рамката и да направим ръба със син цвят и широк 6 пиксела.

```
<HTML>
<HEAD>
  <TITLE>рамки</TITLE>
</HEAD>
<FRAMESET COLS="100,30%,*" BORDER="6" BORDERCOLOR="0000FF">
  <FRAME NAME="first" NORESIZE>
  <FRAME NAME="second" NORESIZE>
  <FRAME NAME="third" NORESIZE>
</FRAMESET>
</HTML>
```

Вижте как ще изглежда тази страница. Натиснете [ТУК](#). Ако зададете **BORDER="0"** ръба на рамката няма да се вижда. Можете да задавате ширината и цвета на ръба на всяка рамка поотделно. Ширината се задава с атрибута **FRAMEBORDER** на елемента **<FRAME>**, а цвета с **BORDERCOLOR**.

Рамките могат да се влагат една в друга. Например, можете да разделите страницата на две части хоризонтално, а долната част да разделите на още 2 и т.н.. Вижте как става:

```
<HTML>
<HEAD>
  <TITLE>рамки</TITLE>
</HEAD>
<FRAMESET ROWS="100,*" >
  <FRAME NAME="first" NORESIZE>
```

```
<FRAMESET COLS="20%,80%">
  <FRAME NAME="second" NORESIZE>
  <FRAME NAME="third" NORESIZE>
</FRAMESET>
</FRAMESET>
</HTML>
```

Така горната рамка е по цялата дължина на прозореца на браузъра и има ширина 100 пиксела, а долната е разделена на две вертикални части, съответно на 20 и 80 процента от ширината на прозореца. Натиснете [ТУК](#) за да видите тази страница. Ако искате можете да разделите страницата първо вертикално на две, като втората рамка я разделяте допълнително на две хоризонтални рамки:

```
<HTML>
<HEAD>
  <TITLE>рамки</TITLE>
</HEAD>
<FRAMESET COLS="100,*" >
  <FRAME NAME="first" NORESIZE>
  <FRAMESET ROWS="20%,80%">
    <FRAME NAME="second" NORESIZE>
    <FRAME NAME="third" NORESIZE>
  </FRAMESET>
</FRAMESET>
</HTML>
```

Вижте [ТУК](#) как изглежда. Сега нека направим една наистина работеща страница с рамки. Нека страницата да бъде от две рамки. Първата да се казва "menu" и да съдържа хиперлинкове до известни български сайтове. Втората рамка с име "main" ще бъде празна, като линковете от първата ще се показват във втората. За целта първо трябва да направим html документа който да се показва в първата рамка :

```
<HTML>
<HEAD>
  <TITLE>menu</TITLE>
</HEAD>
<BODY>
  <A HREF="http://www.search.bg">search.bg</A><BR>
  <A HREF="http://www.dir.bg">dir.bg</A><BR>
  <A HREF="http://www.gyuvetc.bg">гювеч.бг</A>
```


</BODY>

</HTML>

Запазете този файл като menu.html. Неговата функция е да ни препраща към сайтовете при кликане върху надписите search.bg , dir.bg и гювеч.бг . Този html документ ще се визуализира в лявата рамка. Дясната рамка първоначално ще я оставим празна. Нека сега да направим html документа който да раздели страницата на рамки :

<HTML>

<HEAD>

<TITLE>frame</TITLE>

</HEAD>

<FRAMESET COLS="100,*" BORDER="0">

<FRAME NAME="menu" SRC="menu.html" TARGET="main">

<FRAME NAME="main">

</FRAMESET>

</HTML>

Този html документ го запазете като frame.html. Чрез него разделяме страницата на две рамки вертикално. Едната е широка 100 пиксела, а втората заема останалата част от прозореца на браузъра. В първата рамка сме задали да се покаже файлът menu.html който направихме за тази цел по-горе. На тази рамка сме дали името "menu". Новото тук е атрибута <TARGET> на елемента <FRAME>. този атрибут показва на браузъра в коя точно рамка да се отворят хиперлинковете, които съдържа страницата в тази рамка. В този случай на втората рамка сме дали име "main" и атрибута <TARGET> от първата рамка сочи именно към втората. Натиснете [ТУК](#) за да видите резултата.

Понеже сме забранили да се показва ръба на рамките (BORDER="0"), тази страница ще изглежда сякаш няма рамки. Когато кликнете върху някой от хиперлинковете обаче, в дясната част на страницата(т.е. във втората рамка) ще се отваря съответния сайт.

Ако страницата която се отваря в рамката е по-голяма от размера на самата рамка, браузъра по подразбиране показва хоризонтален или вертикален плъзгач(scrollbar) с помощта на който можете да разглеждате цялата страница. Този плъзгач излиза в дясната или долната част от рамката, или и в двете в зависимост от големината на страницата. Вие можете да определите дали този плъзгач да се показва или не с атрибута SCROLLING на елемента <FRAME>. Този атрибут от своя страна има три параметъра. SCROLLING="yes" задължава рамката винаги да показва плъзгачите, независимо дали са нужни или не. SCROLLING="no" забранява показването им във всички случаи. Тази забрана обаче би била много неудобна ако страницата е по голяма от рамката, защото няма да можете да я видите цялата. SCROLLING="auto" е стойността по подразбиране. С параметър "auto" плъзгачите се показват само когато има нужда от тях. Сега нека направим

една страница с две рамки. И в двете да се показва сайта на сърч.бг, само че в първата плъзгачите да са забранени, а във втората да се показват задължително.

```
<HTML>
<HEAD>
  <TITLE>frame</TITLE>
</HEAD>
<FRAMESET COLS="50%,50%">
  <FRAME NAME="search1" SRC="http://www.search.bg" SCROLLING="no">
  <FRAME NAME="search2" SRC="http://www.search.bg" SCROLLING="yes">
</FRAMESET>
</HTML>
```

Вижте [ТУК](#) как изглежда.

Какво научихте за рамките :

1. Рамките разделят страницата ви на отделни части, във всяка от които се показва друга страница.
2. Можете да задавате големината на рамката, дебелината и цвета на ръба, както и да забранявате или разрешавате промяната на размера на рамката.
3. Можете да насочвате хиперлинкове от една рамка да се показват в друга рамка.

МЕТА елементи

В главата на HTML документа HEAD освен заглавието на web страницата може да се поставя информация, която се използва от търсещите машини. Това става с помощта на META таговете. Синтаксиса на META елемента е следния :

```
<META NAME="keywords" VALUE="..стойности...>
```

<META> тагът няма съответния затварящ елемент </META>. Нека направим сега една страница, като използваме META тагове :

```
<HTML>
<HEAD>
  <TITLE> META tags </TITLE>
  <META NAME="author" VALUE="Peter Svetoslavov">
  <META NAME="description" VALUE="страница, съдържаща META елемент">
  <META NAME="keywords" VALUE="html, meta, tag, dokument">
</HEAD>
<BODY>
</BODY>
</HTML>
```

Атрибута **NAME** определя типа на информацията за страницата, а **VALUE** нейната стойност.

В примера **NAME="author" VALUE="Peter Svetoslavov"** показва на търсачката по **META** характеристики че автора на страницата е Петър Светославов.

NAME="description" задава описанието на страницата. Винаги се старайте описанието да бъде кратко и ясно.

NAME="keywords" задава ключови думи при търсенето. След **VALUE** трябва да изпишете всички ключови думи, характерни за страницата ви, като внимавате да не се повтарят. Старайте се да обхванете всичко. Така страницата ви ще бъде по-лесно откриваема за търсачките.

META таговете могат да извършват и действия, които не са свързани със съдържанието на страницата. Ето някои от тях :

<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-1251">

Този META елемент зарежда предварително кодовата таблица на символите, в случая кирилица. Изписвайте този ред винаги, когато страницата ви е на кирилица. В противен случай някои браузър може да покаже маймунски символи.

<META HTTP-EQUIV="Content-Language" CONTENT="bg"> - указва че съдържанието на страницата е на български език.

<META HTTP-EQUIV="refresh" CONTENT="5"> - презарежда страницата през определен период от време, в случая 5 секунди.

<META HTTP-EQUIV="refresh" CONTENT="5;URL=http://www4u.search.bg" > - пренасочва към друга страница след определен период от време, в случая след 5 секунди ви пренасочва към www4u.search.bg

<META HTTP-EQUIV="expires" CONTENT="friday", 21 July 2000, 15:23:22 GMT"> - показва, че съдържанието на страницата след тази дата и час вече не е актуално и забранява четенето от кеша на диска.

СПЕЦИАЛНИ СИМВОЛИ

Има символи, които не бихте могли да визуализирате в HTML документа просто като ги напишете. Представете си, че искате да покажете на екрана символа **<**. Ако решите да напишете следното:

<HTML>

<HEAD>

<TITLE>специални символи</TITLE>

</HEAD>

<BODY>

<

</BODY>

</HTML>

..вие просто ще объркате брауъра, защото когато срещне символа <, той решава че това е началото на елемент. За тази цел вие трябва да използвате така наречените специални символи. Това са символи, които могат да объркат брауъра, като <,>,#,&,\$ и т.н., или пък не можете да наберете директно от клавиатурата, като №,®, и др.

Специалните символи започват със знака & и завършват с ;, например символа за < е < така че горния HTML документ е правилно да се напише така :

```
<HTML>
<HEAD>
  <TITLE>специални символи</TITLE>
</HEAD>
<BODY>
  &lt;
</BODY>
</HTML>
```

Ето и някои от най-често използваните специални символи :

! ;	"	# ;
$;	%	&
: ;	;	<
>	?	@
™ ;	 (интервал)	©
®	±	½ ;

Обърнете внимание, че на някои символи отговаря специален символ име, а на други число. Числото е ASCII кода на съответния символ. На всеки специален символ отговаря някакво число от ASCII таблицата. Например на ® отговаря ® , така че дали ще напишете ® или ® ефекта ще е същия, показването на символа ®.

Cascading style sheets

Cascading style sheets е нов и удобен начин за форматиране на текста, шрифтовете, изображенията и всичко останало на вашата страница. Cascading style sheets ви позволяват да разположите различните елементи на страницата където си поискате, с точност до последния пиксел. Например, ако промените един стил, зададен в началото на HTML страницата, промените се отразяват върху целия документ.

Да кажем, че сте направили стил за заглавния таг (`<H3>`). Във вашия стил настройвате цвета на текста, на всичките `<H3>` да бъде червен. Написали сте около десет `<H3>` елемента, но изведнъж решавате, че цветът на заглавията трябва да е син. Е, вече не се налага да се връщате назад и да променяте цвета на текста за всеки `<H3>`, ами просто променяте стила, който сте създали в главата (`<HEAD>`) на вашия HTML документ. Сега заглавията ще бъдат в син цвят с много по-малко писане.

Още по-удобно е да използвате външен style sheet за няколко HTML страници едновременно. Може например, да направите заглавията във всичките ви страници да са сини. По този начин не се налага да променяте цвета за всяка страница по отделно, ами просто ще укажете всяка да използва вашия стил за заглавията.

Използване на атрибута STYLE

Един от начините да прибавите стил към страницата, е да го зададете вътре в някой HTML елемент. Това става като прибавите атрибута `STYLE=""` към HTML елемента. Обичайно това става по следния начин:

`STYLE="свойство: стойност"`

Ако не разбирате, за какво става дума вижте следния пример. Да кажем, че искате цветът на текста за елемента `<P>` да е червен:

```
<P style="color: red">Аз съм червен текст, благодарение на CSS.</P>
```

Ето какво ще получите:

Аз съм червен текст, благодарение на CSS.

Естествено може да зададете и повече от едно свойство в `STYLE` атрибута.

Например:

```
<P style="color: red; font-weight: bold; font-family: Arial"> Удебелен шрифт Arial.</P>
```

Сега имаме удебелен, червен текст с шрифт Arial:

Удебелен шрифт Arial.

Може да добавяте колкото искате свойства в `STYLE` атрибута, стига да отделяте всяко с точка и запетая.

Сигурно ще си кажете - Как да сложа свойства като неznam нито едно?! Не се тревожете, по-нататък в уроците са разгледани всички свойства, а има и пълен справочник на свойствата и техните стойности.

Задаване на стил в главата (<HEAD>) на HTML документа.

И така, изяснихме **STYLE** атрибута, но както забелязахте той едва ли прави нещо по-лесно, от добре познатите HTML елементи. Сега ще обясня другия начин за задаване на стил в страницата, който ще направи животът ви по-лесен (и сънят ви по-дълготраен), защото няма да се налага да пишете едно и също нещо по десетки пъти за да постигнете желанния ефект. Задаването на стил става чрез **<STYLE></STYLE>** елемента. Ето един пример на стил за елемента ****:

```
<HEAD>
```

```
<STYLE>
```

```
<!--
```

```
SPAN { color: red; font-weight: bold }
```

```
-->
```

```
</STYLE>
```

```
</HEAD>
```

Както виждате, непосредствено след **<STYLE>** тага, слагаме знак за HTML коментар. Той скрива съдържанието на стила от браузърите, които не го разпознават.

Сега вижте този ред:

```
SPAN { color: red; font-weight: bold }
```

Той казва на браузъра, че текста във всички `` тагове на вашата страница ще червен и удебелен. Запомнете, че когато се декларира стил за някой HTML елемент, чрез `<STYLE></STYLE>`, не се използват знаците за по-малко и по-голямо - `< >`. Така `` става `SPAN`, `<P>` става `P`, `<TABLE>` става `TABLE` и тн. Освен това не използваме знакът за равенство и кавички - `=""`, за да зададем свойствата. Вместо това ги заграждаме в скоби - `{ }`. Свойствата в скобите са отделени с точка и запетая - `;`, както при `STYLE` атрибута.

Сега, след като имате горния пример в главата на HTML страницата, просто използвате ``, за да направите текста удебелен и с червен цвят:

```
<SPAN>Червен, bold-ван текст,</SPAN> следван от обикновен текст.  
<BR>  
<BR>  
<SPAN>Отново bold и червен.</SPAN>
```

Червен, bold-ван текст, следван от обикновен текст.

Отново bold и червен.

Някои версии на Netscape може и да не разпознаят свойствата за `` тага, затова се снабдете с 4.7, или най-добре с Internet Explorer 5.

И така, сега може да използвате свойствата на стиловете за почти всеки HTML елемент. По интересно приложения обаче, стиловете намират когато се комбинират с атрибутите `CLASS` и `ID`.

Деклариране на стилове, чрез класове

Както вече видяхте в предишния урок, стиловете могат да се прилагат за почти всеки HTML елемент. Да кажем обаче, че искате да направите половината текст на страницата червен, а другата половина - син. Ето един пример, какво можем да направим с наученото дотук:

```
<HEAD>
```

```
<STYLE>
```

```
<!--
```

```
DIV { color: red }  
P { color: blue }
```

```
-->  
</STYLE>
```

```
</HEAD>
```

```
<DIV>Червен текст.</DIV>  
<BR>  
<BR>  
<P>Син текст.</P>
```

Червен текст.

Син текст.

И така, постигнахме желания ефект. Но ако искате, например, на страницата да има и зелен, и оранжев цвят? Ще трябва да зададете стил за четири HTML елемента, което едва ли е много удобно. По-простия начин е да използвате класове. Вижте долния пример:

```
<HEAD>
```

```
<STYLE>  
<!--
```

```
.red { color: red }  
.blue { color: blue }
```



```
-->  
</STYLE>
```

```
</HEAD>
```

```
<P class="red">Червен текст.</P>  
<BR>  
<BR>  
<P class="blue">Син текст.</P>
```

Какво направихме? Създадохме два класа - "red" и "blue" - и им зададохме стил. След това указахме, елементът `<P>` веднъж да ползва стила "red" и после да ползва стила "blue". Това е изключително удобно, защото може да използвате няколко различни класове с един и същи HTML елемент. Класовете се задават в `<STYLE></STYLE>` елемента, в началото на страницата и се "извикват" като се прибави атрибута `CLASS` към съответния HTML елемент. Например:

```
<P class="red">Червен текст.</P>  
<P class="blue">Син текст.</P>  
<P class="green">Син текст.</P>  
<DIV class="black">Син текст.</DIV>  
<SPAN class="black">Син текст.</SPAN>
```

Обърнете внимание на следния ред:

```
.red { color: red }
```

Класовете се задават с точка - . - последвана от името на класа. Например:

```
.red, .blue, .blabla и тн.
```

Деклариране на стилове, чрез атрибута ID

И тук, стиловете се дефинират по същия начин само че, вместо точката пред името на стила, се използва знакът `#`. Например:

```
<HEAD>
```

```
<STYLE>
```

```
<!--
```

```
#red { color: red }  
#blue { color: blue }
```

```
-->
```

```
</STYLE>
```

```
</HEAD>
```

"Извикването" става, чрез атрибута **ID**:

```
<DIV id="red">Червен текст.</DIV>
```

```
<DIV id="blue">Син текст.</DIV>
```

Червен текст.

Син текст.

Ако искате вашите новосъздадени класове да се използват от много страници, може да използвате т.нар. външен Style Sheet.

Използване на външни Style Sheets

Всичко, което разгледахме в предишните уроци се отнасяше за страницата, която правим в момента. Класовете спестяват много писане и усилия в **текущия** HTML документ. Да предположим обаче, че искате определена група от стилове да се използва от много HTML страници едновременно. Това става чрез external(външни) style sheets.

За да направите external style sheet, отворете някой текстов редактор (например Notepad) и напишете примерно следното:

```
DIV { font-family: Arial }  
P { color: blue }
```

След това запазете файла (save as...) като "style.css" (слагаме името на файла в кавички, за да се запази с разширение *.css; иначе ще се запази като style.css.txt). Може да използвате всякави имена - style.css, bla.css, fish.css и тн. - но разширението трябва да е *.css.

Вашият external style sheet е готов. Сега, за да укажете на дадена страница да го използва просто трябва да сложите елемента `<LINK>` в главата (`<HEAD>`) на HTML документа. Най-общо това става по следния начин:

```
<LINK rel="stylesheet" type="text/css" name="име" href="адрес">
```

Атрибутът `rel="stylesheet"` казва на страницата какъв документ да търси (в нашия случай stylesheet). `type="text/css"` указва какъв е типът на съдържанието във файла, който ще се използва. `name=""` задава име (може да е всякакво) и `href=""` указва къде се намира файла (интернет адрес). За нашия пример, `<LINK>` елемента ще изглежда по следния начин:

```
<LINK rel="stylesheet" type="text/css" name="Pesho Dulgiq" href="style.css">
```

Адресът е само името на файла, ако сте го запазили в същата директория, където са и вашите страници. И така, за да ползвате вашият стил в други HTML документи, просто ги отворете и сложете горния ред в главата им. Ако искате да промените текста на няколко страници, просто редактирайте файла style.css и запазете промените.

Форматиране на текст с CSS

Тук ще обясня накратко как да форматирате текста на страницата ви, използвайки CSS (Cascading style sheets). Както споменах в [Урок I](#) CSS се състои от много свойства и съответните им стойности. Сега ще обърна внимание на свойствата отнасящи се до текста, а за списък на всичките свойства и стойности погледнете в [Речника](#).

Първото свойство, което ще разгледаме е `color`:

Свойство: COLOR

Стойности:

име на цвета;

цвят по схемата #RRGGBB;

цвят в RGB;

Свойството `color` определя цветът на текста за даден елемент. Например:

<P style="color: red">Червен цвят на текста</P>

<P style="color: #00FFFF">Светлосин цвят на текста</P>

<P style="color: rgb(0,0,0)">Черен цвят на текста</P>

Червен цвят на текста

Светлосин цвят на текста

Черен цвят на текста

Има три начина за задаване. Първият е задаване с името на цвета. Напр. **red**, **blue**, **gold**, **silver** и тн. Другият начин е задаване на цвета по схемата **#RRGGBB**. Например #FFFFFF (бяло), #000000 (черно), #FFFF00 (червено) и тн. Третият начин е задаване според дяла на червеното, зеленото и синьото. Например rgb(255,255,255) - бяло; rgb(0,0,0) - черно; rgb(255,0,0) - червено.

Свойство: FONT-FAMILY

Стойности:

име на шрифта;

семејство на шрифта;

Свойството **font-family** задава шрифта, който ще се ползва при представянето на текста. Например:

<P style="color: red; font-family: Arial"> Текст с шрифт Arial</P>

<P style="color: #00FFFF; font family: Courier">Текст с шрифт Courier</P>

<P style="color: rgb(0,0,0); font-family: serif">Текст със серифен шрифт</P>

Текст с шрифт Arial

Текст с шрифт Courier

Текст със серифен шрифт

За стойност може да зададете име на шрифта, например Arial, Helvetica, Verdana, или да се зададе само семејството на шрифта - serif, sans-serif, monospace, fantasy, cursive.

Свойство: FONT-STYLE

Стойности:

normal;

italic;

oblique;

Свойството **font-style** определя стила на шрифта. Например:

<P style="font-style: normal">Нормален шрифт</P>

<P style="font-style: italic">Наклонен шрифт (italic)</P>

<P style="font-style: oblique">Шрифт oblique</P>

Нормален шрифт

Наклонен шрифт (italic)

Шрифт oblique

Свойство: FONT-WEIGHT

Стойности:

абсолютни величини;

числови стойности;

Определя дебелината на шрифта:

<P style="font-weight: bold">Удебелен шрифт</P>

<P style="font-weight: normal">Шрифт с нормална дебелина</P>

<P style="font-weight: 100">Тънък шрифт</P>

Удебелен шрифт

Нормален шрифт

Тънък шрифт

Могат да се зададат както абсолютни величини (normal, bold, bolder), така и числови стойности (100, 200, 300 и тн. 400 отговаря на normal, а 700 на bold).

Свойство: FONT-SIZE

Стойности:

абсолютни величини;

стойност спрямо предходния размер;

типографски величини;

Свойството определя дебелината на шрифта. Например:

<P style="font-size: x-large">Много голям шрифт</P>

<P style="font-size: 12pt">Шрифт с големина 12 пункта</P>

<P style="font-size: 12px">Шрифт с големина 12 пиксела</P>

<P style="font-size: larger">Шрифт, една степен по-голям от предходния</P>

Много голям шрифт

Шрифт с големина 12 пункта

Шрифт с големина 12 пиксела

Шрифт, една степен по-голям от предходния

Свойството **font-size** има много стойности. Вижте [Речника](#) за по-подробно описание. Най-общо стойностите се делят на три групи - абсолютни стойности, стойности спрямо предходния размер и типографски величини. Абсолютните стойности са фиксирани размери, които имат име. Например: xx-small, medium, large и тн. Под стойности спрямо предходния размер се разбира една степен по-голям или по-малък шрифт от предходния. Например, ако имаме шрифт 14 пункта, със стойността larger ще зададем шрифт с големина 15 пункта. Последната група от стойности е

най-използваната. Това са типографски величини за определяне на размера. Например: pt (пунктове), px (пиксели), em (коефициент спрямо актуалния размер), проценти.

Свойство: TEXT-DECORATION

Стойности:

none;

underline;

overline;

line-through;

blink;

Това свойство добавя някои ефекти към текста. Например:

```
<P style="text-decoration: underline">Подчертан текст</P>
```

```
<P style="text-decoration: line-through">Зачеркнат текст</P>
```

```
<P style="text-decoration: none">Текст без декорация</P>
```

Подчертан текст

Зачеркнат текст

Текст без декорация

Всичките стойности са: none, underline, overline, blink, line-through. Някои от тях обаче не работят в Netscape. Това свойство е полезно ако искате да се отървете от досадното подчертаване на линковете. Например:

```
<A style="text-decoration: none" href="#">Неподчертан линк</A>
```

Неподчертан линк

В Internet Explorer дефинирането на стилове за линкове става по следния начин:

```
<STYLE type="text/css">
```

```
<!--
```

```
A:link { text-decoration: none }
```

```
A:visited { text-decoration: none }
```

```
A:hover { text-decoration: underline }
```

```
-->
```

```
</STYLE>
```

A:link определя свойствата само за линковете, отделно от `<A>`

елемента. A:visited се отнася за посетените линкове, A:hover определя

свойствата, когато минете с мишката върху линка (не работи в Netscape).

Свойства на CSS1

Тук ще изброя всички свойства и възможните им стойности, включени в CSS1, както и някои нововъведения в CSS2. За по-бърза навигация натиснете тук, за да изберете някое свойство.

Мерни единици
Background
d
Border
Clear
Color
Display
Float
Font
Height
Letter-spacing

Скрий менюто

Мерни единици, означения

Мерни единици, означения		
Д ъ л ж ин и	р х	Пиксели.
	е х	Височината на буквата х.
	pt	Пунктове. Величина за определяне размера на шрифта.
	р с	Пики. 1рс отговаря на 12pt.
	in	Цол (на англ. инч). 1in = 2,54cm.
	с т	Сантиметри.

	m m	Милиметри.
П р о ц е н т и		Обозначават се със знак за процент след числото.
Ц в е т о в е	И м е	Име на цвета. Например red, gold, green.
	С т о й н о с т	Например #FFFFFF (виж HTML/Цветове).
	rg b	Според дяла на червеното, зеленото и синьото. Напр. rgb(255,255,255).
U R L		Комплексен Интернет-адрес. Например: { background: url("main.jpg") }.

[Народе](#)

Background

Background-color		
Ст ой но сти	цв ят	Цветът се задава чрез един от трите описани по-горе начини (вж. Мерни единици, означения).
	tra ns pa re nt	Не се дефинира никакъв цвят.

Обяснение	Дефинира фоновия цвят на HTML-документа.
Пример	{ background-color: #606060 }

Background-image	
Стойности	Отказ от фоново изображение.
	Адрес на файла за изобразяване (вж. Мерни единици, означения).
Обяснение	Зарежда фоново изображение за актуалния документ, което се установява върху нормалния фон цвят.
Пример	{ background-image: url("main_bg.jpg") }

Background-repeat	
Стойности	no-repeat Няма повторение на изображението. Графиката се представя еднократно.

	r e p e a t - x	Повторението е изключително хоризонтално.
	r e p e a t - y	Повторението е изключително вертикално.
	r e p e a t	Хоризонтално и вертикално повторение.
Обяснение		Определя дали фоновото изображение трябва да се повтаря или да се престава еднократно.
Пример		{ background-repeat: no-repeat }

Background-attachment		
Становности		Фоновото изображение се превърта с останалото съдържание на документа.

		Изображението има фиксирана позиция и не се влияе от превъртането на съдържанието.
Обяснение		Определя дали фоновото изображения трябва да се фиксира неподвижно на своята позиция или не.
Пример		{ background-attachment: fixed }

Background-position		
Стойности	percent	Чрез процентна стойност - 0% 0%: горен, ляв ъгъл. 100% 100%: долен, десен ъгъл.
	center	Центриране в средата на Web страницата.
	top	Подравняване спрямо горния край на прозореца.
	bottom	Подравняване спрямо долния край на прозореца.
	left	Подравняване спрямо левия край на прозореца.

	ri g ht	Подравняване спрямо десния край на прозореца.
Обяснение		Позиционира фоновото изображение чрез една X/Y-двойка стойности. Например left/top или 10%/30%.
Пример		{ background-position: 50% 50% }

Също така може да използвате и обобщеното свойство **background**, за да дефинирате всичките подсвойства едновременно. Например:

```
<STYLE type="text/css">
```

```
<!--
```

```
P { background: url("main_bg.jpg") blue 50% 50% no-repeat fixed }
```

```
-->
```

```
</STYLE>
```

[Народе](#)

Border

Border-top-width		
Ст ой но сти	thin	Тънка линия.
	medium	Средно дебела линия.
	thick	Дебела линия.
	дължина	Задава се дължина.
Обяснение	Определя ширината на горната част на рамката.	

Пр им ер	{ border-top-width: medium }
----------------	------------------------------

Border-bottom-width		
Ст ой но сти	thin	Тънка линия.
	medium	Средно дебела линия.
	thick	Дебела линия.
	дължина	Задава се дължина.
Об яс не ни е	Определя ширината на долната част на рамката.	
Пр им ер	{ border-bottom-width: thin }	

Border-left-width		
Ст ой но сти	thin	Тънка линия.
	medium	Средно дебела линия.
	thick	Дебела линия.
	дължина	Задава се дължина.
Об яс не ни е	Определя ширината на лявата част на рамката.	
Пр им ер	{ border-left-width: 3px }	

Border-right-width		
Ст ой но сти	thin	Тънка линия.
	medium	Средно дебела линия.
	thick	Дебела линия.
	дължина	Задава се дължина.
Обяснение	Определя ширината на дясната част на рамката.	
Пример	{ border-right-width: thick }	

Border-width		
Ст ой но сти	thin	Тънка линия.
	medium	Средно дебела линия.
	thick	Дебела линия.
	дължина	Задава се дължина.
Обяснение	Определя ширината на рамката.	
Пример	{ border-width: 3px }	

Border-width обединява четирите описани по-горе свойства. Затова за **Border-width** могат да се зададат до четири стойности:

- Ако зададете една стойност всички елементи на рамката получават еднаква ширина. Например: **{ border-width: 2.5mm }**
- При две стойности, първата се отнася за горната и долната, а втората за лявата и дясната част на рамката.
- При три стойности първата се отнася за горната част на рамката, втората се отнася за лявата и дясната, а третата - за долната част. Например:

```

.
. <STYLE type="text/css">
. <!--
.
. DIV.bord { width: 200; border-width: 2mm 3mm 1mm; border-color: black;
. }
.
. -->
. </STYLE>
.
. <BODY>
.
. <DIV class="bord">Рамки,рамки,рамки</DIV>
.
. </BODY>
.






```

Рамки, рамки, рамки

- При четири стойности индивидуално се настройват всички части на рамката.

Border-color		
Ст ой но сти	стойн ост	По схемата #RRGGBB (вж. HTML/Цветове).
	име	Име на цвета. Например red, blue, white.
Об яс не ни е	Дефинира цвета на рамката.	
Пр им ер	{ border-color: #00FFFF }	

Border-style		
Ст ой но сти		Рамката е невидима.

		Прекъснатата линия.
		Линия, състояща се от малки черти (пунктир).
		Непрекъснатата линия.
		Двойна линия.
		Линия с триизмерен изглед. Може да бъде още и ridge , inset или outset .
Обяснение	Настройване на външния вид на рамката.	
Пример	<code>{ border-style: double }</code>	

Освен **border-top-width** може да използвате и **border-top-style** или **border-top-color**, за да настроите цвета или стила само за една част от рамката. Това работи само с Internet Explorer затова не Ви го препоръчвам. И тук, както при **background**-свойствата, може да използвате обобщеното **border**, за да настроите едновременно цвета, стила и големината на рамката. Например:

```
<STYLE type="text/css">
<!--

P { border: solid medium black }

-->
</STYLE>
```

Други полезни обобщителни свойства са **border-top**, **border-bottom**, **border-left**, **border-right**. С тях може да настройвате едновременно стила, големината и цвета само за определена част от рамката. Синтаксисът е както при горния пример.

[Народе](#)

Clear

Clear		
Ст ой но сти		Други елементи са допустими от всички страни.
		При всеки друг елемент, дадения елемент се позиционира в левия край.
		При всеки друг елемент, дадения елемент се позиционира в десния край.
		Елементът се позиционира при всеки друг елемент.

Обяснение	Свойството определя дали около даден елемент може да се появяват и други.
Пример	{ clear: none }

[Нагоре](#)

Color

Color		
Стройности	с т о й н о с т	По схемата #RRGGBB (вж. HTML/Цветове).
	и м е	Име на цвета. Например red, blue, white.
	R G B	Дял на червеното, зеленото и синьото. Например rgb(255,255,255) е бял цвят.
Обяснение	Описва цвета на текста за даден елемент.	
Пример	{ color: rgb(130,75,183) }	

[Нагоре](#)

Display

Display		
Стойности		Елементът и рамката му не се представят.
		Елементът се представя в рамка.
		Елементът се представя в рамка, вмъкната в текста, на същият ред като предходния елемент.
		Съответства на block, но към рамката се поставя и знак за списък.
Обяснение		Описва как трябва да се представи даден елемент на Web страницата.
Пример		{ display: block }

Float

Float		
Ст ой но сти		Елементът се представя на мястото, където се вгражда в документа.
		Подравяване вляво; Текстът се намира около елемента.
		Подравняване вдясно; Текстът се намира около елемента.
Об яс не ни е	Дефинира посоката на основния текст около даден елемент.	
Пр им ер	{ float: none }	

[Народе](#)

Font

Font-family		
Ст ой но сти	им е	Името, под което шрифтът е инсталиран. Например: Arial, Courier, Times New Roman.
	се ме йс	Семейството на шрифта. Например: serif, sans-serif, cursive, fantasy, monospace.

	ТВО	
Обяснение	Настройване на шрифта, който браузърът да зареди, за даден елемент.	
Пример	{ font-family: Arial, Verdana, Helvetica, sans-serif }	

Font-style		
Стойности	normal	Нормален шрифт.
	italic	Наклонен шрифт.
	oblique	Наклонен шрифт.
Обяснение	Определя стила на шрифта.	
Пример	{ font-style: italic }	

Font-variant		
Стойности	normal	Нормален шрифт.
	small-caps	Всички малки букви се заменят с големи, но те са по-малки от нормалните големи букви.

	l - c a p s	
Обяснение	Определя дали шрифтът да е нормален или да е т.нар. small-caps-шрифт.	
Пример	{ font-variant: small-caps }	

Font-weight		
Стойности	с т о й н о с т	Установени числови стойности. Например: 100, 200, 300 и тн. 400 = normal, 700 = bold.
	l i g h t e r	По-тънък от нормалния шрифт.
	n o r m a l	Нормален шрифт.
	b o	Удебелен шрифт.

	l d	
	b o l d e r	По-удебелен шрифт.
Обяснение	Дефинира дебелината на шрифта.	
Пример	{ font-weight: bold }	

Font-size			
Ст ой но сти	Абсолютни и размери	x x- s m a l l	Още по-малък шрифт.
		x- s m a l l	По-малък шрифт.
		s m a l l	Малък шрифт.
		m e d i	Нормален шрифт.

		u m	
		la rg e	Голям шрифт.
		x- la rg e	По-голям шрифт.
		x x- la rg e	Още по-голям шрифт.
	Спря мо пред ходн ия шри фт	la rg er	По-голям шрифт от предходния.
		s m al le r	По-малък шрифт от предходния.
	Типо граф ски вели чини	pt	Пунктове (вж. мерни единици, означения).
		e m	Коефициент спрямо актуалния размер. 1.5em = един размер и половина.
		п р о ц е н т и	Също коефициент. 150% отговарят на един размер и половина.
	Об яс не	Дефинира дебелината на шрифта.	

ни е	
Пр им ер	{ font-size: 12pt }

Естествено и тук има обобщаващото свойство **font**. Например:

P.neshtosi { font: bold small-caps 16pt Arial, sans-serif }

[Наро̀е](#)

Height

Height		
Ст ой но сти	а у т о	Дължината се променя автоматично спрямо съдържанието на елемента.
	д ъ л ж и н а	Вж. мерни единици, означения . Дължината не трябва да е отрицателна.
Об яс не ни е	Определя височината на даден елемент.	
Пр им ер	{ height: 300 }	

[Наро̀е](#)

Letter-spacing

Letter-spacing

Ст ой но сти	п о р т а л	Няма промяна.
	д ъ л ж и н а	Вж. мерни единици, означения . Може да се задват и отрицателни дължини.
Об яс не ни е	Определя разстоянието между отделните букви и знаци.	
Пр им ер	{ letter-spacing: 3 }	

Запознаване с JavaScript.

JavaScript е най-широко разпространеният език за програмиране в интернет, след HTML. Въпреки че го нарекох "език за програмиране", с негова помощ не се пишат програми, а скриптове които се вмъкват в HTML документа. В този смисъл JavaScript е език за писане на скриптове, докато JAVA е език за програмиране. Освен съвпадението в част от името, двата езика нямат кой знае какви прилики, дори са разработени от различни корпорации (JAVA е дело на SUN, а JavaScript е разработка на Netscape). JAVA е мощен език за програмиране не само на интернет приложения, но и на самостоятелни програми за различни платформи. Интернет приложенията на JAVA се наричат аплети. Те са файлове с разширение .class и се вмъкват в HTML документа между таговете **<APPLET>** и **</APPLET>**. Тук няма да се спираме подробно на JAVA аpletите.

Нека разгледаме възможностите на JavaScript, какво можете и какво не можете да правите с него :

- Ефекти с изображения. Rollover ефекти, слайд шоу, и много други.
- Управление на прозорци и рамки. Отваряне и затваряне на прозорци, задаване на размера на прозорец, управление на един прозорец от друг и т.н.

- Разпознаване на типа на браузъра, операционната система, разделителната способност на екрана и дълбочината на цветовете.
- Много други работи :-)

Какво не можете да правите с помощта на JavaScript :

- Не можете да записвате информация на сървъра (не можете да организирате форуми, да обработвате бази данни)

JavaScript кода се вмъква в HTML документа между двойката елементи `<SCRIPT>` и `</SCRIPT>`. Когато срещне тагът `<SCRIPT>`, браузъра разбира че трябва да спре интерпретирането на HTML кода и да започне да обработва скрипта, намиращ се между `<SCRIPT>` и `</SCRIPT>`. Този скрипт не е задължително да бъде написан на JavaScript. Има и други езици за писане на скриптове, например VBScript. Затова когато пишете отварящия таг за скрипт, трябва да укажете на браузъра на какъв език ще бъде скрипта. Ако този език е JavaScript трябва да напишете `<SCRIPT LANGUAGE="JavaScript">`. Нека да направим една уеб страница в която да вмъкнем JavaScript код, който да изписва на екрана "Здравей!"

```
<HTML>
<HEAD>
<TITLE>javascript</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
  document.write("Здравей!")
</SCRIPT>
</BODY>
</HTML>
```

За Ваше улеснение от сега нататък ще изписваме HTML таговете с червени букви, а JavaScript операторите със сини.

Препишете горния код в някой текстов редактор и запазете файла като javascript.html. След това го стартирайте с някой браузър.

! Някои по-стари версии на Netscape Navigator и Internet Explorer не разбират JavaScript. Затова трябва да скривате скрипта в HTML коментари <!-- и //-->. Така по старите версии на браузърите ще помислят JavaScript кода за коментар и няма да съобщят за грешка.

Ето как ще изглежда горния HTML документ, но със скрипт заграден с коментари

:

```
<HTML>
<HEAD>
<TITLE>javascript</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
  <!--
  document.write("Здравей!")
  //-->
</SCRIPT>
</BODY>
</HTML>
```

Обърнете внимание, че JavaScript операторите се пишат с малки букви. Горния пример не прави нищо особено. Той самоизписва на екрана надписа "Здравей!". Можете да форматирате текста, като в javascript кода вмъкнете html таг:

```
<HTML>
<HEAD>
<TITLE>javascript</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
    <!--
    document.write("<B>Здравей!</B>")
    //-->
</SCRIPT>
</BODY>
</HTML>
```

Така текста ще се покаже удебелен. Текстовете в javascript се изписват между кавички " и " .

Променливи

Променливите в javascript са от типа величини, които могат да променят стойността си. Другия тип величини са константите. Те никога не променят стойността си. В предишния пример текста "Здравей!" е константа. Нека да направим предния пример, като използваме променлива :

```
<HTML>
<HEAD>
<TITLE>javascript</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
    <!--
    greeting="Здравей!"
    document.write(greeting)
    //-->
</SCRIPT>
</BODY>
</HTML>
```

Тук името на променливата е `greeting`, а стойността и е "Здравей!". Така когато отпечатваме променливата `greeting` ние всъщност отпечатваме стойността и "Здравей!".

При формирането на имената на променливите в javascript трябва да се спазват някои правила. Името може да съдържа големи и малки латински букви, цифри, както и знака за подчертаване `_`. Всички останали символи са забранени. Имената на променливите трябва да започват винаги с буква или знак за подчертаване. Те не трябва да започват с цифра. Ето няколко правилни имена на променливи : `greeting_card` `greet2` `_my_greeting`

Неправилни са : `123` `var.2` `greeting 3` -първата започва с цифра, втората съдържа непозволен символ точка, а в третата има интервал. Нека направим още един пример с променливи :

```
<HTML>
<HEAD>
<TITLE>javascript</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
    <!--
    a=5
    b=7
    greeting="Здравей!"
    document.write(greeting+a)
    document.write("<br>")
    document.write(a+b)
    //-->
</SCRIPT>
</BODY>
</HTML>
```

Тук имам три променливи, `a`, `b` и `greeting`. На първата променлива присвояваме стойност 5 (`a=5`) на втората 7 (`b=7`) а на третата "Здравей!" (`greeting="Здравей!"`). Тук е момента да ви обясня за типовете променливи. В общи линии те се разделят на целочислени, стрингови, булеви и други, които ще разгледаме по-нататък. Първите две променливи, `a` и `b` са от целочислен тип, защото на тях се присвояват цели числа. Третата променлива `greeting` е от стрингов тип. Към този тип променливи се присвоява някакъв текст (стринг) заграден с кавички или апострофи.

С променливите могат да бъдат извършвани математически операции, като събиране, изваждане, умножение, деление и т.н. Нека се върнем на

горния пример. В началото на всяка променлива присвояваме някаква стойност, на `a=5`, `b=7` и `greeting="Здравей!"`. В следващия ред от скрипта се извършва операцията събиране на две променливи и резултата се показва на екрана(`document.write`) Обърнете внимание, че първите две променливи които събираме са от различни типове, едната е от целочислен другата от стрингов тип. Когато се извършат математически операции в които участват променливи от различни типове и едната е от тип стринг, резултата винаги е от тип стринг. Така че събирането на двете променливи `greeting` и `a` ще доведе до резултат `"Здравей!5"`. Събирането на целочислени променливи, както и на такива с плаваща запетая (в математиката им казват "дроби" :-) води до резултат друго число, което означава, че събирането на `a` и `b` ще бъде равно на 12 (`a=5`, `b=7`, `a+b=12`). Запазете горния пример в отделен HTML документ и го стартирайте в браузъра. На екрана трябва да видите следното :

Здравей!5

12

Забележете че резултата от събирането на `greeting` и `a` е `Здравей!5` , т.е. двете променливи се "слепват". Ако искате между тях да има интервал трябва да включите празен стринг `" "`. Това са две кавички една след друга. Ако сега напишете `document.write(greeting+" "+a)` резултата ще бъде `Здравей! 5` с интервал между тях. Обърнете внимание на преминаването на нов ред - `document.write("
")`. Когато вътре в скрипта включите HTML таг, той се изпълнява.

Математически операции

Както вече разбрахте, можете да извършвате различни математически операции с променливите от целочислен тип, както и с променливите с плаваща запетая. Операторите са следните :

+	събиране
-	изваждане
*	умножение
/	деление
%	целочислено деление

Когато изпълнявате математическа операция с повече числа, можете да използвате скоби за да разграничите приоритетите на изчисленията. Например ако искате да съберете 2 със 5, а после да разделите резултата на три трябва да напишете `(2+5)/3` .Ако изпуснете скобите ще се получи `2+5/3`. Тогава първо 5 ще се раздели на 3, а после към резултата ще се прибави 2. Запомнете, че умножението и делението са по голям приоритет от събирането и изваждането.

Нека да направим един скрипт в който въвеждате годината в която сте родени, след това да изчислява на колко години ще бъдете през 2010 година и най-накрая да се отваря прозорец който да съобщава резултата :

```
<HTML>
<HEAD>
    <TITLE>javascript</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="javascript">
<!--
    var year=prompt("Въведете година на раждане","1980")
    old=2010-year
    alert("През 2010 година Вие ще бъдете на "+old+" години")
//-->
</SCRIPT>
</BODY>
</HTML>
```

var е друг начин да се обявяват променливите (var name="Ivan" , var number=14). Със prompt се отваря прозорец с поле, в което можете да въвеждате информация. Структурата на prompt е следната - prompt("message","value") където message е съобщението което ще се появи заедно с прозореца, а value е някаква начална стойност. Можете да пропуснете value, тогава полето ще бъде празно. В примера чрез prompt присвояваме на променливата year стойността която ще напишете в полето. В следващия ред създаваме променлива old на която присвояваме стойност 2010 минус стойността на year. Ако оставите в полето по подразбиране числото 1980, то тогава year ще бъде равно на 1980, а old на 30 (2010-1980). И най-накрая извеждаме съобщение, че през 2010 г. вие ще бъдете на old години, като променливата old се заменя от стойността и. Вижте как работи този скрипт [ТУК!](#) Самото съобщение се извежда с alert прозорец (alert("През 2010 година Вие ще бъдете на "+old+" години")).

Този примерен скрипт не прави проверка за верността на въведените от потребителя данни, така че ако въведете някакъв стринг вместо число скрипта ще даде грешка.

Други аритметични операции с променливи са събиране с едно ($x++$) и изваждане с едно ($x--$). $x++$ е същото като $x=x+1$, а $x--$ като $x=x-1$. Ако например променливата x има стойност 5, след $x++$ x ще е равно на 6.

Ако присвоявате стойността на x на друга променлива, например y и в същото време увеличите x с едно, стойността на двете променливи ще бъде една и съща.

```
y=x++
```

..така x и y ще бъдат с еднаква стойност. Ако напишете обаче :

```
y=++x
```

.. y ще приеме стойността на x и чак след това x ще се увеличи с 1. Значи да приемем че $x=10$. Тогава при $y=x++$ x ще стане 11 и y ще стане 10. Ако напишем $y=++x$ тогава y ще стане 11, а x 12.

Булеви изрази. Условия.

Булевите изрази са операции за сравнение между две променливи. Резултата от такова сравнение има само две стойности - **true** или **false**. Например при сравнение дали променливата a е по-голяма от променливата b се връща резултат **true**, ако е истина или **false** ако не е истина.

Ето списък на булевите изрази :

$x==y$	ако x е равно на y стойността е true
$x!=y$	ако x не е равно на y стойността е true
$x<y$	ако x е по-малко от y стойността е true
$x<=y$	ако x е по-малко или равно на y стойността е true
$x>y$	ако x е по-голямо от y стойността е true
$x>=y$	ако x е по-голямо или равно на y стойността е true
$!x$	ако x е false , връща стойност true
$x\&\&y$	ако x и y са едновременно true , връща стойност true
$x\ \ y$	ако x и y са едновременно false , връща стойност true

Булевите изрази обикновено са част от операторите за условно изпълнение if...else... Синтаксиса на условните оператори е следния :

if (условие)

 оператор1

 else

 оператор 2

Смисълът на този оператор е следния : Ако(if) е изпълнено условието, тогава се изпълнява оператор1, иначе(else) се изпълнява оператор 2.

Нека сега да направим примера от предишната страница така, че да се проверява дали рожденната дата не е преди 1900 година и след 2000 година:

```
<HTML>
```

```
<HEAD>
```

```
    <TITLE>javascript</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
<SCRIPT LANGUAGE="javascript">
```

```
<!--
```

```
    var year=prompt("Въведете година на раждане","1980")
```

```
    if (year<1900)
```

```
        document.write("На Вашата възраст не бих седял пред компютъра")
```

```
    else
```

```
        if (year>2000)
```

```
            document.write("Още нероден, а вече гений!!!")
```

```
    else {
```

```
        old=2010-year
```

```
        document.write("През 2010 година Вие ще бъдете на "+old+" години")
```

```
    }
```

```
    /-->
```

```
</SCRIPT>
```

```
</BODY>
```

```
</HTML>
```

От примера виждате, че условните оператори могат да се влагат един в друг. Първо проверяваме дали въведената година не е по-малка от 1900 и ако е така, изписваме съответното съобщение. Ако условието не отговаря правим втора проверка дали годината е по-голяма от 2000. Ако това условие отговаря на истината се изкарва друго съобщение. Ако не отговаря на истината се счита че годината е между 1900 и 2000 и се изкарва трето съобщение. Можете да видите [ТУК](#) как работи скрипта.

Обърнете внимание, че ако операторите които се изпълняват след условието са повече от един се заграждат с фигурни скоби `{ }`. Когато оператора е само един може и без скоби :

```
if (year<1900)
    document.write("На Вашата възраст не бих седял пред компютъра")
.....
if (year<1900) {
    document.write("На Вашата възраст не бих седял пред компютъра")
    a=10
    document.write(a)
}
```

В някои случаи операторът `else` може да бъде пропуснат. Тогава скрипта ще се изпълнисамо ако е вярно поставеното условие, иначе няма да се изпълни.

```
var x=prompt("въведете положително или отрицателно число")
if (x<0) {
    x=-x
}
document.write(x)
```

Този скрипт ви подканва да въведете положително или отрицателно число, след което прави проверка дали числото е отрицателно (`x<0`) и ако е така го преобразува в положително (`x=-x` обръща знака на променливата `x`) и накрая го показва на екрана. Вижте [ТУК](#) как работи скрипта.

Оператори за цикъл

В езика javascript често се налага да се повтаря една и съща операция няколко пъти. За тази цел се използва оператор за цикъл. Първия оператор за цикъл който ще разгледаме е `for`. Синтаксиса му е следния :

```
for (i=0; условие; i++) {
```

```
}
```

...където *i* е променлива, която се използва за брояч, а условие е булев израз. Нека да направим скрипт, който да извежда на екрана числата от 0 до 9 :

```
for (i=0;i<10; i++) {  
document.write(i)  
document.write("<BR>")  
}
```

За удобство от сега нататък вече няма да пиша HTML частта от документа.

Оператора `for` е оператор за цикъл, който се управлява от една променлива, наречена брояч. В случая това е променливата *i*, но можете да и дадете каквото искате име. Можем да разделим мислено оператора `for` на две части. Първата част следва веднага след `for` и е заградена с кръгли скоби. Втората част е тялото, т.е. там са разположени операторите които ще се изпълняват по време на цикъла.

В първата част (да я наречем "условна" защото там се поставя условието) се извърша броенето и проверката дали дадено условие е изпълнено. Нека да я разгледаме по-подробно.

```
for (i=0;i<10; i++)
```

Първо създаваме една променлива *i* като и даваме начална стойност 0 (*i=0*). След това правим проверка дали стойността на променливата *i* е по-малка от 10 (*i<10*). Ако това условие е вярно, продължаваме нататък, ако не е, цикълът спира. Третата стъпка в условната част е увеличаването на *i* с 1 (*i++*). Ако цикълът не е спрял поради изпълняване на условието, той се повтаря отново докато не се изпълни, но без да се нулира променливата отначало. Така при първото завъртане на цикъла *i* ще бъде равно на 0, при второто на 1 и т.н. докато не стане 9. Когато стане 9 цикълът ще се прекъсне.

Тялото на цикъла се изпълнява толкова пъти, колкото се завърта самия цикъл. В този случай в тялото имаме оператор за отпечатването на променливата *i* на екрана и преминаването на нов ред :

```
{  
document.write(i)  
document.write("<BR>")  
}
```

Така изпълнявайки този цикъл ще видите на екрана изписани цифрите от 0 до 9 . Вижте [ТУК](#) как се изпълнява скрипта. Ако искате да се изписва и числото 10, трябва да промените условието, например вместо `i<10` да напишете `i<=10`. Трябва да знаете че всички оператори вътре в условната част трябва да се отделят един от друг с точка и запетая (;).

С помощта на цикъла `for` можете да правите доста сложни скриптове, но това ще го разгледаме по-подробно в готовите скриптове.

Другия тип цикъл в javascript е :

```
while (условие) {  
    оператори  
}
```

Този цикъл е малко по-опростен. При него стойностите на всички променливи се задават предварително, а самия цикъл се изпълнява докато условието в кръглите скоби отговаря на истината. Нека направим примера с отпечатването на числата от 0 до 9 с помощта на `while` :

```
i=0  
while (i<10) {  
    document.write(i)  
    document.write("<br>")  
    i++  
}
```

Този скрипт работи по същия начин като предишния, само че с друг оператор за цъкъл. Първо приавояваме начална стойност 0 на променливата `i` . След това поставяме условие докато `i<10` да се изпълнява частта от скрипта във фигурните скоби. Там се увеличава и стойността на `i` с единица.

Функции

Функциите в javascript са блокове от код, които първо се дефинират, а след това се изпълняват. Структурата на функциите е следната :

```
function name() {  
    оператори  
}
```

..където `name` е името на функцията, последвано от отваряща и затваряща кръгли скоби, които може да съдържат някакви аргументи, но може и да са празни, и най-накрая тялото на функцията заградено с фигурни скоби.

Това е начина да се дефинира функцията и това става обикновено в главата (HEAD) на HTML документа. Самото извикване на функцията става в тялото (BODY) на HTML документа чрез изписване на името и, както и аргументите, ако има такива. Вижте един пример за функция без аргументи :

```
<HTML
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
    function firstName() {
        name=prompt("Как се казвате ?")
        alert("Здравейте"+name)
    }
</SCRIPT>
<BODY>
<SCRIPT LANGUAGE="Javascript">
    firstName()
</SCRIPT>
</BODY>
</HTML>
```

В главата на документа се дефинира функция с име `firstName` без аргументи. Самата функция изпълнява следното: на променливата `name` се присвоява стойността, която ще напишете в подканващия прозорец "Как се казвате ?". След това в `alert` прозорец ще се изпише съобщението "Здравейте " + името което сте въвели. Веднъж дефинирана функцията може да се извика навсякъде в тялото на HTML документа само чрез името и. Ето [ТУК](#) можете да видите как работи функцията.

Сега нека направим една функция с аргументи.

```
<HTML
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
    function fontSize(num) {
        document.write("<font size="+num+">")
        document.write(num)
        document.write("</font>")
    }
</SCRIPT>
</HEAD>
<BODY>
    fontSize(12)
</BODY>
</HTML>
```

```

}
</SCRIPT>
<BODY>
<SCRIPT LANGUAGE="Javascript">
    for(num=1;num<=7;num++) {
        fontSize(num)
    }
</SCRIPT>
</BODY>
</HTML>

```

Тук имаме функция `fontSize` с един аргумент с име `num`. Действието на функцията е следното. Изписва HTML тага ``. На следващия ред се изписва самата променлива `num`. И най-накрая се изкарва затварящия таг ``. Не забравяйте че когато пишете HTML тагове с `document.write` те не се изписват на екрана, а се изпълняват като истински HTML тагове. Самата функция сама по себе си е безполезна, ако не дадем някакви стойности на променливата `num`. Това го правим с друг скрипт в тялото на документа. В него използваме цикъл с който задаваме стойността на `num` на единица. След това се проверява дали `num` не е по малко или равно на 7, и ако е така стойността се увеличава с едно. В тялото на цикъла се извиква функцията `fontSize`. Нека проследим стъпка по стъпка изпълнението на функцията :

Първо на `num` се задава стойност 1. След това се извиква функцията с параметър 1 (`fontSize(num)`). Самата функция изпълнява HTML тага : ` 1 `. Така на екрана се изписва цифрата 1 (стойността на `num` в момента на изпълнението на функцията) с големина на шрифта 1 (`font size=1`).

При следващото завъртане на цикъла стойността на `num` ще бъде вече 2 (`num++`) и функцията ще изпише цифрата 2 с големина на шрифта 2.

.....и така до `num=7` където цифрата седем ще се изпише с най-големия размер на шрифта - 7. Вижте [ТУК](#) как работи този скрипт.

Можете да създавате функции с повече от един аргумент. В такъв случай аргументите се разделят със запетай :

```
function argument(arg1, arg2, arg3)
```

Когато създавате такава функция, трябва винаги когато я извиквате да задавате същия брой аргументи, както когато сте я дефинирали.

Правилата за създаване на имена на функциите са същите като при променливите. Хубаво е ако името на функцията съдържа повече от една дума, първата дума да се изписва с малки букви, а всички останали да започват с главна буква, например `fontSize`.

Масиви

Масивите са тип променливи, които съдържат множество елементи. Ето структурата на един масив :

```
masiv=new Array()
```

..където `masiv` и името на масива, а `new Array()` е начина на създаването му. По този начин създадохме празен масив с неопределен брой елементи. Вместо празни скоби, можем да зададем броя на елементите в масива :

```
masiv=new Array(10)
```

Броя на елементите винаги трябва да бъде цяло неотрицателно число. В горния случай създадохме масив с име `masiv` и 10 елемента със стойност `NULL`, т.е. празни. Ако не зададем предварително броя на елементите в масива, то това можем да направим по-късно, като задаваме стойност на всеки елемент поотделно. Самото индексване на елементите става с квадратни скоби `[]`, като броенето започва от нула. Така първия елемент в масив с десет елемента е `masiv[0]` а последния `masiv[9]`. Нека да създадем масив със седем елемента и да им дадем стойности дните от седмицата :

```
masiv=new Array(7)
```

```
masiv[0]="понеделник"
```

```
masiv[1]="вторник"
```

```
masiv[2]="сряда"
```

```
masiv[3]="четвъртък"
```

```
masiv[4]="петък"
```

```
masiv[5]="събота"
```

```
masiv[6]="неделя"
```

...същото нещо можем да направим по по-лесен начин:

```
masiv=new
```

```
Array("понеделник","вторник","сряда","четвъртък","петък","събота","неделя")
```

така вече всеки елемент от масива си има някаква стойност. Тези стойности могат да бъдат променяни свободно по всяко време, както и типа на променливата. Например ако напишем `masiv[5]=122` шестия елемент ще бъде от целочислен тип, докато останалите ще са от стрингов тип.

Много лесен начин да си осигурите достъп до елементите на масива е, чрез използване на цикъл :

```
masiv=new
Array("понеделник","вторник","сряда","червъртък","петък","събота","неделя")
for(i=0;i<7;i++) {
    document.write(masiv[i] + "<br>")
}
```

Този скрипт ще покаже на екрана всеки един елемент от масива, като след всеки ще минава на нов ред. Вижте [ТУК](#) как работи примера.

Обекти

JavaScript е обектноориентиран език. Обектите в javascript се описват заедно със свойствата и методите им. Методите на обектите се отнасят до начина на изпълнението на самия обект. Нека да вземем за пример обект от реалния свят, какъвто е прозореца. Методите на обекта прозорец ще бъдат *отваряне* и *затваряне*, а свойствата му например *ширина* и *височина*. В javascript нещата не са по-различни. Ето например ние досега използвахме един обект заедно със метода му в почти всички примери дотук. Това е обекта `document` заедно със метода му `write`. И докато в разговорния език можем да обясним обекта със свойствата и методите му като кажем "широкия прозорец се отваря", в javascript за тази цел се използва разделител точка "." (`document.write` , `image.border` и т.н.)

Сега ще разгледаме по-подробно някои обекти в javascript за да разберем за какво става дума. Може би най-често срещания обект в javascript и прозореца на браузъра. Той носи името `window`. С помощта на javascript можете да го отваряте, затваряте, както и да правите други работи с него. Вижте например най-долната лента в прозореца на браузъра. Тя се нарича статус-бар (status bar). В нея се описва моментното състояние на браузъра, като скорост на връзката (при Netscape) URL на който се намирате в момента и т.н. Вие можете да промените информацията в статус-бара с помощта на свойството `status` на обекта `window`. Вижте как става това :

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
window.status="statusbar"
</SCRIPT>
<BODY>
```


</BODY

</HTML>

Редът `window.status` променя стойността на статусбара. Така например можете да направите надписа в статус бара да бъде заглавието на страницата ви. Вижте [ТУК](#) как работи този скрипт.

За разлика от свойствата на обектите, методите описват действията които могат да се извършват с тях. Методи са `open`, `write` и т.н. Ако сравняваме javascript с разговорния език, обектите са съществителните, свойствата на обектите са прилагателни, а методите на обектите са глаголи (те в действителност са си глаголи, `open`, `close`, `write`, `click`, `sort`, `concat` и т.н.)

Да разгледаме метода `open` на обекта `window`. Този метод отваря нов прозорец. Синтаксиса му е `window.open()`, като в скобите се слага URL на адреса който трябва да се отвори с новия прозорец. В следващия пример ще сложим на екрана картинка и при посочване с мишката върху нея ще се отваря нов прозорец с адрес `www4u.search.bg` :

<HTML>

<BODY>

</BODY>

</HTML>

В този случай при настъпването на събитието `onmouseover` се отваря нов прозорец (виж урок [събития](#)). Вижте [ТУК](#) как работи скрипта.

Ето и някои от най-често срещаните обекти заедно с методите и свойствата им :

обект	свойства	методи
document	bgColor image location title	write writeln open
image	border height width src	
window	location history frames name	close open prompt scroll

Събития

С помощта на javascript можете да следите какво става на страницата ви. Например посочването с мишката върху някаква картинка е събитие. Кликването върху картинката е друго събитие и т.н. Javascript обработва събитията с така наречените манипулатори на събития. Така при кликане ще се извика манипулатора onClick, при посочване onMouseover и т.н. Структурата на манипулаторите на събития е следния :

onСъбитие=(оператори)

Където *Събитие* е името на събитието, а в скобите са операторите които ще се изпълнят при възникване на събитието. Самия манипулатор на събитието може да бъде извън таговете `<SCRIPT></SCRIPT>`. Можете да го сложите например в така `<A>`. Вижте един пример, при който създаваме хипервръзка и като посочите с мишката върху нея се показва alert прозорец с обяснение за връзката:

```
<A href="http://www4u.search.bg" onMouseOver="alert('HTML и JavaScript  
уроци!');return true;">WWW4U</A>
```

Посочете сега с мишката върху връзката по надолу и ще видите резултата.

[WWW4U](#)

Ако смените манипулатора `OnMouseover` със `OnClick`, `alert` прозореца ще се показва при кликане вместо при преминаване с мишката върху хипервръзката. Много добър пример със събитието `onMouseover` има в урока за [rollover върху Image Map](#). Вижте [го!](#) Друго събитие е `onMouseout`. То настъпва когато курсорът на мишката вече не посочва върху обекта. С помощта на събитията `onMouseover` и `onMouseout` се правят така наречените rollover ефекти с картинки при които след посочване с мишката една картинка се заменя с друга. Но затова по нататък..

Освен събития свързани с движението на мишката, javascript обработва и събития свързани с отваряне и затваряне на прозорци, както и такива свързани с формуляри. Едно от събитията свързани с прозорците на брауъра е `onLoad`. То се извиква когато се зареди страницата. Ето например ако направите следната страница :

```
<HTML>  
<BODY onLoad="alert('Здравейте!')">  
</BODY>  
</HTML>
```

веднага след зареждането на страницата ще се появи alert прозорец с надпис "Здравейте!". Ако замените събитието `onLoad` със `onUnload`, което се извиква при напускане на страницата, "Здравейте!" ще се показва при

затваряне на страницата. Нека променим горния пример така, че при отваряне на страницата да се показва "Здравейте!" а при затваряне "Довиждане!" :

```
<HTML>  
<BODY onLoad="alert('Здравейте!)" onUnload=alert('Довиждане!)">  
</BODY>  
</HTML>
```

Ето [ТУК](#) можете да видите как работи примера. След това затворете прозореца на брауъра и ще видите съобщение "Довиждане!". Сигурно сте забелязали някои доста нагли сайтове, които отварят нов прозорец със друг сайт веднага щом се опитате да излезете от предишния. Те използват именно събитието `onUnload`. Друго събитие е `onAbort`. То настъпва когато прекъснете зареждането за страницата с бутона Stop на брауъра.

Ето и някои други събития и обяснението за действията им :

`onError` - възниква при грешка в скрипта

`onSelect` - възниква при избиране (селектиране) на текст

`onSubmit` - възниква когато изпратите формуляр за обработка

`onBlur` - възниква при напускане на обект

`onFocus` - възниква когато обекта е на фокус

`onChange` - възниква когато се промени съдържанието на обекта (например формуляр)