Тема 12

Интернет програмиране

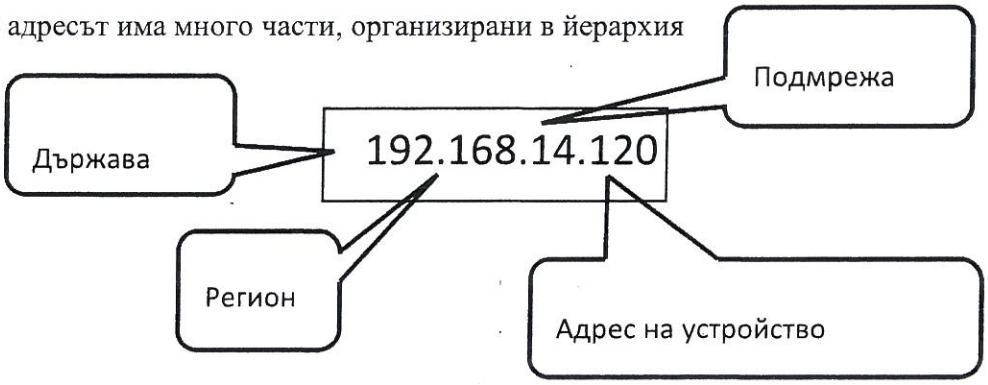
1. Обяснява и диференцира различните протоколи.

Мрежов протокол - Набор от правила и стандарти, които позволяват комуникация между мрежовите устройства. Мрежовите протоколи включват механизми за идентифициране на устройствата и осъществяване на връзка помежду си.

IP(Internet Protocol)- Един от най-важните протоколи, използвани в интернет комуникацията, е Интернет протоколът (IP).

Всички устройства в Интернет имат адреси, тези адреси се наричат IP адреси. IP адресът е уникален за всеки компютър или устройство в края на мрежата.

IP адресът има много части, организирани в йерархия



Тази версия на IP адресиране се нарича IPv4

IPv4 е последователност от четири, трицифрени числа, разделени от точка

Всяко число може да бъде число от нула до 255. IPv4 не е достатъчен за всички мрежови устройства, свързани към интернет. През 1995 г. е създадена нова версия на интернет протокола, наречена IPv6.

IPV6 използва 128 бита. Тези 128 бита са организирани в осем 16-битови части. Всеки 16-битов блок се преобразува в шестнадесетичен и се разделя с двоеточие.

Пример на пълен IPv6 адрес:

3FFE:F200:0234:AB00:0123:4567:8901:ABCD

TCP протокол (Transmission Control Protocol) - осигурява зависим от връзката трафик между две устройства в мрежата. Когато се изпраща съобщение, между компютъра, който изпраща и компютъра получател трябва да се изгради сесия (връзка). Съобщението, което се изпраща се разделя на сегменти, като всеки сегмент има пореден номер. При достигането на сегментите при получателя, те се подреждат по тези номера, за да се получи оригиналното съобщение. Ако някой от сегментите не е

пристигнал се изисква неговото повторно изпращане. *Протоколът TCP осигурява гарантирано достигане на информацията до получателя.*

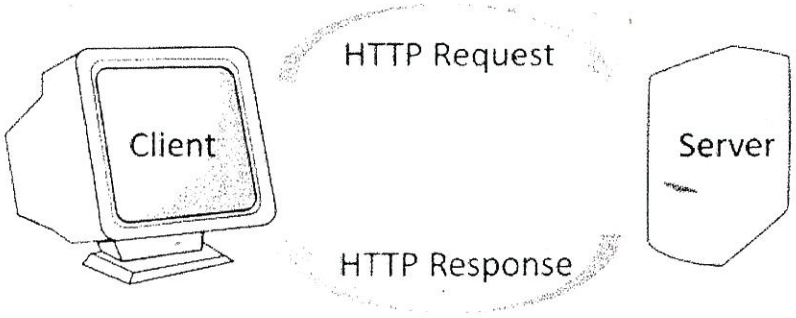
UDP (User Datagram Protocol) протокол - Протоколът UDP осигурява негарантирана доставка на данни.UDP осъществява изпращането на данните и не се занимава с проверка на получаването им. Съобщението, което се предава се нарича дейтаграма (datagram).

Програмите, използващи UDP протокола трябва да реализират самостоятелно:

- препредаване на загубени дейтаграми;
- игнориране на повторения;
- фрагментиране и обединяване на големи потоци данни.

2. Дефинира понятието HTTP заявка, прави изводи за различните HTTP методи и избира метод за конкретна ситуация.

HTTP (**Hyper Text Transfer Protocol**) – мрежов протокол за пренос на хипертекст. В HTTP протокола се използват понятия като клиент (обикновено това са Web-браузърите (или web навигаторите) – т.е. самите приложения и сървър (това са уеб сървърите – т.е. самите приложения).



HTTP дефинира методи за посочване на желаното действие, което трябва да се извърши върху идентифицирания ресурс.

Метод	**Описание**
GET	Извличане / зареждане на ресурс
POST	Създаване / съхраняване на ресурс
PUT	Актуализиране на ресурс
DELETE	Премахване на ресурс

HTTP съобщение за заявка

Съобщение за заявка, изпратено от клиент, се състои от:

- HTTP линия за заявка
  - Метод на заявка (GET / POST / PUT / DELETE / …)
  - URI (URL)
  - Версия на протокола
- HTTP хедъри
  - Допълнителни параметри
- Тяло на HTTP заявка - незадължителни данни, напр. публикувани формулярни полета

```
<method> <resource> HTTP/<version>
<headers>
(empty line)
<body>
```

Get метод на заявка:

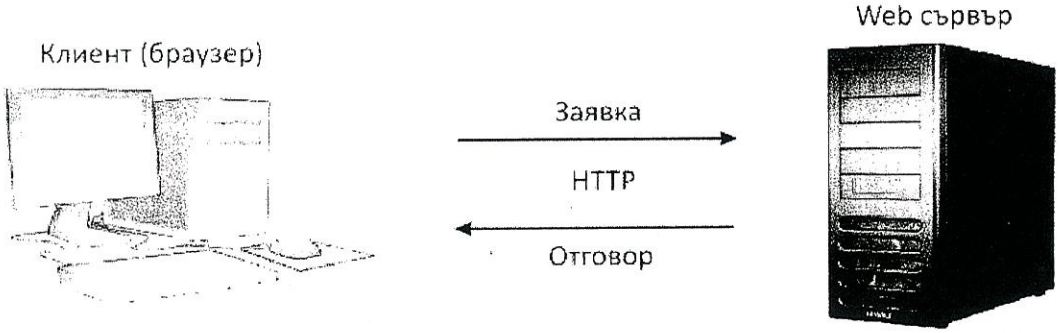
```
<form method="get">
   Name: <input type="text" name="name" />
   Age: <input type="text" name="age" />
   <input type="submit" />
</form>
```

HTTP съобщение за отговор - Съобщението за отговор, изпратено от HTTP сървъра, се състои от:

- HTTP линия на състоянието на отговора представя:
    - Версия на протокола
    - Код на състоянието
    - Текст на състоянието
- Хедърите:
    - Предоставят метаданни за върнатия ресурс
- Тяло на отговора е:
    - Съдържанието на HTTP отговора (данни)

3. Обяснява и представя графично клиент-сървърната комуникация.

Клиент/сървър архитектурата не винаги се ограничава до комуникацията с единствен сървър. Понякога клиентските заявки се разпределят между много сървъри. Но в най-честия случай края на клиент/сървър веригата е сървърът с базите данни, а клиентското приложение се грижи за логиката и графичния интерфейс.



Комуникацията между Web клиента и Web сървъра се осъществява чрез използване на протокола HTTP. **HTTP** (**H**yper **T**ext **T**ransfer **P**rotocol) служи за обмен на документи между сървър и клиент, и е част от протоколния стек TCP/IP за управление на поток от данни в Интернет. Всъщност протокола HTTP функционира на базата на проста схема от тип "въпрос–отговор". Клиентът изпраща заявка към сървъра, на която сървърът отговаря. Архитектурата клиент/сървър има три основни компонента: клиент, сървър и връзката помежду им.

4. **Различава смисъла на употребата и необходимостта от HTML, CSS и JavaScript**.

HTML – **H**yper**T**ext **M**arkup **L**anguage - Нотация за описание, изгражда структура на документ и представя начин за форматиране (презентация). Езика е изграден на базата на тагове. Таговете предоставят метаинформация за съдържанието на страницата и определят нейната структура. Един HTML документ се състои от много тагове, които се влагат. Тагове са най-малкият елемент в HTML. Елементите са комбинация от отварящ, затварящ таг и атрибути. Чрез атрибути определяме свойсвтата на таговете - размер, цвят и т.н

CSS определя стила на HTML елементите като шрифтове, цветове, полета, размери, позициониране и др.

CSS се декларира в следния формат: **свойсвто:стойност** Вграденият CSS дефинира правила за форматиране на определен HTML елемент:

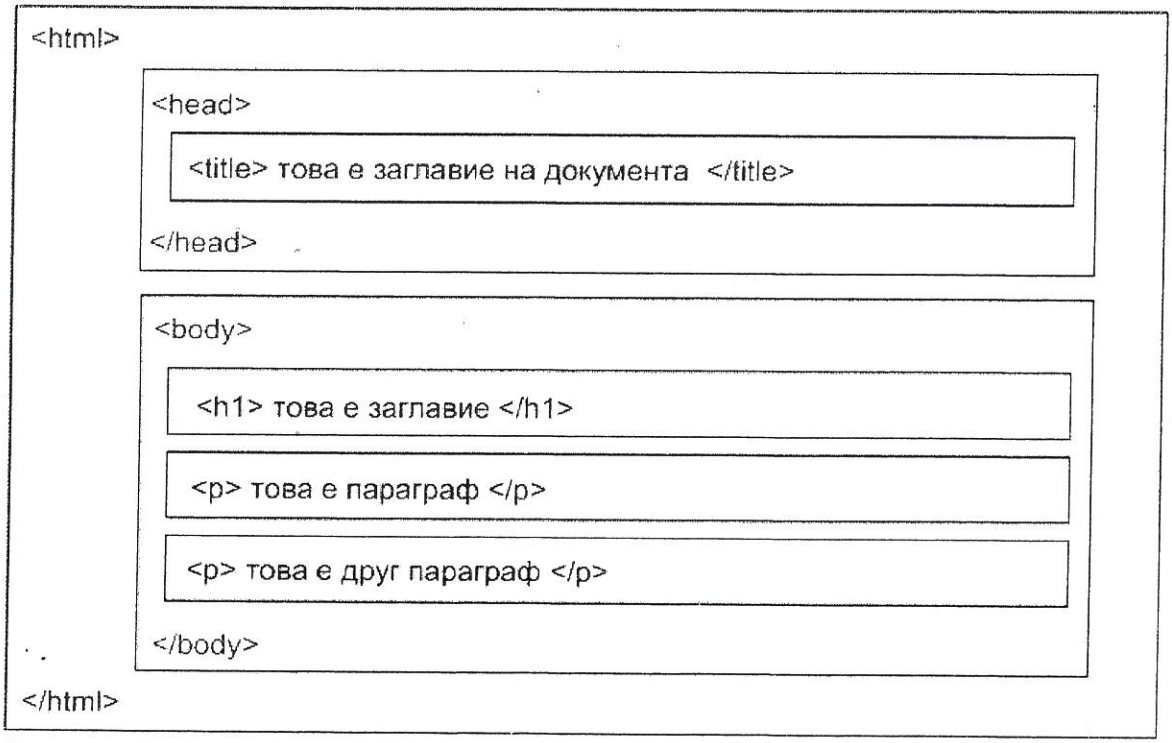
** I am a RED text paragraph **

HTML описва текст с форматиране, изображения, таблици, формуляри и т.н.
CSS добавя стилизиране към HTML документите. Уеб сайтовете се състоят от HTML + CSS + изображения като може да съдържа и JavaScript код.

Наред с **HTML** и **CSS**, **JavaScript** е една от **3-те основни технологии** в уеб света. JavaScript позволява динамичност и интерактивност в уеб страниците. Има достъп до **DOM** и API(известия, геолокация и т.н.)

5.Дефинира и използва коректно HTML тагове.

Основната структура на HTML документа включва три задължителни елемента <html>, <head>и <body>, както е показано на фигурата. Документът започва с деклариране на типа на документа . Този елемент указва на браузъра какъв набор от стандарти е използван в документа. Чрез <!DOCTYPE html> (няма затварящ таг) може да се валидира софтуера, като предварително се зададе версията на използвания HTML код. След него се намира отварящият таг <html>, който формира целия HTML документ. Затварящият таг </html> е последното нещо, което съдържа всеки HTML документ. Елементът може да съдържа други елементи, например заглавна част на документа, формирана чрез двойката тагове <head> и </head>.



В нея се разполага информация за документа (метаданни), която не се визуализира в браузъра. Там се задава и заглавието на самия документ, посредством двойката тагове <title> и </title>. Елементът <body> формира цялото визуално съдържание на HTML документа.

77

5

Някои от таговете са за:

Заглавия:

Html има шест различни HTML заглавия

<h1> определя най-важното заглавие.

<h6> определя най-малко важното заглавие.

Параграфи

Тагът **<p>** дефинира параграф

Тагът **
** дефинира нов ред

Булети и Номерирани Списъци

Тагът <ul> - за булет списък

Тагът <ol> - за номериран списък

Пример:

```
<ul>
  <li>First item</li>
  <li>Second
  item</li> <li>Third
  item</li>
</ul>
```

Хипервръзки:

Създават се с тага <a>

```
<a></a>
```

Адресът се посочва в **href=""** атрибута

```
href="https://abv.bg"
```

Външна хипервръзка

```
<a href="https://abv.bg">Препратка към abv поща.</a>
```

Локални хипервръзки

```
<a href="welcome.html">Показва "welcome.html"</a>
```

Снимки, изображения

Изображенията са външни файлове, които се вмъкнати чрез <img> тага.

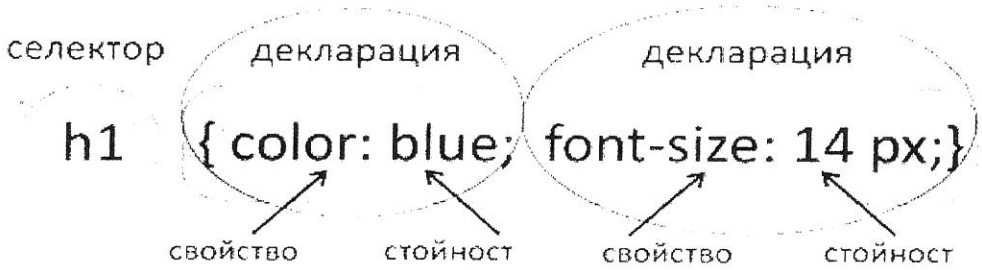
```
<img
  src="images/SoftUni-logo.png"
  alt="SoftUni logo (blue)"
  width="400"
  height="313" />
```

Таблици – дефинира се с тага <table>, в който се посочват редове <tr> и клетки<td>

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
</table
```

6. Задава свойства на HTML компонентите чрез CSS.

Всяко CSS правило има две основни части: селектор и една или повече декларации. Ако трябва да сме още по-прецизни, всяка декларация се състои от комбинацията на свойство и стойност. Синтаксисът на правилото е следният:



Можем да вмъкнем по три различни начина CSS кода в HTML документа:

• от външен файл, който представлява самостоятелен CSS файл, и за който е необходимо обръщение от HTML документа;

• деклариране на кода вътре в документа – блок със стилове в документа, затворен от таговете и поставен в заглавната част на HTML документа ();

• вграден стил на даден елемент – чрез деклариране на CSS кода в реда. За да се използва външен CSS файл, е необходимо в HTML документа да се добави обръщение към конкретния CSS файл в частта , като се използва следният синтаксис:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

Едни от най-често прилаганите свойства за изграждане на стилове са:

- **color**: определя цвета на буквите
- **font-family**: трябва да съдържа няколко шрифта. Ако браузърът не поддържа първия, той ще опита следващия
- **font-size**: задава размера
- Блокови елементи (** ;<h1>;<p>**):
  - Винаги започвайте на нов ред
  - Заемат цялата налична ширина
- ** ** елемента:
  - често се използва като контейнер за други HTML елементи
- Вградени елементи (**;<a>;<img>**):
  - Не започват на нов ред
  - Заемат само толкова ширина, колкото е необходима
- **** елемент:
  - Често е използван за контейнер за текст
- **border**: определя типа, дебелината, цвета
- **border-radius**: закръгля граничните краища
- **background**: задава фона

Външни Отстояния

- Използва се за генериране на пространство около елементи
- **margin** свойсвото задава размера на празното пространство извън границата

Вътршени Отстояния

- Използва се за генериране на пространство около съдържанието
- Свойството **padding** задава размера на празното пространство вътре в границата

CSS Селектори

- **.class** – избира група елементи с посочения клас
- **#id** – избира уникален елемент
- **tag** – избира всички посочени тагове
- * - избира всичко

Пример:

```
<!DOCTYPE html>
<html>
 <head>
  <link rel="stylesheet" type="text/css"
 href="styles.css"> </head>
 <body id="content">
 <p>This is a <span class="special"> special beer</span> for <span
class= "special">special drinkers</span>.</p>
 </body>
</html>
```

style.css

```
#content {
 background: #EEE;
}
p {
 font-size: 24pt;
}
.special {
 font-style: italic;
 font-weight: bold;
 color: blue;
}
```

7. Описва и демонстрира употребата на семантични елементи за създаване на семантична страница.

Семантичният уеб е обединяващо звено на различните стилове, като HTML Markup, CSS правила, JavaScript код, снимки, аудио, видео и други ресурси, като ги съчетава в един общ формат.

Семантични елементи

- Елемент <header> – представлява заглавие на елемент или група от елементи. Тага <header> позволява използването на няколко заглавия в един документ за разлика от възможността която предоставя HTML4.
- Елемент <nav> – дефинира навигацията на сайта. Той е предназначен за големи блокове от навигационни връзки. Не всички връзки трябва да са в <nav> елемента, а само главното навигационно меню.

- Елемент <article> – представлява предмет/статия. Съдържанието в <article> трябва да има смисъл само по себе си и да може да се разпространява независимо и отделно от останалата част на сайта. Използва се най-често за публикация във форум; публикация в блог; новини; коментари и др.
- Елемент </section/> е тематично групирано съдържание, обикновено със заглавие. Този елемент позволява влагане на нова сегмантична структура във вече съществуващ елемент.
- Елемент <aside> – той представя съдържание което е странично от основното съдържание на страницата. Съдържанието в <aside> се счита за неважно и може да бъде пропуснато, ако сайта се възпроизвежда на устройство с малък екран като телефон.
- Елемент <footer> – може да съдържа информация за автора на страницата; навигация на сайта; авторските права върху използваните материали и връзки към други (подобни) публикации.

8. **Обяснява и демонстрира начините за създаване на адаптивен (responsive) дизайн**.

Отзивчивия уеб дизайн (Responsive Web Design - RWD) представлява стилизирана уеб страницата, с цел да улесни взаимодействието с потребителите и промени нейната визуализация към компютър, таблет, мобилен телефон и др. устройства. Този вид уеб дизайн се адаптира спрямо различните резолюции на екрана. По този начин се премахва нуждата от мащабиране на екрана.

Когато към описанието на елементите в CSS се въведат допълнителни свойства и правила за представяне, блоковете в уеб страницата заемат последователна позиция и изпълват видимото пространство на браузъра. Това се реализира с:

1) въвеждане на мета тага viewport;

2) структура на решетката - фиксирана ширина на блоковете в проценти при различните ширини на екрана;

3) медийни заявки -**@media** за различни ширини на екрана.

Чрез мета тага **viewport** дизайнерите имат възможност да контролират прозореца на представяне на сайта. Метода се въвежда с HTML 5.

<meta name="viewport" content="width=device-width, initial-scale=1.0“>

1) Тагът <meta> с име viewport дава инструкции на браузъра да контролира размерите на уеб страницата и мащабирането.

2) Характеристиката width=device-width, определя ширината на страницата да следва ширината на екрана на устройството.

3) Характеристиката initial-scale=1.0, определя първоначалното мащабиране на страницата когато тя се зарежда в браузъра.

9. Увод в JavaScript – работа с обекти и събития.

JavaScript е обектно-базиран скриптов език от страна на клиента, който се използва, за да се направят по-динамични Web страници. Обектно-базиран означава, че могат да се използват елементи като обекти; от страна на клиента означава, че JavaScript се изпълнява от софтуера, който използва посетителя, а не Web сървъра на сайта, обслужващ тази страница; скриптов език означава, че не изисква компилиране преди изпълнение.

Поставяне на JavaScript в HTML файл.

Използва се тагът <SCRIPT> (</SCRIPT>) със следните атрибути:

- language – определя скриптовия език;

Пример:

```
<SCRIPT language="JavaScript">
<!--
document.write ("Първи ред, написан с JavaScript");
//-->
</SCRIPT>
```

- src – извикване на външни скриптове (текстови файлове, съдържащи само JavaScript код и с разширение .js).

Пример:

```
<SCRIPT language="JavaScript"  src="my script.js">
</SCRIPT>
```

Работа с обекти.

Обектът е начин на моделиране на нещо реално, въпреки че самият обект е абстрактна величина. Когато мислим за един обект, ние си представяме нещо общо, без да се интересуваме от конкретният му вид – напр. кола.

Но колите притежават и други характеристики, които наричаме свойства – цвят, големина, брой врати и т. н. На свойствата може да се дават стойности., според типа. Достъпът до свойствата на един обект се извършва посредством оператора точка "." (обект.свойство=стойност).

Методите представляват функции, които извършват различни операции със свойствата на обектите. Всеки метод е част от даден обект. Достъпът до метод на даден обект се извършва чрез точка. (обект.метод).

Има предварително зададени обекти в JavaScript, а има и възможност за създаване на собствени обекти.

Пример за предварително зададени обекти в JavaScript е обектът window.

Това е обект, който се създава за всеки прозорец, който се показва на екрана. Обекти window се явяват главният прозорец, наборът от фреймове или отделен фрейм, както и всеки нов, създаден чрез JavaScript прозорец.

Свойства на обекта са:

- closed – Свойството се използва, за да се провери дали потребителят е затворил даден прозорец.
- name – Съдържа името на текущия прозорец и позволява да дадем име на прозорец. Задаването на име на прозорец става в HEAD частта.
- defaultStatus – съдържа текста, който да се извежда в лентата на състоянието, когато не е указано нищо.

80

Пример: <BODY onLoad="window.defaultStatus='Welcome!';">
- location – Свойството съдържа текущият URL адрес на прозореца, като той може да бъде променян в движение, за да се пренасочи потребителя към нова страница, подобно на връзка.
Методи на обекта window са:
- alert() – Извежда прозорец с предупреждение, а потребителят трябва да щракне върху бутона ОК.
- confirm() – Извежда прозорец за потвърждение/отказ.
- print() – Методът извиква диалоговия прозорец Print, чрез който потребителят може да зададе желаните настройки и да отпечата документа.
- prompt() – Методът извиква прозорец за въвеждане на данни, резултатът се присвоява на променлива.
Пример: var pr=window.prompt("Въведете година:", "2010")
- open() – Позволява да се отвори нов прозорец чрез JavaScript.

Използване на функции за обработка на събития.

Функцията за обработка на събитие се добавя като допълнителен атрибут към HTML тага, като стойността на атрибута може да бъде JavaScript код, заграден с кавички.

Пример:

```
<HTML>
<BODY>
<FORM>
<INPUT type="button"  onClick="window.alert ('здравей!'); ">
</FORM>
</BODY>
</HTML>
```

1. **събитието Click (onClick)**

Събитието Click настъпва, когато потребителят щракне върху определени области от Web страницата. Тези области се намират в елементи от форми и връзки.

2. **събитието Mouseover (onMouseOver)**

Събитието Mouseover настъпва, когато потребителят премести показалеца на мишката над хипервръзка. Ако в примера вместо onClick използваме onMouseOver, предупреждението се появява при посочване на хипервръзката.

3. **събитието Mouseout (onMouseOut)**

Това събитие настъпва, когато потребителят премести показалеца на мишката извън хипервръзка.

4. **събитието Focus (onFocus)**

Събитието Focus настъпва, когато потребителят предаде фокуса на прозорец или елемент от формуляр в Web страница. Потребителят дава фокуса като щракне върху елемента. Например, когато потребителят щракне в текстово поле (преди да въведе нещо), той предава фокуса на текстовото поле. А ако щракне в неактивен прозорец, той го прави активен и му предава фокуса. Тази функция може да се използва в елемент от формуляр или в отварящия таг (при предаване на фокуса към прозореца).
Има и много други събития като събитието Submit (onSubmit), Keydown (onKeyDown), Keyup (onKeyUp) и т.н.