

Типове категории на елементите - Block / Inline

Обикновено HTML
елементите са от
block или Inline тип.

block елементи

```
<div>Block елемент 1</div>
```

```
<div>Block елемент 2</div>
```

inline елементи

```
<span>inline елемент 1</span> <span>inline елемент 2</span>
```

block елемент, в който има inline елемент

```
<div> <span>Block елемент, в който има inline елемент</span> </div>
```

Block level елементи

Основните наща които трябва да знаем за Block елементите:

1. Главната цел на block - вия елемент е да очертае / създаде главната структура на документа;
2. По подразбиране всеки нов блоков елемент започва на нов ред;
3. Блоковите елементи се рендират по вертикала в потока на документа;
4. Блоковите елементи запълват цялото съдържание по дължина на родителският елемент;
5. Блоковите елементи могат да съдържат в себе си както block така и inline елементи;

Примерни block елементи са: `<div>`, `<h1>`, `<p>`, `<hr>` и други.

Inline (инлайн) елементи

Втория тип елементи са така наречените инлайн (на английски inline) елементи.

Има няколко важни неща, които трябва да запомните за тях:

1. Ширината им е равна на съдържанието, което ги изпълва.
2. Inline елементите се характеризират с това, че не променят структурата на уеб страницата, а следват нейното съдържание;
3. Inline елементите винаги започват и завършват на един и същи ред;
4. Inline елементите могат да съдържат в себе си само inline елементи;

Примери за инлайн елементи са: ``, `<a>`, ``, `` и други.

Inline елементи - 2

Инлайн елементите са предназначени за описване на същинското съдържание в документа, за стилизиране (ограничено) или за добавяне на друго съдържание.

“ Те са съдържанието на документа “

Ще категоризираме inline елементите в няколко неофициални групи:

1. Семантични тагове;
 - а. Тагове за важност на текста
 - б. Тагове които обикновено описват фрази или изречени
 - с. Тагове които съдържат единични символи
2. Таговете за визуална промяна на текст;
3. Тагове, които променят потока на документа;
4. Тагове-контроли или просто контроли.

Inline елементи - 3 Семантични

Тя е най-голямата от всички групи, защото в нея влизат повечето инлайн тагове. Това е нормално тъй, като това е тяхното главно предназначение, а именно да опишат възможно най-точно смисъла на текста, който маркират.

Тагове за важност на текста са:

- ``
- ``
- ``

Тези тагове са доста полезни за SEO оптимизация.

`` също както `<div>` тага има неутрално значение, затова е и с най-ниска важност от останалите елементи от своята група.

Тагове които обикновено описват фрази или изречени:

- `<a>`
- `<cite>`
- `<code>`
- `<kbd>`

Тагове които обикновено описват дум:

- `<acronym>`
- `<abbr>`
- `<dfn>`

Тагове които съдържат единични символи:

- `<sub>`
- `<sup>`

Inline елементи - 4 Визуална промяна

Значението на тези тагове е много просто. Те означават това, че съответния текст трябва да изглежда така. ``, `<i>`, `<u>`

Тук е момента да се споменем и за връзката и разликите между визуално съответстващите тагове на:

`<i>` ``
`` ``

Визуално тези тагове правят едно и също, но имат съвсем различна тежест в интернет търсачките и скрийн-рийдърите.

Важно е да се знае за тази група, че всички browsers стилизират тези тагове визуално по собствени начини (наклонен, удебелен или друг шрифт),

Inline елементи - 5 Променящи потока

Включва във себе си тагове, които променят потока на документа:

`
`, `<bdo>`

**`
`** го ползваме когато искаме даден елемент да слезе на следващият ред.

`<bdo>` Контролира посоката на текста, ползва се заедно с атрибута `dir`.

`<bdo dir="rtl">`

`rtl` - right to left (от дясно на ляво)

`ltr` - left to right (от ляво на дясно)

Inline елементи - 6 Тагове-контроли

Всички те служат за попълване на данни от потребителя в документа, под формата на свободен текст или възможност за избор между предварително избрани опции.

Всеки от тези тагове има собствени особености във синтаксиса си и могат да бъдат използвани само във форми <form>

<input />, <button>, <textarea>, <select>



block-level

inline

Структурни блокове - ul - ol

List item

<p>

List item

</p>

List item

<p>

List item

</p>

Структурни блокове - ol

**** тага е блоков елемент създаващ списък с един или повече артикули подредени в определен ред, които се създават с **** тага.

**** тага може да съдържа само прекия си наследник **** тага с който изброяваме в точно определен ред неговите наследници.

```
<ol>
  <li>
    List item
  </li>
  <li>
    <p>
      List item
    </p>
  </li>
</ol>
```

Структурни блокове - table

Доста остаряла практика е да се оформя главната структура на документа с таблици, което трябва да се избягва задължително по редица причини.

За логическото групиране на една или повече различни части от документа се използва **<div>**, който приспада в групата на мултифункционалните блокове, но също така е и структурен блок.

```
<table>
```

```
  <thead>
```

```
    <th>
```

```
      <td>
```

Клетка 1

```
    </td>
```

```
  </th>
```

```
</thead>
```

```
...
```

```
...
```

```
  <tbody>
```

```
    <tr>
```

```
      <td>
```

Клетка 1

```
    </td>
```

```
  </tr>
```

```
</tbody>
```

```
</table>
```

Терминални блокове

Това са елементи, които съдържат текст или inline елементи. Това са контейнери предназначени за същинското съдържание.

Не може да съдържат други блокови елементи. Всички тагове от тази категория се използват за маркирането на inline съдържание.

За разлика от **Структурните** блокове, **Терминалните** блокове и **inline** елементите се различават само по семантична си стойност.

<h1>...<h6>, <p>, <blockquote>, <dt>, <address>, <caption>

<p> тага е за маркиране на параграф от един или повече редове текст.

<blockquote> служи за маркиране на цитати, вътре в който може да използваме прекият му наследник, **<cite>**.

Многофункционални блокове - 3

Многофункционалните блокове могат да съдържат други блокове или съдържание, но не и двете заедно.

Това за което трябва да се внимава, когато се използват такива тагове е да се избягва наличието на “смесено съдържание” в тях.

Смесено съдържание означава в един блоков елемент да има поставени друг блоков елемент и инлайн елемент, като директни роднини.

`<div>`

`<div>`

Някакъв текст

`</div>`

`<div>`

Това най-вероятно няма да бъде разпознато, като грешка от повечето валидатори, но е много грешно. Като оставим логическото противоречие на страна, по-големият практически проблем идва от това, че бразъра не може да рендира block и inline съдържание едновременно, тъй като block падат на долу в документа, а inline съдържанието се изчертават напречно.

Семантично значение на елементит

Всеки таг в **HTML** има собствено семантично предназначение и трябва да се използва за маркирането на съответните за него неща.

Много е важно таговете да се използват за точно това, за което са предназначени, за да не се губи основната идея на целия език.

Въпреки, че не е задължително да се помнят всички налични тагове в езика е добре разработчика да се запознае с цялата документация, за да знае какво се предлага като набор от възможни опции от технологията, които да използва за максимално ефективна семантика в документите си. <http://www.w3schools.com/html/default.asp>

Семантично значение на елементит

Heading таговете (h1 до h6 / h7 няма :D) се използват основното заглавие във всеки **HTML** документ.

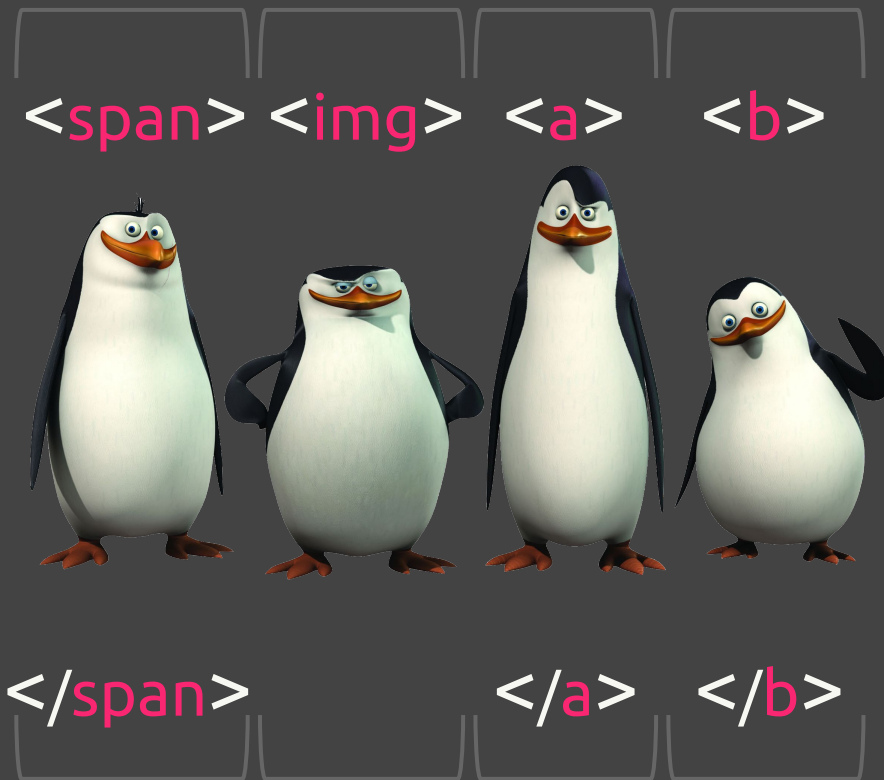
В h1 трябва да се поставя само съдържание, което се предполага, че е основното заглавие на документа.

Unordered List - се използва за списъци, които нямат строго определен ред в подредбата си.

```
<ul>  
  <li>  
    Някакъв текст  
  </li>  
</ul>
```



Block vs Inline



Коя е правилната структура?

```
<ul>  
  <li>  
    List item  
  </li>  
  <div>  
    <li>  
      List item  
    </li>  
  </div>  
</ul>
```

```
<ul>  
  <li>  
    List item  
  </li>  
  <li>  
    <div>  
      List item  
    </div>  
  </li>  
</ul>
```

Коя е правилната структура?

List item

<div>

List item

</div>



List item

<div>

List item

</div>



Кой е правилният код?

<h1>

Lorem Ipsum is simply *dummy* text
of the printing and typesetting industry. Lorem Ipsum has
been the industry's

</h1>

<h1>

Lorem Ipsum is simply *dummy* text
of the <div>printing</div> and typesetting industry. Lorem
Ipsum has been the industry's

</h1>

Кой е правилният код?

<h1>

Lorem Ipsum is simply dummy text
of the printing and typesetting industry. Lorem Ipsum has
been the industry's

</h1>



<h1>

Lorem Ipsum is simply dummy text
of the <div>printing</div> and typesetting industry. Lorem
Ipsum has been the industry's

</h1>



Основни HTML елементи

Преди да продължим със CSS ще разгледаме някои от основните HTML елементи:

- Headings (Заглавия) - `<h1>` до `<h6>`
- Paragraphs (Параграфи) - `<p>`
- Links (Линкове) - `<a>`
- Bold & Italic (удебелен и курсив) - ``, `<i>`
- Superscript и Subscript - `<sup>`, `<sub>`
- Line Breaks и Horizontal Rules - `
`, `<hr>`
- Strong & Emphasis - ``, ``
- Lists (Видовете Списъци) - ``, ``, `<dl>`

Основни HTML елементи

- Tables (Таблици) - `<table>`
- Forms (Форми) - `<form>`, `<select>`, `<textarea>`, Inputs (полета за въвеждане и видовете типове) `<input>`
- Comments (Коментари в HTML) - `<!-- -->`
- Attribute:
 - `ID`
 - `class`

Headings (Заглавия)

- Използваме ги за маркиране на заглавията в страницата.
- Основното заглавие трябва винаги да е най - горе.
Лоша практика е да се слага в края на страницата, както и <h6> тага да е преди <h1> в страницата.
- Заглавията трябва винаги да нарастват.
Лоша практика е да се прескачат заглавия <h1> и след това <h3>

Headings (Заглавия)

В едитора

отварящ таг

затварящ таг

`<h1>Заглавие h1</h1>`

съдържание

`<h2>Заглавие h2</h2>`

`<h3>Заглавие h3</h3>`

`<h4>Заглавие h4</h4>`

`<h5>Заглавие h5</h5>`

`<h6>Заглавие h6</h6>`

В браузъра

Заглавие h1

Заглавие h2

Заглавие h3

Заглавие h4

Заглавие h5

Заглавие h6

Paragraphs (Параграфи)

- Използваме ги за маркиране на блокове от текст.
- По подразбиране браузърите показват всеки параграф на нов ред, като оставят определено разстояние от горе и долу между тях и всеки следващ елемент.

отварящ таг

съдържание

затварящ таг



`<p>` Блок от текст, който маркираме като параграф `</p>`

Paragraphs (Параграфи)

В едитора

<p>

Lorem ipsum dolor sit amet,
consectetur adipisicing elit.

</p>

<p>

Repellendus consequatur
obcaecati, maxime similitque
accusantium.

</p>

В браузъра

Lorem ipsum dolor sit
amet, consectetur
adipisicing elit.

Repellendus
consequatur obcaecati,
maxime similitque
accusantium.

Links (Линкове)

- Използваме линкове за връзка / препращане към друга част от страницата, към друга страница или външна препратка.
- Потребителя клика между отварящия и затварящия таг

отварящ таг съдържание затварящ таг

```
<a href = "www.google.com"> www.google.com </a>
```

страницата, която искаме да отворим текста на който потребителя клика

Links (Линкове) Атрибути

- Атрибута **title** ползваме за подсказващ текст.
- Атрибута **target** ползваме за да окажем къде да бъде отворена страницата в текущата страница или в нова.
_blank , **_self** ...

отварящ таг

```
<a href = "http://google.com" title = "Google" target = "_blank" >  
    www.google.com  
</a>
```

затварящ таг

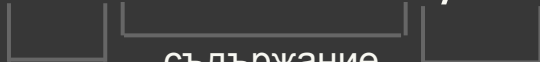
Bold & Italic (удебелен и курсив)

- **** правим маркираният текст да се покаже в удебелен вид.

В едитора

<p> Lorem ipsum dolor sit amet, ****consectetur**** adipisicing elit.**</p>**

****consectetur****



отварящ таг

затварящ таг

В браузъра

Lorem ipsum dolor sit amet, **consectetur** adipisicing elit.

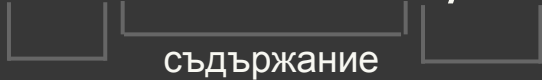
Bold & Italic (удебелен и курсив)

- `<i>` правим маркираният текст да се наклони на дясно.

В едитора

`<p>Lorem ipsum dolor sit amet, <i>consectetur</i> adipisicing elit.</p>`

`< i > consectetur </ i >`



отварящ таг

затварящ таг

В браузъра

Lorem ipsum dolor sit amet,
consectetur adipisicing elit.

Superscript и Subscript

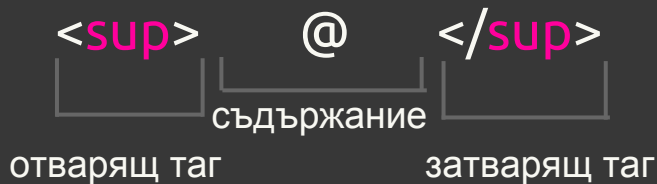
- `<sup>` правим маркираният текст да се качи на горе спрямо основният текст.

В едитора

`<p>Lorem ipsum dolor sit amet, [@] adipiscing elit.</p>`

В браузъра

Lorem ipsum dolor sit amet,[@] adipiscing elit.



Superscript и Subscript

- `<sub>` правим маркираният текст да се смъкне на долу спрямо основният текст.

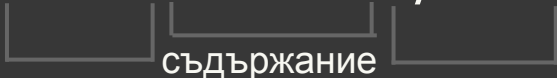
В едитора

`<p>`Lorem ipsum dolor sit amet, `_{`sub text`}` adipisicing elit.`</p>`

В браузъра

Lorem ipsum dolor sit amet, sub text adipisicing elit.

`_{` sub text `}`



отварящ таг

затварящ таг

Line Breaks и Horizontal Rules

- Ползваме `
` тага, ако искаме да накараме конкретен текст / елемент да отиде на нов ред.

В едитора

`<p>Lorem ipsum dolor sit
amet,
new line
adipisicing elit.</p>`

В браузъра

Lorem ipsum dolor sit
amet,
new line adipisicing elit.

Line Breaks и Horizontal Rules

- Ползваме `<hr>` тага, ако искаме да направим нов ред и визуално да го отделим с хоризонтална черта.

В едитора

`<p>`Lorem ipsum dolor sit amet, adipisicing elit.`</p>`

`<hr>`

`<p>`Deleniti consectetur laudantium quaerat ullam culpa.`</p>`

В браузъра

Lorem ipsum dolor sit amet, adipisicing elit.

Deleniti consectetur laudantium quaerat ullam culpa.

Strong & Emphasis

- `` тага прави същото, като `` тага но с разликата, че предава семантично значение на текста.

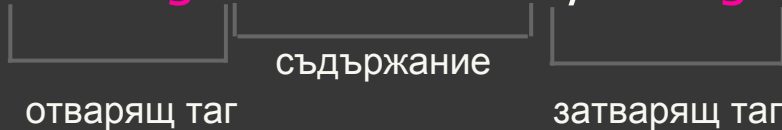
В едитора

`<p>Lorem ipsum dolor sit amet,
consectetur
adipiscing elit.</p>`

В браузъра

Lorem ipsum dolor sit amet,
consectetur adipiscing elit.

` consectetur `



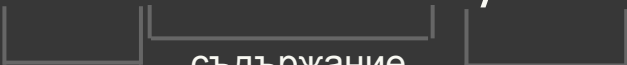
Strong & Emphasis

- `` правим същото, като `<i>` тага, но предава семантично значение на маркираният текст.

В едитора

`<p>` Lorem ipsum dolor sit amet, ``consectetur`` adipisicing elit.`</p>`

`` consectetur ``


отварящ таг затварящ таг

В браузъра

Lorem ipsum dolor sit amet, *consectetur* adipisicing elit.

Lists (Видовете Списъци)

Имаме случай, когато трябва да използваме списъци. HTML ни предоставя три основни типа списъка:

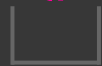
- Подреден списък
- Не подреден списък
- Дефиниран списък <dl>

Lists (Не подреден списък)



отварящ таг

 Стъпка 1



съдържание



отварящ таг

затварящ таг

Стъпка 2

Стъпка 3

Стъпка 4

съдържание



затварящ таг

В браузъра

- Стъпка 1
- Стъпка 2
- Стъпка 3
- Стъпка 4
- Стъпка 5



bullet

Lists (Дефиниран списък <dl>)

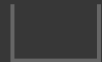
Съставен е от термини последвани от техните дефиниции.

<dl>



отварящ таг

<dt> Термин 1</dt>



съдържание



отварящ таг

затварящ таг

<dd>Дефиниция</dd>



съдържание
съставено от
последователни
термини и
техните
дефиниции

В браузъра

Термин
Дефиниция

</dl>



затварящ таг

Lists (Вложени списъци)

```
<ul>
  <li> Стъпка 1 </li>
  <li>
    <ul>
      <li>Стъпка 2.1</li>
      <li>Стъпка 2.2</li>
    </ul>
  </li>
  <li>Стъпка 3</li>
</ul>
```

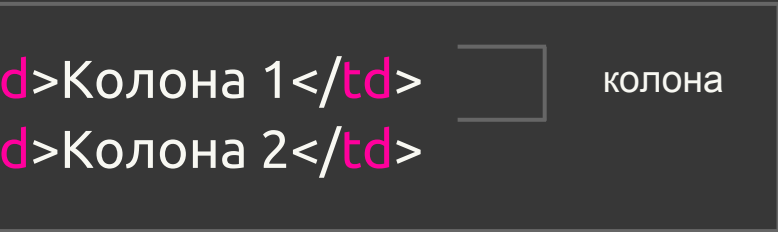
В браузъра

- Стъпка 1
 - Стъпка 2.1
 - Стъпка 2.2
- Стъпка 3

Таблицы <table>

- Таблиците ги ползваме за представяне на комплексни данни, които имат нужда да бъдат в някакъв грид.
- Таблиците са съставени от редове и колони / клетки.

```
<table>  
  <tr>  
    <td>Колона 1</td>  
    <td>Колона 2</td>  
  </tr>  
</table>
```



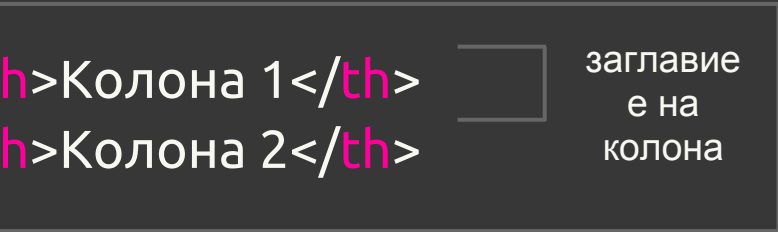
колона

ред

Таблицы <table>

- <th> тага го ползваме, както <td> с разликата, че той определя заглавията на колоните.

```
<table>  
  <tr>  
    <th>Колона 1</th>  
    <th>Колона 2</th>  
  </tr>  
</table>
```



заглавие
е на
колона

ред

Таблицы <table>

Имаме три основни типа тагове, които ни позволяват да разграничим основното съдържание от първата и последната колона: <thead> , <tbody> , <tfoot>

<thead>

<tr>Колона 1</tr>

заглаванта част на таблицата

</thead>

<tbody>

<tr>Колона 1</tr>

основната част на таблицата

</tbody>

<tfoot>

<tr>Колона 1</tr>

footer часта на таблицата

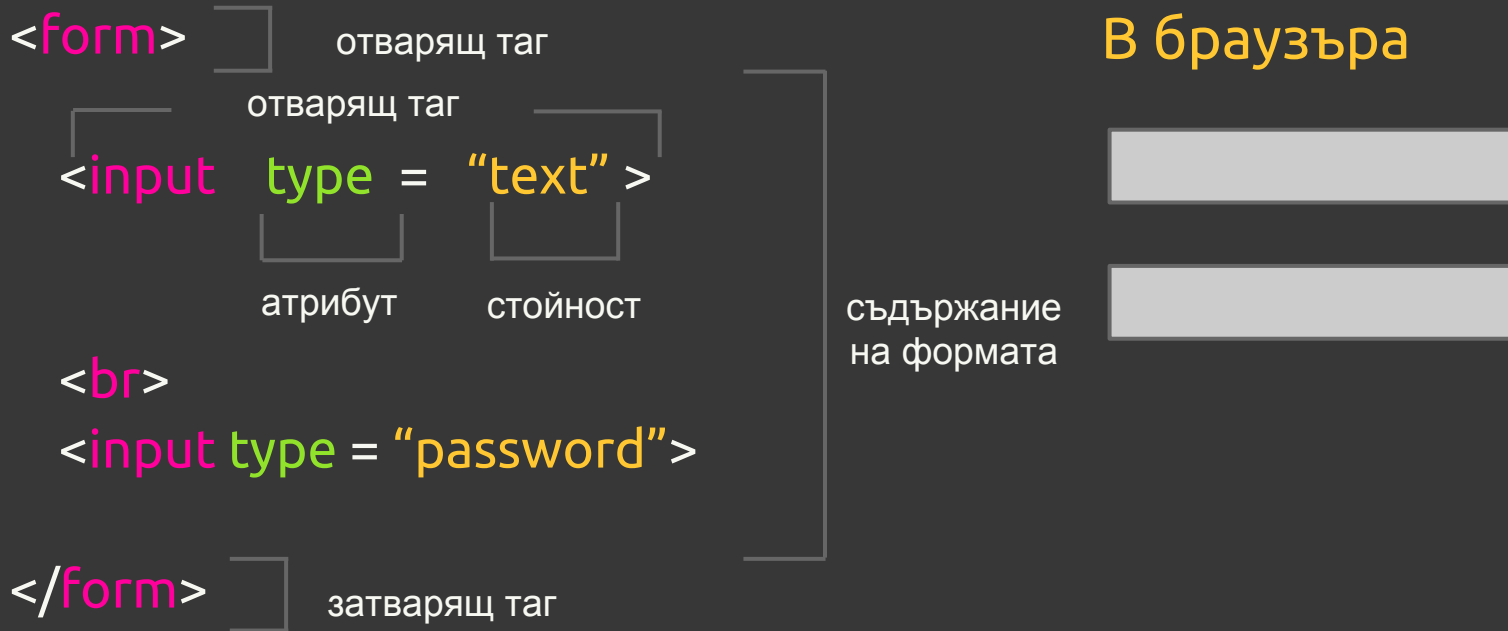
</tfoot>

Forms (Форми)

- Използваме формите за да събираме информация от потребителя.
- С `<form>` тага определяме полето за формата.
- Най - често използваме `<input>` тага, който има много разновидности зависещи от `type` атрибута.
- `<textarea>` използваме за много-редово и неограничено текстов въвеждане.
- `<select>` тага ползваме за падащи менюта

Form и Input тагове

Както знаем `<input>` тага е празен таг и е Inline елемент.



Form и Input тагове

`<label>` ползваме за да подскажем на потребителя какъв е вида на полета, което трябва да попълни.

```
<form>
  <label> Текст </label>
  <br>
  <input type = "text" >
  <br>
  <label> Парола </label>
  <br>
  <input type = "password">
</form>
```

В браузъра

Текст

Парола

Видове Input тагове според типа

`<form>`

`<input type = "text">`

`<input type = "email">`

`<input type = "password">`

`<input type = "radio">`

`<input type = "checkbox">`

`<input type = "button">`

`<input type = "date">`

`<input type = "color">`

`<input type = "file">`

`</form>`

И още много други
може да ги
разгледате

[http://www.w3schools.
com/tags/att_input_type.
asp](http://www.w3schools.com/tags/att_input_type.asp)

Атрибути на Input тага

отварящ таг

```
<input type = "text" name = "text" value = "Текст" >
```

отварящ таг

```
<input type = "text" name = "text" placeholder = "Текст" >
```

```
<label for = "text"> Въведете Текст </label>
```

```
<input type = "text" id = "text" placeholder = "Текст" >
```

<textarea> и <select> тагове

отварящ таг затварящ таг

```
<textarea placeholder = "Оставете коментар"></textarea>
```

отварящ таг затварящ таг

```
<select>  
  <option value = "Колело">Колело</option>  
</select>
```

съдържание

```
<option value = "Кола">Кола</option>  
<option value = "Мотор">Мотор</option>  
</select>
```

Comments (Коментари в HTML)

<!-- Коментар таговете ги ползваме когато искаме да оставим насоки за дадения код без този насоки да се показват в браузъра -->

<!-- div таг в който имаме p таг и глупав текст за запълване -->

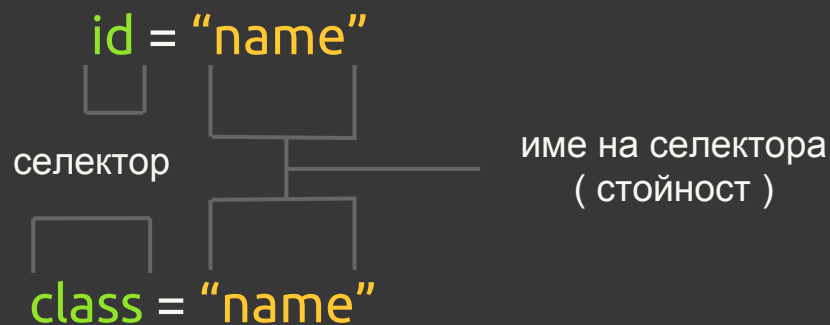
<div>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>

</div>

Attribute: **id** и **class** на HTML елементите

Селекторите **id** и **class** ни позволяват да приложим определени стилове върху HTML елементите, които имат тези селектори. Използваме ги за референция към CSS стиловете.



Разликата между **id** и **class** селектора е, че **id** с *конкретното* име може да се ползва само веднъж, докато **class** се ползва за идентифициране на много елементи.

Attribute: **id** и **class** на HTML елементите

Стойността трябва да започва с буква или с долна черта, но не може да започва с цифри или друг символ, различен от букви.

Важно е да се запомни, че не може да има едно и също **id** с еднаква стойност на една и съща страница.

id и **class** са известни, като глобални атрибути, защото могат да се използват на всеки един HTML елемент.

Attribute: `id` и `class` на HTML елементите

Може да записваме повече от една стойност в `class` атрибутите:

`class = "first-class second-class"`

Не може да имаме повече от една стойност за `id`. Няма да сработи, когато го тестваме.

~~`id = "first-id second-id"`~~

`id = "first-id"`

CSS - Cascading Style Sheets

Тук трябва да се наблегне на думата каскаден защото CSS стиловете работят на принципа (който е най - долу той е с приоритет).

```
p { color: red; }
```

```
p { color: green; }
```

 - параграфа, ще се оцвети в **зелено**.

Със CSS ние създаваме правила за това, как съдържанието на елементите да бъде визуализирано.

CSS - Cascading Style Sheet

Първата версия на езика CSS 1 е била пусната през 1996 г. CSS 2 е пуснат две години по - късно през 1998 г.

Това е език за стилизиране и описване на това как да изглежда и бъде форматиран маркиращ език.

Основното нещо, за което е създаден езика е да раздели съдържанието на документа от презентационната част.

Това разделение на съдържанието от форматирането ни позволява да представим един и същи маркър по различни начини.

CSS - Cascading Style Sheet

CSS стиловете работят на принципа (който е най - долу той е с приоритет).

html

<p> параграф </p>

css

```
10  p { background-color: blue; }  
13  p { background-color: red; }  
27  p { background-color: green;  
    }
```

browser



параграф

Начини за вмъкване на CSS в HTML

Имаме три основни начина, по които можем да включим / вмъкнем CSS за да стилизираме HTML таговете:

- External style sheet - външен файл;
- Internal style sheet - на самата страница;
- Inline style - на самият ред;

External style sheet

Външният CSS е най - добрият от трите начина поради ред причини:

- Когато искаме да стилизираме html на всички страници.
- Променяме стила на едно място и той се отразява на всички страници.
- CSS кода е изчистен и подреден, не е смесен с HTML кода.

External style sheet

- Външният CSS се добавя по следният начин:

празен таг - има само отварящ без затварящ таг

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

rel

тага посочва връзката
между текущия документ
и свързания документ

с link тага казвам
на брауъра къде
да намери файла

type

тага посочваме вида на
документа

оказваме, че
съответният
документ е стил
лист

href

тага посочва пътя
до нашият файл

текстов или css файл

External style sheet

- Може да имаме повече от един CSS файл.
- CSS файлат не трябва да съдържа никакъв HTML код в себе си.

`<head>`

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

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

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

`</head>`

External style sheet

- Има значение реда, в който се подреждат файловете!
- Последният е с по - голям приоритет от първият:
icons.css ще е с най - голям приоритет, след това **ie.css**
и с най - нисък е **main.css**

```
<head>  
  <link rel="stylesheet" type="text/css" href="main.css">  
  <link rel="stylesheet" type="text/css" href="ie.css">  
  <link rel="stylesheet" type="text/css" href="icons.css">  
</head>
```

съдържащ всички CSS



Internal style sheet

Може да го използваме, когато имаме една страница, която изисква минимално оформление.

- Internal css е с по - голям приоритет от External css.
- Обикновено го пишем в `<head>` тага.
- Ако имаме повече от една страница трябва да го копираме на всяка една страница.
- За всяка следваща страница трябва да пишем / копираме css за да го преизползваме, както да дописваме и премахваме ненужният.

Internal style sheet

Вътрешният CSS стил го пишем в `<style>` тага.

`<head>`

`<style>`



отварящ таг

`h1 { background-color: blue; }`

`.box { background-color: green; }`

`.button { font-size: 24px; }`

`#wrapper { width: 100%; }`

CSS

`</style>`



затварящ таг

`</head>`

Inline style

С inline style ние губим от предимствата на Internal и External стиловете:

- Можем само и единствено да стилизираме html тага, върху който прилагаме Inline стила.
- Губим наследяването и много функционалността на вътрешните и външните стилове.
- Html таговете стават много претрупани и трудни за разчитане.
- Inline стила е с най - голям приоритет от трите.

Inline style

Inline style го пишем, като в html тага добавим атрибута **style** и в него пишем нашите CSS стиловете.

style атрибута използваме за inline стил в текущият html тага

пишем CSS стиловете между отварящата и затварящата кавичка

```
<h1 style = "background-color: green ; font-size: 24px ; ">
```

първият CSS стил

вторият CSS стил

Примерно Заглавие

```
</h1>
```

всеки CSS стил винаги завършва с ;

CSS вмъкване и техният приоритет

Както разбрахме имаме три начина за вмъкване на CSS в Html.

- External style sheet - външен файл;
- Internal style sheet - на самата страница;
- Inline style - на самият ред;

С най - голям приоритет е Inline след това Internal и на последно място е External стила.

CSS вмъкване и техният приоритет

Какво ще рече това с най - голям приоритет?

Нека да разгледаме следния код:

```
<head>
  <link rel="stylesheet" type="text/css" href="main.css">
  <style>
    h1 { color: red; font-size: 36px; }
  </style>
</head>
```

main.css

```
h1 {
  color: blue;
  font-size: 18px;
}
```

```
<h1 style = "color: green; font-size: 24px;">Примерно  
Заглавие</h1>
```

CSS вмъкване и техният приоритет

Как го разбира браузъра или как го подрежда?

Нека си представим, че имаме един CSS файл, който е виртуален и всички други се обединяват в него.

`h1 { color: blue; font-size: 18px; }` External CSS (`main.css`)

`h1 { color: red; font-size: 36px; }` Internal CSS

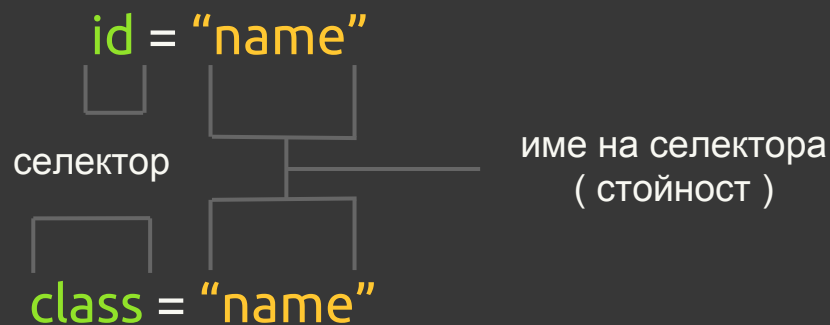
`h1 { color: green; font-size: 24px; }` Inline CSS

в браузъра

Примерно
Заглавие

Attribute: **id** и **class** на HTML елементите

Селекторите **id** и **class** ни позволяват да приложим определени стилове върху HTML елементите, които имат тези селектори. Използваме ги за референция към CSS стиловете.



Разликата между **id** и **class** селектора е, че **id** с *конкретното* име може да се ползва само веднъж, докато **class** се ползва за селектиране на много елементи.

Attribute: `id` и `class` на HTML елементите

Стойността трябва да започва с буква или с долна черта но не може да започва с цифри или друг символ различен от букви.

Важно е да се запомни, че не може да има едно и също `id` с еднаква стойност на една и съща страница.

`id` и `class` са известни, като глобални атрибути, защото могат да се използват на всеки един HTML елемент.

Селекторите `id` и `class` са чувствителни относно главните и малките букви: `.big-title` и `.Big-title` са различни класове.

#id

You are **special**.

Attribute: **id** и **class** на HTML елементите

Може да записваме повече от една стойност в **class** атрибутите, като ги разделяме с интервал:

```
class = " first-class second-class another-class "
```

Не може да имаме повече от една стойност за **id**.

```
id = " first-id second-id "
```

```
id = " first-id "
```

Начини за записване на **id** и **css** имена

Както знаем вече имената трябва да започва с буква или с долна черта и са чувствителни спрямо главните и малки букви и не може да започва с цифри или друг символ различен от букви.

Naming conventions:

- camelCase - конвенция, която е съставена от много думи обединени в една, като първата дума започва с малка буква, а всяка следваща с Главна без да се оставя разстояние между тях.

Стандартно изречение: The quick brown fox jumps over the lazy dog.

Изречение в CamelCase: theQuickBrownFoxJumpsOverTheLazyDog

Начини за записване на **id** и **css** имена

Naming conventions:

- BEM (Block__Element–Modifier)

```
<div class="block block--modifier">  
  <p class="block__element"> ... </p>  
</div>
```

- Title CSS - Глобалните стилове пишем с първа Главна, а за всички класове, които променят главният клас, както и за потомците с първа малка буква.

```
<div class="Title isModified">  
  <p class="descendant"> ... </p>  
</div>
```

Начини за записване на **id** и **css** имена

Naming conventions:

- SMACSS (Scalable and Modular Architecture for CSS)
- OOCSS (Object Oriented CSS)
- Има и още други начини за именуване на класовете, със задълбочаване в материята ще си изберете или създаде свои методи. За начало нека да стартираме с обикновено именуване на имената:

```
<div class="block remove-border">  
  <p class="block-paragraph"> ... </p>  
</div>
```

Как пишем **html** тага, **id** и **class** в CSS?

html

```
<h1 id = "title" class = "big-title">
```

Заглавие

```
</h1>
```

CSS

```
h1 { border: 1px solid red; }
```

```
#title { border: 1px solid blue; }
```

```
.big-title { border: 1px solid yellow; }
```

browser

Заглавие

**Какъв ще е
цвета на
контура?**

Как пишем **html** тага, **id** и **class** в CSS?

html

```
<h1 id = "title" class = "big-title">
```

Заглавие

```
</h1>
```

CSS

```
h1 { border: 1px solid red; }
```

```
#title { border: 1px solid blue; }
```

```
.big-title { border: 1px solid yellow; }
```

browser

Заглавие

Защо е син?

Ще научим в
следващите слайдове ;

Структура на селекторите: **html**, **id**, **class**

```
h1 { color: #3EBA77 ; }
```

селектор

декларация

```
#title { font-size : 24px ; }
```

идентификатор за

id

селектор

декларация

```
.big-title { font-size : 24px ; }
```

идентификатор за

• **class**

Изброяване на селектори

`h1, h2, #title, .big-title, .box {`



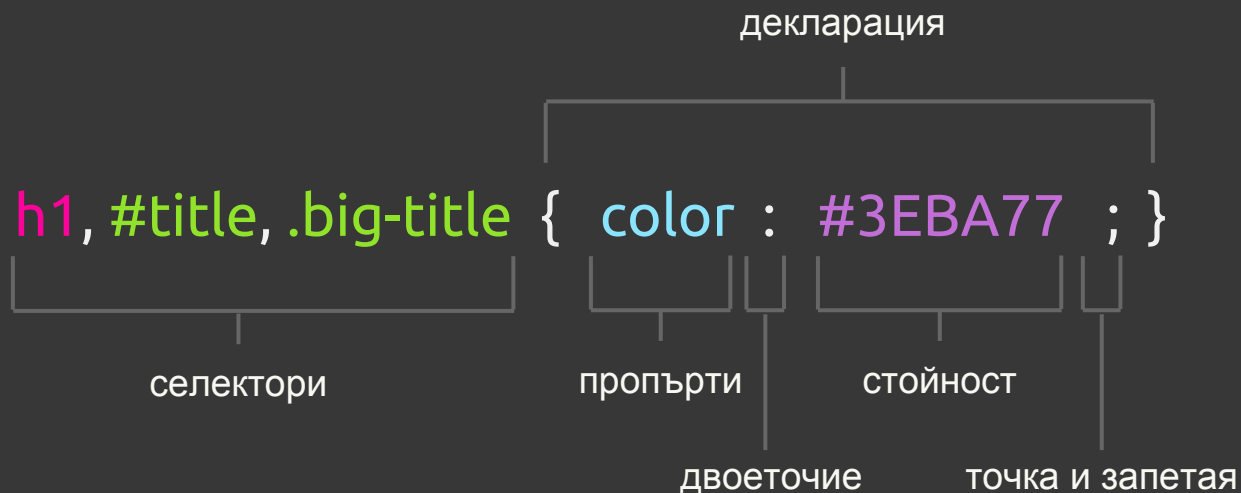
разделяме селекторите със запетая ,

```
color: #3EBA77;  
}
```

Можем да изреждаме много селектори, които да бъдат причислени към една декларация на CSS, като ги изреждаме един след друг и ги разделяме със запетая “,”

Структура на декларацията

Декларацията на CSS е между двете къдрави скоби и е изградена от две части: **пропърти** и **стойност**, разделени от двоеточие. Всяка CSS декларация завършва с точка и запетая ;



Тежест на селекторите в CSS

Всеки един CSS селектор си има тежест или приоритет. Тежестта определя кой точно стил ще бъде приложен от браузъра.

style атрибут = 1, 0, 0, 0

class = 0, 0, 1, 0

id = 0, 1, 0, 0

html елемент = 0, 0, 0, 1



style атрибут

,



id

,



class

,



html елемент

Тежест на селекторите в CSS

На коя позиция трябва да отиде?

h1 = 1



style атрибут

,



id

,



class

,

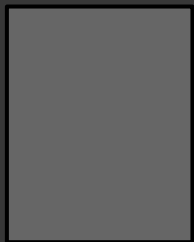


html елемент

Тежест на селекторите в CSS

На коя позиция трябва да отиде?

h1 = 1



style атрибут

,



id

,



class

,



html элемент

Тежест на селекторите в CSS

Колко е резултата и на коя позиция трябва да го впишем?

h1 span



style атрибут

,



id

,



class

,



html элемент

Тежест на селекторите в CSS

Колко е резултата и на коя позиция трябва да го впишем?

h1 span



style атрибут

,



id

,



class

,



html элемент

Тежест на селекторите в CSS

Колко е резултата и на кои позиция трябва да ги впишем?

h1.title



style атрибут

,



id

,



class

,



html элемент

Тежест на селекторите в CSS

Колко е резултата и на кои позиция трябва да ги впишем?

h1.title



style атрибут

,



id

,



class

,



html элемент

Тежест на селекторите в CSS

Колко е резултата и на кои позиция трябва да ги впишем?

h1.title span



style атрибут

,



id

,



class

,



html элемент

Тежест на селекторите в CSS

Колко е резултата и на кои позиция трябва да ги впишем?

h1.title span



style атрибут

,



id

,



class

,



html элемент

Тежест на селекторите в CSS

Колко е резултата и на кои позиция трябва да ги впишем?

h1**#section-title** **.title** **span**



style атрибут

,



id

,



class

,



html элемент

Тежест на селекторите в CSS

Колко е резултата и на кои позиция трябва да ги впишем?

h1**#section-title** **.title** **span**



style атрибут

,



id

,



class

,



html элемент

Тежест на селекторите в CSS

Колко е резултата и на кои позиция трябва да ги впишем?

```
<h1 id="section-title" class="big-title" style="color: red">  
    DON'T MISS YOUR CHANCE!  
</h1>
```



style атрибут

,



id

,



class

,



html элемент

Тежест на селекторите в CSS

Колко е резултата и на кои позиция трябва да ги впишем?

```
<h1 id="section-title" class="big-title" style="color: red">  
    DON'T MISS YOUR CHANCE!  
</h1>
```



style атрибут

,



id

,



class

,



html элемент

Изчислете следните тежести:

1. `.photography div a` = ?
2. `ul#nav li.active a` = ?
3. `<p style="color: red"> ... </p>` = ?
4. `div.box div.wrapper #wrapper ul > li a.link` = ?
5. `<p style="color: red">`
 ` ... ` = ?
 `</p>`

```
.big-title {  
  font-size: 50px;  
  color: blue !important;  
}
```

```
<p id="color-text">  
  <span class="big-title"> ... </span>  
</p>
```

Кой от стиловете ще се отрази?

`<p id="food">...</p>`

1. `#foo { background: blue; }`
`p { background: green; }`

`<h1 class="title">...</h1>`

1. `.title { color: #fff; }`
`h1 { color: #ccc !important; }`

`<div class="pop">`
 `<p>...</p>`
 `<p class="corn">...`
`</p>`
`</div>`

1. `.pop { background: red; }`
2. `.pop p.corn {`
 `background: yellow;`
`}`

CSS селектори

Селектор	Пример	Обяснение
1. <code>*</code>	<code>*</code>	Селектира всички елементи.
2. <code>html</code> елемент	<code>p</code>	Селектира всички p елементи.
3. елементи	<code>div, p</code>	Селектира всички div и p елементи.
4. <code>id</code>	<code>#section</code>	Селектира id с име section.
5. <code>class</code>	<code>.title</code>	Селектира всички класове с име title.
6. <code>:hover</code>	<code>a:hover</code>	Когато мишката мине над елемента.

[css селектори](#) и [The 30 CSS Selectors you Must Memorize](#)

Наследяване в CSS

Както в HTML имаме йерархия, всеки един елемент се съдържа от друг:

*р има родител **div** елемента, а той и има за родители **body** и **html**.*

```
<html>
  ...
  <body>
    <p> ... </p>
  </body>
</html>
```

html { color: blue; } - *р* *така*
ще наследи цвета от html
родителя си.

body { color: green; }

Така и в CSS имаме наследяване на класове. Родителите придават стилове към потомците си.

Наследяване в CSS

```
<html>
  ...
  <body>
    <div>
      <p> ... </p>
    </div>
    <p> ... </p>
  </body>
</html>
```

```
html { color: blue; }
```

1. Какъв ще е цвета на *div p* ?
2. Какъв ще е цвета на *p* ?

```
body { color: green; }
```

3. Какъв ще е цвета на *div p* ?
4. Какъв ще е цвета на *p* ?

```
div p { color: yellow; }
```

5. Какъв ще е цвета на *div p* ?
6. Какъв ще е цвета на *p* ?

Наследяване в CSS

```
<html>
  ...
  <body>
    <div>
      <p> ... </p>
    </div>
    <p> ... </p>
  </body>
</html>
```

```
html { color: blue; }
```

1. Цвета на *div p* ще е син.
2. Цвета на *p* ще е син.

```
body { color: green; }
```

3. Цвета на *div p* ще е зелен.
4. Цвета на *p* ще е зелен.

```
div p { color: yellow; }
```

5. Цвета на *div p* ще е жълт.
6. Цвета на *p* ще е зелен.

[https://docs.google.
com/presentation/d/1Z_PRA9aLBfpeK7IC4iJ
WG35zeyz3kAeA-GXxqJLIR2o/edit?
usp=sharing](https://docs.google.com/presentation/d/1Z_PRA9aLBfpeK7IC4iJWG35zeyz3kAeA-GXxqJLIR2o/edit?usp=sharing)

[https://drive.google.com/file/d/0B-
69QOUwAhUsV2hXWW40V0Jta1E/view?
usp=sharing](https://drive.google.com/file/d/0B-69QOUwAhUsV2hXWW40V0Jta1E/view?usp=sharing)